JOHANNES KEPLER
UNIVERSITÄT LINZ
Netzwerk für Forschung, Lehre und Praxis

Mobile intelligente Agenten als Wegweiser im Distance Teaching/Coaching/Learning

DISSERTATION

zur Erlangung des akademischen Grades

DOKTOR DER TECHNISCHEN WISSENSCHAFTEN

Angefertigt am *Institut für Informationsverarbeitung und Mikroprozessortechnik (FIM)*

Betreuung:
o.Univ.-Prof. Dr. Jörg R. Mühlbacher

Eingereicht von:
Dipl.-Ing. Susanne Reisinger

Gutachter
o.Univ.-Prof. Dr. Jörg R. Mühlbacher
Prof. Dipl.-Ing. Dr. Helmut Schauer

Linz, September 2002

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Dissertation selbständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt bzw. die wörtlich oder sinngemäß entnommenen Stellen als solche kenntlich gemacht habe.

Linz, im September 2002

Reisinger Susanne

Danksagung

Niemand schreibt eine Dissertation allein. Immer gibt es Menschen, die einem direkt oder indirekt zur Seite stehen und Hilfestellung bieten.

Besonders bedanken möchte ich mich daher bei Herrn o. Univ.-Prof. Dr. Jörg R. Mühlbacher, der zu diesem Thema anregte, immer wieder neue Impulse, unterschiedliche Sichtweisen, etc. aufwarf und diese Dissertation stetig betreut hat.

Auch Prof. Dipl.-Ing. Dr. Helmut Schauer sei gedankt für die Möglichkeit zum Gedankenaustausch und die dadurch entstandenen Ideen, die in diese Dissertation eingeflossen sind.

Weiters möchte ich mich bei Mag. Dr. Michael Schröder bedanken, der mich bei der Umsetzung des Prozessmodells unterstützt hat.

Auf der anderen Seite wurde mir aber auch indirekt geholfen. Damit meine ich eine persönliche Hilfestellung, die mir nicht nur während dieser Dissertation entgegengebracht wird, sondern auch in anderen Phasen meines Lebens.

Daher möchte ich mich an dieser Stelle bei meinen Eltern Brigitte und Johann Reisinger bedanken, die mir nicht nur ein Studium ermöglichten, sondern mir mein gesamtes Leben lang immer mit Rat und Tat zur Seite standen und stehen.

Auch Dipl.-Ing. Reinhard Loidl sei für seine Begleitung und Unterstützung gedankt und dafür, dass er immer ein offenes Ohr und wertvolle Ratschläge für mich hat.

Bedanken möchte ich mich auch bei Freunden und Kollegen, die nicht alle namentlich erwähnt werden können, deren Unterstützung und Hilfe mir jedoch sehr wichtig war.

Herzlichen Dank

Reisinger Susanne

KURZFASSUNG

Diese Dissertation hat ihren Fokus auf Agenten und Distance Education und der Kombination dieser beiden Themen.

Zunächst werden Agenten und Distance Education umfassend erläutert. Zum Thema Agenten werden grundlegende Informationen in den Bereichen Definitionen, Anwendungsgebiete, geschichtliche Entwicklung, Agentensysteme und -plattformen, agentenorientiertes Software-Engineering, Agentensprachen, etc. aufgezeigt und detailliert beschrieben.

Weiters werden Aspekte der Distance Education (DE) beschrieben. Unterschiedliche Lernmodelle, Vor- und Nachteile, wichtige Elemente, die Durchführung, etc. werden dabei erläutert. Weiterführend wird auf eine konkrete Umsetzung von DE im Rahmen eines Projekts (PIAF – Projekt Informatik als Fernstudium) eingegangen. Speziell aufbereitetes Lernmaterial, Settings, die Lernplattform WeLearn und Pilotstudien werden dabei vorgestellt.

Aufgrund der Erfahrungen im Bereich der Agenten und der Distance Education wird eine Zusammenführung dieser beiden Themen aufgezeigt. Dabei wird zunächst ein für DE-Kurse entwickeltes Prozessmodell dargestellt, in dem die notwendigen Prozessschritte und Aufgaben von der Kursvorbereitung, über die Kursdurchführung bis hin zur Kursnachbereitung modelliert und beschrieben wurden. Im Prozessmodell wird auch die Rolle von Agenten und deren Einsatzmöglichkeiten detailliert dargelegt.

Weiters wurde eine agenten-orientierte Wissensplattform (knowledge platform for agents – KNOWPAG) entwickelt, die in dieser Dissertation beschrieben wird. Dessen Elemente, wie XML-Schema, prototypisches Web-Interface und vor allem Agententypen und Agentenprotokolle werden dabei ausführlich dargestellt. Das XML-Schema spezifiziert die Struktur von Metadaten für DE-Kurse. Darin ist nicht nur die Struktur exakt beschrieben, sondern auch Datentypen und Defaultwerte sind dadurch definiert. Alle Komponenten (Datenbanken, Protokolle, Agenten, etc.) basieren auf diesem Schema. Dadurch werden eventuelle Missinterpretationen und Inkompatibilitäten ausgeschlossen. An konkreten Agenten wird der generelle Lebenszyklus eines Agenten, die Agentenkommunikation und der Austausch und die Gewinnung von Daten aufgezeigt. Abschließend werden Agenten zur DE-Kurssuche vorgestellt, die auf den Elementen von KNOWPAG basieren.

ABSTRACT

This thesis focuses on the topics of Agent Technology and Distance Education and their combination.

Firstly, the main aspects of Agents and Distance Education are described. With respect to agents this thesis discusses fundamental information in the areas of definitions, application areas, historical development, agent systems and platforms, agent-oriented software engineering, agent languages, etc.

Secondly, the thesis treats important aspects of Distance Education (DE). Different learning models, advantages and disadvantages, important elements, development and realization of DE-Courses, etc. are covered herein. In addition Distance Education courses developed and delivered in the context of a project (PIAF – Projekt Informatik als Fernstudium) are mentioned. Purposely developed learning material, Learning arrangements (Settings), the WeLearn learning platform and pilot studies are presented thereby.

This thesis specifically addresses the topic of applying Agents in the field of Distance Education based on the aforementioned experiences. Firstly, a process model developed for Distance Education courses is presented, in which the necessary process steps and tasks of course development, course realization and course evaluation were modeled and described. The role of agents and their possible tasks are also presented in this process model.

Secondly, an agent-oriented knowledge platform (KNOWPAG) developed in the context of this thesis is presented. Its elements, like an XML schemata (definition of the structure and the type of meta data), a prototype of a web-interface and, foremost, different types of agents and agent protocols are presented in detail thereby. The XML schemata specifies the structure of metadata for DE-courses. Within this not only the structure, but also the data types and predefined values are defined. All components (databases, agent protocols, agents, etc.) are based on this XML schema. Therefore misinterpretations or incompatibilities are excluded. Several sample agents are presented in this context upon which the agent life cycle, the communication between agents and the exchange and retrieving of data are demonstrated and explained in general. Finally agents for searching DE-courses are presented which work based on the elements of KNOWPAG.

Inhaltsverzeichnis

1. EINLEITUNG 1

2. AUSGEWÄHLTE THEMEN & ÜBERBLICK ÜBER AGENTEN 4

2.1. DEFINITION EINES AGENTEN 4

2.2. KLASSIFIKATION VON AGENTEN 7

2.2.1. EIGENSCHAFTEN 7

2.2.2. AGENTENTYPEN 10

2.2.3. AGENTENFUNKTIONEN 13

2.3. GESCHICHTE DER AGENTEN 14

2.4. AKTUELLE FORSCHUNGSBEREICHE UND ANWENDUNGSGEBIETE 17

2.4.1. AGENTENTHEORIEN 17

2.4.2. ANWENDUNGSGEBIETE 18

2.4.2.1. System- und Netzwerkmanagement 18

2.4.2.2. Intelligente Benutzerschnittstellen 19

2.4.2.3. Informationsmanagement 20

2.4.2.4. "Workflow Management"/Arbeitsgruppen 21

2.4.2.5. Electronic Commerce 22

2.4.2.6. Mobile Computing - Persönliche Digitale Assistenten 22

2.4.2.7. Weitere Einsatzgebiete 24

2.5. AGENTEN: MÖGLICHER NUTZEN, VORTEILE, PROBLEME UND UNGELÖSTE FRAGEN 25

2.5.1. MÖGLICHE VORTEILE UND NUTZEN 25

2.5.2. MÖGLICHE PROBLEME UND UNGELÖSTE FRAGEN 26

3. AGENTEN-PORTALE/PLATTFORMEN 29

3.1. DEFINITION EINES PORTALS/EINER PLATTFORM 29

3.2. PORTALE 29

3.2.1. ZWECK 29

3.2.2. UNTERSCHIEDLICHE AUSPRÄGUNGEN- TAXONOMIE VON PORTALTYPEN 30

3.2.2.1. Horizontale Portale – Mega-Portale 30
3.2.2.2. Vertikale Portale - Vortale 31
3.2.3. AUFBAU EINES PORTALS 34
3.3. AGENTENSYSTEME/AGENTENPLATTFORMEN 35
3.3.1. MULTIAGENTENSYSTEME (MAS) 36
3.3.1.1. Definition 36
3.3.1.2. Agentenkommunikation 37
3.3.1.3. Interaktionsprotokolle 41
3.3.1.3.1. Organisatorische Strukturierung 41
3.3.1.3.2. Vertragsnetzprotokoll 42
3.3.1.3.3. Multiagentenplanung 43
3.4. BEZIEHUNG PORTAL UND AGENTENPLATTFORM 44
3.4.1. INFORMATIONSMANAGEMENT 44
3.4.2. PERSONALISIERUNG 46
3.4.3. EREIGNISÜBERWACHUNG UND AUTOMATISIERUNG 46

4. AGENTEN ALS NEUES SOFTWAREENGINEERING-PARADIGMA 48

4.1. GEGENÜBERSTELLUNG VON MODELLEN 48
4.1.1. CLIENT-SERVER-MODELL 48
4.1.2. OBJEKTMODELL 49
4.1.3. AGENTENMODELL 51
4.1.4. AGENTEN VERSUS OBJEKTE 51
4.2. AGENTENORIENTIERTE SOFTWAREENTWICKLUNG (AOSE) 53
4.3. WERKZEUGE UND METHODEN 56
4.3.1. AGENTENKOMMUNIKATIONSSPRACHEN (AGENT COMMUNICATION LANGUAGES – ACL) 56
4.3.1.1. Knowledge Query and Manipulation Language – KQML 58
4.3.1.2. Knowledge Interchange Format – KIF 61
4.3.2. SPRACHEN ZUR IMPLEMENTIERUNG VON AGENTEN 62
4.3.2.1. JAVA 63
4.3.2.2. C# 64
4.3.2.3. Telescript, Odyssey 65

4.3.2.4. Tcl, Safe Tcl, Agent Tcl, D´Agents 66
4.3.3. XML-FAMILIE 67

5. DISTANCE EDUCATION 79

5.1. FORMEN VON DISTANCE EDUCATION 82
5.2. WARUM DISTANCE EDUCATION? 85
5.3. MÖGLICHE VOR- UND NACHTEILE VON DISTANCE EDUCATION 89
5.3.1. MÖGLICHE VORTEILE VON DISTANCE EDUCATION [MUE02B] 89
5.3.2. MÖGLICHE NACHTEILE VON DISTANCE EDUCATION [MUE02B] 91
5.4. WICHTIGE ELEMENTE IN DER DISTANCE EDUCATION 92
5.4.1. LERNUMGEBUNG – PLATTFORM 92
5.4.2. KURSSTRUKTUR UND LERNMATERIAL 94
5.4.3. DIDAKTISCHES MODELL 96
5.4.4. KOMMUNIKATION UND INTERAKTION 99
5.4.5. FEEDBACK 102
5.5. UMSETZUNG VON DISTANCE EDUCATION 103
5.5.1. WICHTIGE ÜBERLEGUNGEN FÜR DIE VORBEREITUNG 103
5.5.2. DURCHFÜHRUNG UND REFLEXION 105

6. PIAF- PROJEKT INFORMATIK IM FERNSTUDIUM 107

6.1. BEGRIFFSBESTIMMUNG IM RAHMEN DER PILOTSTUDIEN VON PIAF 107
6.1.1. DISTANCE TEACHING 107
6.1.2. DISTANCE LEARNING 107
6.1.3. DISTANCE COACHING 108
6.2. WELEARN – WEB ENVIRONMENT FOR LEARNING 110
6.2.1. LERNPLATTFORM 110
6.2.2. EINGEBETTETES KURSMATERIAL 113
6.2.3. LERNARRANGEMENTS – SETTINGS 114
6.3. PILOTSTUDIEN VON PIAF 115
6.3.1. LEHRERAUS- UND FORTBILDUNG 115
6.3.2. PROPÄDEUTIKUM AUS INFORMATIK AN DER JKU LINZ 116

6.3.3. EINFÜHRUNG IN DIE BETRIEBSSYSTEME AN DER JKU LINZ 117
6.3.4. WEITERE LEHRVERANSTALTUNGEN, ABGEHALTEN AN DER JKU LINZ 117
6.3.5. VIRTUELLE KOMMUNIKATION AN DEN UNIVERSITÄTEN VON ZÜRICH, MÜNCHEN UND GRAZ 117
6.3.6. PROPÄDEUTIKUM AUS INFORMATIK FÜR SCHÜLER (IM ALTER VON 14 BIS 16) AN ZWEI VERSCHIEDENEN HÖHEREN SCHULEN 118
6.3.7. LEHRGANG AKADEMISCHEN CHARAKTERS AN DER FH VORARLBERG FÜR BERUFSTÄTIGE ZUR WEITERBILDUNG 118

7. EINSATZ VON AGENTEN IM BEREICH DER DISTANCE EDUCATION – AGENTEN-ORIENTIERTE WISSENSPLATTFORM (KNOWLEDGE PLATFORM FOR AGENTS - KNOWPAG) 120

7.1. VISION 120
7.2. EINSATZMÖGLICHKEITEN VON AGENTEN IN DISTANCE EDUCATION KURSEN 121
7.2.1. PROZESSMODELL: DISTANCE EDUCATION KURS 122
7.2.1.1. Beschreibung der Zeichnungselemente 122
7.2.1.2. Das Prozessmodell 124
7.3. AGENTEN-ORIENTIERTE WISSENSPLATTFORM (KNOWLEDGE PLATFORM FOR AGENTS - KNOWPAG) 180
7.3.1. ZIELE 180
7.3.2. XML-SCHEMATA 181
7.3.2.1. Exkurs: Basiskonzepte 182
7.3.2.1.1. Exkurs: File-Schemata 182
7.3.2.1.2. Exkurs: VDL – Vienna Definition Language 184
7.3.3. INFORMATIONEN ÜBER DISTANCE EDUCATION KURSE 185
7.3.3.1. Beschreibung von Bildungsangeboten 187
7.3.3.1.1. XML-Schema: Agentenprotokoll – WeLearnCourseMetaData 187
7.3.4. AGENTEN 207
7.3.4.1. Agent 1: WEMemberDataAgent 207
7.3.4.2. Agent 2: CourseMDAgent 225

8. ZUSAMMENFASSUNG 229

Abbildungsverzeichnis

Abbildung 1: Agentenbeschreibung nach [RUS95] 5
Abbildung 2: Modell eines Agenten nach [CAG98] 6
Abbildung 3: Einflussgebiete nach [BRE98] 7
Abbildung 4: Klassifikation von Agenten [NWA96a] 10
Abbildung 5: Teilansicht der Agententopologie nach [NWA96a] 12
Abbildung 6: Dreidimensionale Charakterisierung von Agenten nach [IBM96] 13
Abbildung 7: Einsatz von Agenten beim Netzwerkmanagement [MAG96] 19
Abbildung 8: Informationsmanagement [JEN96] 20
Abbildung 9: Business Process Management [JEN96] 21
Abbildung 10: Beispiel eines elektronischen Marktplatzes [HOR] 22
Abbildung 11: Einsatz von Agenten beim Mobile Computing [ROT97] 23
Abbildung 12: Büro der Zukunft [BOD] 24
Abbildung 13: Multiagentensystem – das Ganze ist mehr als die Summe seiner Teile [JAC01] 36
Abbildung 14: Interaktionstypen in Multiagentensystemen 37
Abbildung 15: Anordnung von Blöcken (Blocks World Problem) [NWA96b] 39
Abbildung 16: Beispiel eines Informationsmultiagentensystems [SYC96] 40
Abbildung 17: Client-Server-Modell 48
Abbildung 18: Objekte als Instanzen einer Klasse 49
Abbildung 19: Einfache Vererbung und Mehrfachvererbung 50
Abbildung 20: Polymorphismus 50
Abbildung 21: Gegenüberstellung der Fähigkeiten Objekt und Agent [BURa] 52
Abbildung 22: Schichtaufbau einer KQML-Nachricht 58
Abbildung 23: Strukturierter Aufbau von Mitgliedsdaten 68
Abbildung 24: Anzeige der erzeugten HTML-Ausgabe der Inhalte (Version 1) 75
Abbildung 25: Anzeige der erzeugten HTML-Ausgabe der Inhalte (Version 2) 77
Abbildung 26: Funktionsweise eines DOM-Parsers 77
Abbildung 27: Funktionsweise eines SAX-Parsers 78
Abbildung 28: Einsatz von E-Learning derzeit (n=98) [UNI] 79
Abbildung 29: Anteile der Mitarbeiter in eLearning-Maßnahmen (n=40) [UNI] 79
Abbildung 30: Anteil eLearning-Budget (n=42) [UNI] 80
Abbildung 31: Schulungsthemen für eLearning heute (n=71) [UNI] 80
Abbildung 32: Schulungsthemen für eLearning zukünftig (n=69) 81
Abbildung 33: Hemmnisse von eLearning (n=78) [UNI] 81
Abbildung 34: Chancen von eLearning (n=76) [UNI] 82
Abbildung 35: Unterschiedliche Formen von Unterrichtssystemen [TEL98] 83
Abbildung 36: Traditionelle Auffassung vom Lehren und Lernen [MAN01] 86
Abbildung 37: Wissens- und Kompetenzmanagementprozess [LED01] 87

Abbildung 38: Konstruktivistische Auffassung vom Lernen [MAN01] 88
Abbildung 39: Wichtige Aspekte einer Lernplattform [UNI] 93
Abbildung 40: Unterschiedlich Arten von Lernmaterialstrukturierung 94
Abbildung 41: Planung einer Lernumgebung [ROT01] 104
Abbildung 42: Möglicher Ablauf einer Distance Education Veranstaltung 106
Abbildung 43: Erscheinungsbilder von WeLearn 111
Abbildung 44: Technischer Aufbau von WeLearn [REI02] 111
Abbildung 45: Typisches WeLearn-Setting für ein hybrides Lernmodell [REI02] 114
Abbildung 46: Darstellung von Aktivitäten 122
Abbildung 47: Mögliche Symbole in Prozessdiagrammen 123
Abbildung 48: Übersicht über das Prozessmodell: Distance Education Kurs 125
Abbildung 49: Prozessdiagramm „Kursvorbereitung“ 126
Abbildung 50: Detaildiagramm „Wahl der Werkzeuge“ 128
Abbildung 51: Prozessdiagramm „Entwicklung, Wahl, Aufbereiten des Lernmaterials“ 133
Abbildung 52: Detaildiagramm „Lernmaterial ergänzen“ 134
Abbildung 53: Prozessdiagramm „Umsetzung der Lernlandschaft“ 136
Abbildung 54: Prozessdiagramm „Kursdurchführung“ 139
Abbildung 55: Prozessdiagramm „Kick-Off“ 141
Abbildung 56: Detaildiagramm „Gruppenbildung“ 143
Abbildung 57: Prozessdiagramm „DE-Kursdurchführung“ 147
Abbildung 58: Prozessdiagramm „Kurs ist reiner Distance Education Kurs (ohne reale Treffen)“ 151
Abbildung 59: Prozessdiagramm „Kurs ist ein DE-Kurs im hybriden Modell (teilweise online, teilweise reale Treffen)“ 155
Abbildung 60: Detaildiagramm „Veranstaltet Treffen“ 155
Abbildung 61: Prozessdiagramm „Kurs ist traditionell geführt mit DE Online-Unterstützung“ 157
Abbildung 62: Detaildiagramm „Veranstaltet periodisch reale Treffen“ 157
Abbildung 63: Prozessdiagramm „DE-Kurs Abschluss“ 160
Abbildung 64: Detaildiagramm „Legt Abschluss des DE-Kurses fest“ 161
Abbildung 65: Prozessdiagramm „Abschlussveranstaltung“ 163
Abbildung 66: Prozessdiagramm „Prüfung/Zertifizierung“ 165
Abbildung 67: Prozessdiagramm „Kursnachbereitung“ 167
Abbildung 68: Rollen im Prozessmodell Distance Education Kurs 170
Abbildung 69: Informationssysteme DE-Kurs 177
Abbildung 70: Beispiel einer Liste nach [WED73] 183
Abbildung 71: Schema für Personaldaten nach [WED73] 183
Abbildung 72: Baumdarstellung des VDL-Formalismus für ein Objekt Person (nach [MUE75]) 185
Abbildung 73: Graphische Darstellung der WeLearnCourseMetaData-Struktur 192
Abbildung 74: WEMemberDataAgent im POND Agentensystem 208
Abbildung 75: Auswahl der Funktionen des Agenten WEMemberDataAgent 208

Abbildung 76: Dialog für ID-Eingabe 208
Abbildung 77: Zwei getrennte Agentensysteme 209
Abbildung 78: WEMemberDataAgent ist zum anderen Agentensystem migriert 209
Abbildung 79: Agent ist zum Ausgangsagentensystem zurück migriert 210
Abbildung 80: Dialog mit den gewünschten Benutzerdaten 210
Abbildung 81: Zweite Funktion des Agenten 211
Abbildung 82: Eingabe der ID´s 211
Abbildung 83: Dialog mit den noch offenen Gebühren eines Benutzers 212
Abbildung 84: Eingabe der Kreditkartendaten des Benutzers 213
Abbildung 85: Rückmeldung durch den WEMemberDataAgent über den Status der Bezahlung der offenen Gebühren 213
Abbildung 86: Lebenszyklus und Kommunikationsablauf der Agenten 214
Abbildung 87: Datenbank-Tabelle für die Mitgliedsdaten 216
Abbildung 88: Datenbankeinträge der Mitgliedsdaten 217
Abbildung 89: Aufbau der Strukturen, Ablauf des Datenaustauschs und der -gewinnung 224
Abbildung 90: Web-Interface des CourseMDAgent 226
Abbildung 91: Web-Interface des CourseMDAgent mit den gewünschten Benutzerdaten 227
Abbildung 92: Web-Interface mit den gefundenen Kursen 228

Tabellenverzeichnis

Tabelle 1: Aufgaben von Agenten in den einzelnen Einsatzgebieten 24
Tabelle 2: Unterschiede zwischen Objekten und Agenten 53
Tabelle 3: Kategorien von KQML-Nachrichten 59
Tabelle 4: Gegenüberstellung von instruktionalen Ereignissen in Lehr- und Lernsystemen [BRI92] 89
Tabelle 5: Typische Bestandteile von Lernangeboten in hybriden Lernarrangements [KER02] 97
Tabelle 6: Internetgestützte Kommunikationsformen [COS] 100

VORWORT ZUR ARBEIT

Ich möchte vorausschicken, dass in der Dissertation personenbezogene Formulierungen generell in der grammatikalisch männlichen Form dargestellt sind, sprich: der Benutzer, der Lernende, etc. Gemeint sind damit aber immer beide Geschlechter. Auf eine Geschlechtsneutralität, wie der Benutzer/die Benutzerin, der Lernende/die Lernende, der Entwickler/die Entwicklerin, etc. wurde aufgrund einer flüssigeren Lesbarkeit des Textes verzichtet.

1. EINLEITUNG

Viele Arbeitsbereiche die erst durch den Einsatz von Informationstechnologie erschlossen wurden bzw. in denen Informationstechnologie ein bestimmender Faktor ist, werden heutzutage durch die Vielfalt der Möglichkeiten derselben zunehmend komplexer und unüberschaubarer. Die Mannigfaltigkeit der Informationstechnologie ist nicht nur eindimensional, sondern mehrdimensional zu sehen. Für die einzelnen Einsatzgebiete gibt es verschiedene Werkzeuge (Softwareprodukte), die sich in ihren Funktionen, Darstellungsformen und letztendlich in der Bedienbarkeit durch den Benutzer unterscheiden.

Dazu kommt noch eine stetig zunehmende Menge von elektronischen Daten. Dies begründet sich zum einen aus der Tatsache heraus, dass heute die elektronische Datenverarbeitung Gang und Gebe ist und Dokumente zumeist nicht mehr mittels „Papier und Bleistift“, sondern mittels „Computer und Textverarbeitungsprogrammen“ verfasst werden. Zum anderen haben wir uns zu einer vernetzten Gesellschaft entwickelt. Als solche beschäftigen wir uns nicht mehr ausschließlich mit unseren eigenen Dokumenten und ggf. mit den Dokumenten und Informationsquellen unserer unmittelbaren Umgebung. Durch die Vernetzung ist die ganze Welt zu unserer unmittelbaren Umgebung geworden und wir haben die Möglichkeit uns quasi alle existierenden Informationsquellen zu Nutze zu machen und in ihnen zu „stöbern“.

Wie schon Goethe [GOE01] seinem Götz von Berlichingen sagen lässt „Wo viel Licht ist, ist starker Schatten“. Ebenso vorhält es sich mit den neuen Möglichkeiten und Freiheiten der Informationstechnologie. Die neuen und erweiterten Einsatzmöglichkeiten bringen nicht nur Vorteile mit sich, sondern auch Nachteile. So erwirkt man in manchen Bereichen eine Vereinfachung, andere Bereiche werden hingegen wieder komplexer. Manches wird simpler in der Handhabung, manches wird komplizierter und fehleranfälliger. In bestimmten

Aspekten gewinnt man eine breitere Datenbasis, mit steigender Datenmenge jedoch kann gerade diese breitere Datenbasis sehr rasch unübersichtlich und diffus werden und andere Hilfsmittel müssen eingesetzt werden um sich erneut Übersicht zu schaffen.

An dieser Stelle soll zum besseren Verständnis jeweils ein Beispiel gegeben werden:

- Hinsichtlich der Vereinfachung: Durch die Vernetzung haben wir nicht nur die Möglichkeit auf verschiedene Informationsquellen, sondern auch auf unterschiedliche Hardware (z.B. Drucker, Scanner, etc.) innerhalb eines Netzwerks zuzugreifen. Einerseits wird das Arbeiten im Netzwerk durch Informationstechnologie erleichtert (wir haben je nach Rechtevergabe Zugriff auf mehrere Bereiche des Netzwerks und diverseste Hardware, die Administratoren können über das Netz die Rechner warten, Rechte vergeben, etc.). Andererseits, wird gerade im Bereich der Netzwerk-administration wieder komplex, durch die Mannigfaltigkeit der möglichen Konfigurationen, durch explizite Rechtevergaben, durch implizite Rechte mittels Vererbung und nicht zuletzt auch durch unterschiedliche Zugriffsmöglichkeiten (Lese-/Schreib-/Ausführungsrechte, Einzel- und Gruppenrechte).

- Einfacher in der Handhabung: Die heutigen Emailprogramme bieten über das „simple“ Senden und Empfangen von Emails hinaus noch eine Reihe von zusätzlichen Möglichkeiten. So kann man Emails bereits vorfiltern, in unterschiedliche Ordner kopieren/verschieben, etc. Die zusätzlichen Möglichkeiten können dem Benutzer die Handhabung der täglichen Korrespondenz wesentlich erleichtern, aber gerade für ungeübte Benutzer kann dadurch die Anwendung wieder komplexer werden oder zu einer Fehlbedienung verleiten, wie etwa unabsichtliches Löschen wichtiger Emails durch Setzen von mehrdeutigen Filteregeln.

- Breitere Datenbasis: Das Internet ist wohl die größte Bibliothek unserer Zeit. Sie umfasst Millionen von Dokumenten zu nahezu allen erdenklichen Themen. Dieses elektronische Medium hat aber nicht nur den Vorteil der schier unerschöpflichen Daten, es hat auch den Vorteil, dass diese Daten in elektronischer Form vorliegen und somit für jedermann einfach und schnell abrufbar sind. Aber genau in dieser Daten-fülle liegt auch sein der Nachteil. Nicht nur, dass jeder Daten abrufen kann, jeder kann auch Daten zur Verfügung stellen. So nimmt die Datenmenge täglich zu. Aufgrund des Fehlens eines Standards oder gar einer Reglementierung, wie Daten strukturiert,

dargestellt bzw. formatiert sein sollen, wird die Suche nach Informationen immer aufwendiger und zeitintensiver.

Um der Komplexität entgegen zu wirken und um uns in unserer Arbeit, Freizeitbeschäftigungen und Wissensdrang zu unterstützen, wünschen wir uns etwas, das uns unterstützt und „unter die Arme greift“. Etwas, das uns die damit verbundene Arbeit teilweise oder ganz abnimmt. Sprich, ein „Helferlein“, das uns behilflich ist, welches wir mit einer Aufgabe betrauen können und uns daraufhin die entsprechenden Resultate liefert ohne dass wir uns aktiv mit der Durchführung dieser Aufgabe beschäftigen müssen. Etwas, das uns Informationen beschafft, Käufe abschließt, Netzwerke auf mögliche Fehler überwacht und uns gegebenenfalls alarmiert, usw. Kurz gesagt wir wünschen uns einen Agenten!

Doberkat [Dob00] hat diesen Wunsch noch ausführlicher formuliert: „Stellen Sie sich vor, dass Sie beim Aufwachen, nicht von einem krähenden Hahn empfangen werden, sondern von der Mitteilung, dass die Brötchen frisch aufgebacken warten und der Kaffee frisch aufgebrüht ist. Weiterhin finden Sie Ihre Wohnung wohl temperiert vor, also im Winter angenehm geheizt, im Sommer angenehm gekühlt. Ein Blick auf das Display des Kühlschranks belehrt Sie, dass Sie noch ein wenig Marmelade für das Brötchen haben, es aber höchste Zeit wird, Marmelade und anderen Brotaufstrich zu kaufen. Ach ja: Die elektronische Einkaufsliste an ihrem Kühlschrank erinnert Sie daran, dass Ihr jüngster Sohn morgen Geburtstag hat und sich bestimmt über ein neues Buch freuen würde. [...] Natürlich haben Sie einen PC im Hause. Ihr Haus ist überdies vernetzt, sodass Ihren Agenten leicht die notwendigen Informationen zur Verfügung stehen. [...] Nehmen wir an, Ihr Agent der Sie an den Geburtstag des Sohns erinnert hat, trifft sich im Netz mit dem Agenten Ihres Sohnes. Ihr Agent schlägt als Geschenk ein Buch vor, die beiden Agenten unterhalten sich, dabei kommt zutage, dass das Bücherregal Ihres Sohnes ohnehin schon sehr voll ist. [...] ein weiterer plausibler Wunsch entsteht [..] (der) Ihnen mitgeteilt wird, nämlich dem Herrn Sohn ein Geldgeschenk zu machen. [...] Schließlich haben Sie einen Agenten in Ihren Diensten, der Sie in finanziellen Dingen berät. [...] (Es) ist aber festzustellen, dass mittelalterliche Gemälde insbesondere in englischen Versteigerungshäusern erhebliche Wertsteigerungen zu verzeichnen haben. Ihr Agent stellt fest, dass Sie Geld zu investieren haben, und er entscheidet, dass er für Sie ein mittelalterliches Madonnengemälde kauft. Diese Entscheidungen basieren auf einer Marktanalyse und der Beobachtung, dass Informationstechnikwerte an der Börse sehr starken Schwankungen unterworfen sein können.“

2. AUSGEWÄHLTE THEMEN & ÜBERBLICK ÜBER AGENTEN

2.1. Definition eines Agenten

Bevor hier versucht wird eine Definition für Agenten wiederzugeben, soll zunächst einmal eine Abgrenzung zu den Begriffen Robot (kurz: Bot) und Spider getroffen werden, die manchmal synonym auch für Agenten verwendet werden [PAL99].

Im Internet ist ein Roboter ein Programm, das eingesetzt wird, um zeitintensive Arbeiten zu erledigen, wie zum Beispiel das Durchsuchen von Newsgroups und speichern der gefundenen Informationen in eine Datenbank. Auch bei Suchmaschinen werden oft solche Programme eingesetzt, die hier Spider (Spinnen) genannt werden und in der Lage sind, selbständig WWW-Seiten zu besuchen, und die Inhalte dieser nach bestimmten Mustern zu durchsuchen [MTC01].

Intelligente Agenten sind technisch gesehen Robots. Allerdings sind sie aus mehreren Gründen eine Kategorie für sich. Zuallererst ist die Bezeichnung „Bot“ deshalb nicht angemessen, da man damit eher herkömmliche Programme verbindet, die weder besonders autonom noch intelligent sind. Zweitens ist zu sagen, dass obwohl intelligente Agenten zumeist autonom arbeiten, doch auch Interaktion mit dem Benutzer oder anderen Agenten stattfindet, um die gewünschten Ziele zu erreichen [PAL99].

Im Fischer Wirtschaftslexikon [FIS00] findet man als Definition für den Principal-Agent-Ansatz: moderner wirtschaftstheoretischer Ansatz, behandelt Probleme, die sich aus der Delegation von Aufgaben (innerhalb von Organisationen) ergeben, wenn der Principal (Auftraggeber) seinen Agenten (Auftragnehmer) nicht vollständig kontrollieren kann. Aus dieser Unvollkommenheit können Interessenkonflikte zwischen diesen beiden entstehen, die durch geeignete Anreizmuster zu verhindern sind. Beispiele solcher Principal-Agent-Beziehungen sind: Aktionäre und Manager, Patient und Arzt, Unternehmer und Arbeitnehmer.

Wenn wir diese Definition betrachten, haben wir es also mit Auftraggeber und Auftragnehmer zu tun, wobei allerdings nicht explizit darauf hingewiesen wird, dass es sich hier immer um menschliche Auftraggeber und –nehmer handelt. Im Sinne der Informatik können auch Softwareagenten die Aufträge entgegen nehmen und ausführen und ebenso können

Softwareagenten ihrerseits ebenfalls Aufträge an andere vergeben und werden somit auch zu Auftraggebern werden.

Wie kann man sich nun solche Softwareagenten vorstellen. Es sei vorausgeschickt, dass es zur Zeit keine allgemeingültige in der Informatik akzeptierte Definition für Agenten gibt. Deshalb sind die Definitionen zahlreich und auch der Fachausdruck als solches ist nicht genormt. Daher sprechen manche Forscher von Agenten, andere von Softwareagenten, mobilen Agenten, intelligenten Agenten, etc. und meist werden die Begriffe unterschiedlich gesehen und verstanden.

Wooldridge und Jennings et al. ([JEN96], [JEN98], [LUM01], [WOO97], [WOO98]) sprechen von einem Agenten, wenn die Hard- oder Software folgende Eigenschaften besitzt:

- Autonomie (autonomy): Agenten verfolgen unabhängig von einer direkten Intervention durch Menschen oder anderen Agenten ihre Ziele.
- Soziale Fähigkeit (social ability): Agenten kooperieren bei der Durchführung ihrer Aufgabe mit Menschen oder anderen Agenten.
- Reaktivität (reactivity): Agenten nehmen ihre Umwelt wahr und reagieren auf Änderungen.
- Proaktivität (proactivity): Agenten agieren nicht nur auf Ereignisse ihrer Umwelt, sondern ergreifen auch selbst die Initiative.

Russel und Norvig definieren Agenten als: “An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through effectors” [RUS95].

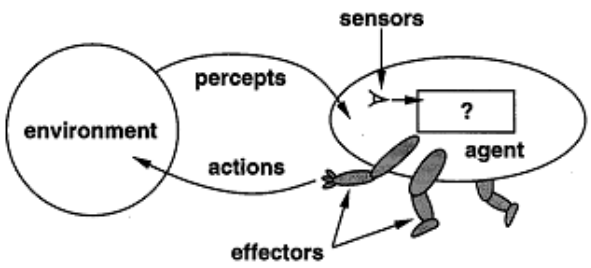


Abbildung 1: Agentenbeschreibung nach [RUS95]

Bei Caglayan [CAG98] findet sich ein Agentenmodell, welches das Verhältnis der funktionalen Architektur eines Agenten, das Zusammenspiel zwischen den einzelnen Systemkomponenten und die Interaktion mit dem Benutzer wiedergibt.

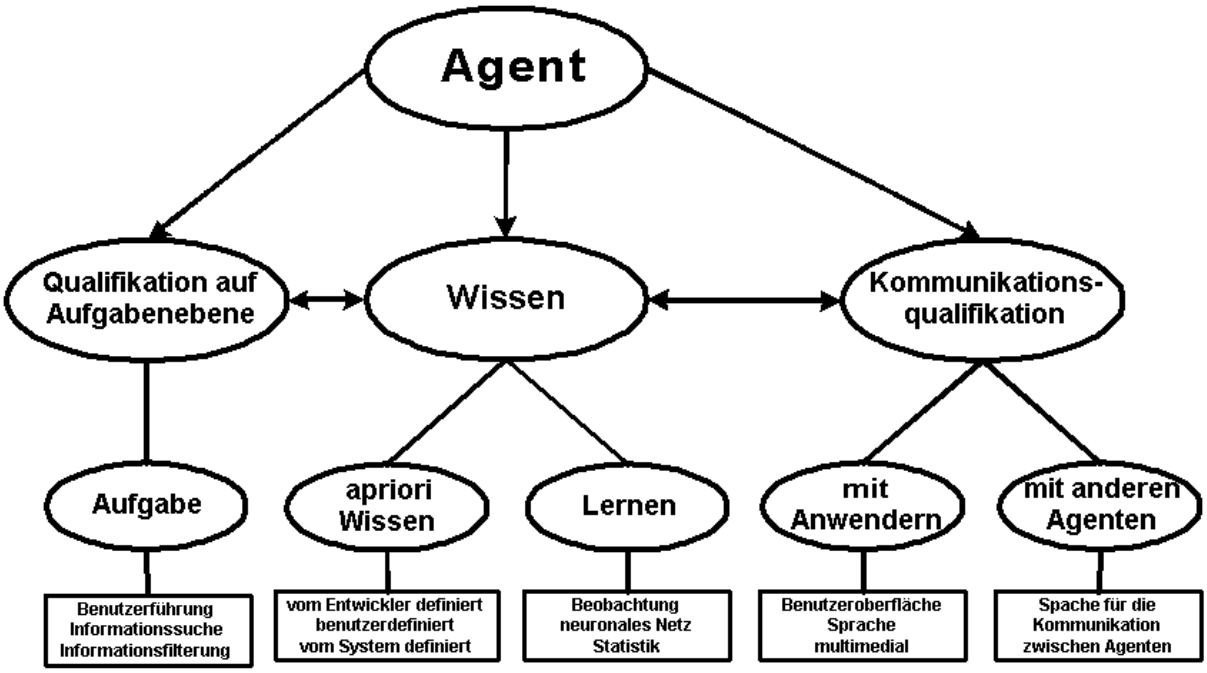


Abbildung 2: Modell eines Agenten nach [CAG98]

Aber auch in die Brockhaus Enzyklopädie hat der Begriff des Agenten bereits Einzug gehalten und die dort zu findende Definition stellt eine umfassende Beschreibung eines Agenten dar [BRO96]: **intelligenter Agent**, *Informatik:* im weiteren Sinn ein Dienst, der für seinen Nutzer Aufgaben, für die ein spezielles Fachwissen notwendig ist oder die aus vielen zeitaufwendigen Einzelschritten bestehen, größtenteils selbständig erledigt. Dabei unterscheidet man **Hardware-** (z. B. Roboter) und **Softwareagenten,** wobei Letztere meist als intelligente Agenten im engeren Sinn bezeichnet werden. Ein Softwareagent ist ein Programm, das für einen Benutzer bestimmte Aufgaben erledigen kann und dabei eine gewisse Intelligenz besitzt, die es befähigt, seine Aufgaben in Teilen autonom durchzuführen und mit seiner Umwelt auf sinnvolle Art und Weise zu interagieren. Je nach Zielsetzung lassen sich dabei **Informationsagenten** (Unterstützung bei der Suche nach Informationen in Netzen, z. B. Suchmaschinen), **Kooperationsagenten** (Lösung komplexer Problemstellung durch Kommunikation und Kooperation mit anderen Agenten, z. B. bei Groupware-Anwendungen) und **Transaktionsagenten** (Ausführung und Überwachung von Transaktionen, z. B. beim Electronic Commerce) unterscheiden.

2.2. Klassifikation von Agenten

Nachdem der Begriff des Agenten definiert ist, soll eine Klassifikation von Agenten wiedergegeben werden. Agent ist nicht gleich Agent. So können Agenten unterschiedliche Eigenschaften aufweisen, unterschiedlichen Typs sein bzw. sich in ihren Agentenfunktionen unterscheiden. Auf diese drei Klassifikationsmerkmale Eigenschaften, Typ und Funktionen wird in den nächsten Kapiteln detailliert eingegangen. Zunächst werden multiple Eigenschaften aufgezeigt, die Agenten aufweisen können. Danach wird eine Unterteilung nach Agententypen dargestellt und abschließend eine Klassifikation nach Agentenfunktionen getroffen.

2.2.1. Eigenschaften

Ebenso wie es zahlreiche Definitionen des Begriffs des Agenten gibt, gibt es auch unterschiedliche Auffassungen über die Eigenschaften, die Agenten besitzen müssen, um als Agenten im Sinne der Informatik zu gelten. Diese Eigenschaften von Agenten werden von unterschiedlichen Gebieten beeinflusst. Dabei leiten sich einige Eigenschaften, wie Lernfähigkeit, Proaktivität, Reaktivität, Kommunikation und Kooperation wesentlich von der (Verteilten) Künstlichen Intelligenz her. Andere Eigenschaften, wie Autonomie entspringen hauptsächlich der Entscheidungstheorie. Mobilität und auch Kommunikation werden stark vom Gebiet der Netzwerke und der Kommunikationswissenschaften beeinflusst. Dies und vor allem auch der starke Einfluss der Psychologie auf Eigenschaften, wie Reaktivität und Charakter, zeigt, dass nicht nur technische Wissenschaften, sondern auch Natur- und Geisteswissenschaften die Eigenschaften eines Agenten wesentlich prägen.

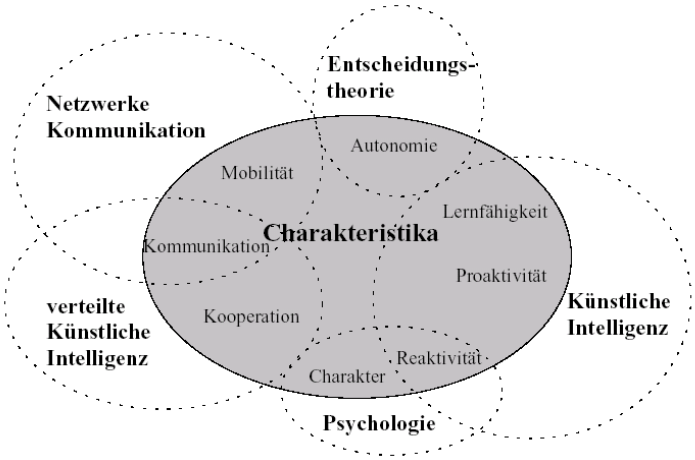


Abbildung 3: Einflussgebiete nach [BRE98]

Im folgenden sind hier einige der wichtigsten Eigenschaften von Agenten vorgestellt:

- Autonomie („autonomy“): Selbständiges ausführen von Aktionen, d.h. der Agent erfüllt eine Aufgabe ohne direkte Interventionen durch den Benutzer oder durch andere Agenten. Dies erfordert ein Zielsystem/-modell (sprich: was soll erledigt werden?) und die entsprechenden Fähigkeiten/Ressourcen, wie Kommunikationsmittel und Zugriffsmechanismen. Es gibt auch gewünschte Grenzen der Autonomie, wie etwa eine definierte Rückfrage, bevor Kaufentscheidungen getroffen werden. Beispiele für Autonomie sind etwa das Beschaffen (Suchen, Finden, Aufbereiten) bestimmter Informationen, die Sicherung von Daten zu einem bestimmten Zeitpunkt oder das Verhandeln über den Kauf eines bestimmten Produktes.

- Reaktionsfähigkeit („reactivity“): Reaktionsfähigkeit ist die Fähigkeit eines Agenten auf Aktionen seiner Umgebung reagieren zu können. Dabei wird unterschieden zwischen reaktiven Agenten mit Sensoren, die Umweltänderungen erkennen können und daraufhin Aktionen setzen und deliberativen Agenten, die ein Modell der Umwelt (mögliche Ereignisse, Vergleichswerte und logische Schlussfolgerungen) besitzen. Beispiel für Reaktivität wäre ein URL-Watcher-Agent. Diese Agenten beobachten permanent bestimmte Websites. Wenn sich Änderungen ergeben, werden diese vom Web heruntergeladen und der Benutzer des Agenten vom Agenten benachrichtigt.

- Proaktivität („proactivity“): Die Proaktivität geht über die Reaktionsfähigkeit hinaus. Der Agent reagiert nicht nur auf Geschehnisse seiner Umwelt, sondern setzt eigenständig Aktionen, bevor bestimmte Ereignisse eintreten. Der Agent ergreift selbst die Initiative.

- Kooperation, Koordination, Kommunikation („cooperation, coordination, communication“): Darunter versteht man, dass Agenten in der Lage sind mit anderen Agenten zusammenzuarbeiten um bestimmte Ziele zu erfüllen [JEN98]. Dies erfordert vor allem Fähigkeiten zur Kommunikation mit der Umwelt, eine Sprache/Protokoll zur Interaktion und die Fähigkeit zum Austausch oder zum Abgleich von Zielen und Wissensständen. Als Beispiel wäre hier ein Kalenderagent ([PAY02]) zu nennen. Jeder Agent kennt den Terminkalender seines Benutzers. Die Agenten können Vereinbarungen über Termine für eine Gruppe von Benutzern treffen, wobei die Präferenzen der einzelnen Benutzer berücksichtigt werden.

- Schlussfolgerungs- und Lernfähigkeit („ability to learn“): Dies heißt, Ergebnisse zu interpretieren, Entscheidungen zu treffen, Schlussfolgerungen anzustellen und zu planen. Der Agent muss seine Umwelt beobachten und bei Änderungen entsprechende Schlüsse ziehen, die ihn in Richtung Zielerfüllung weiterbringen. D.h. aber auch, dass der Agent aus Erfahrung lernen und auf die Wünsche und Präferenzen seines Benutzers acht geben muss. Dies erfordert eine Wissensbasis (ein Modell des Benutzers und der Umwelt), die Fähigkeit, basierend auf den Inhalten der Wissensbasis Schlussfolgerungen zu ziehen und entsprechende Anpassungen aufgrund der Veränderungen der Umwelt vorzunehmen. Ein Beispiel dafür wäre ein News-Agent. Dieser „lernt“, welche NetNews-Artikel für den Benutzer interessant sind anhand der durch den Benutzer ausgewählten Newsgroups und aus der Analyse der gelesenen Artikel.

- Mobilität („mobility“): Der Agent besitzt die Fähigkeit zur Migration zwischen verschiedenen Rechnern auf einem selbstausgewählten Weg. Dies erfordert zum einen eine entsprechende Infrastruktur (Hosts, mobiler Code) und zum anderen einen entsprechenden Schutz der transportierten Daten. Beim Schutz der Daten sind zwei Aspekte zu bedenken: Einerseits müssen die Daten des Agenten vor Angriffen durch Hosts und andere Agenten geschützt werden und andererseits müssen aber auch die Daten der Hosts durch Angriffe oder Manipulation durch Agenten geschützt werden. Beispiel hierfür wäre ein Suchagent, der von einem Laptop oder PDA (Personal Digital Assistent) gestartet wird, wenn dieser mit dem Firmennetz oder dem Internet verbunden (online) ist. Anschließend kann der Benutzer die Verbindung wieder lösen. Der Agent jedoch durchsucht selbständig das Firmennetz/Internet nach der gesuchten Information und zum Zeitpunkt, wenn der Benutzer wieder online geht, liefert der Agent seine Ergebnisse dem Benutzer.

- Sozialfähigkeit („social ability“): Agenten können unter Umständen zur Zielerreichung mit anderen Agenten und eventuell auch mit Menschen über eine Agentenkommunikationssprache („agent communication language“) kommunizieren. Die Kommunikation kann dabei explizit (durch den Austausch von strukturierten Nachrichten) oder implizit (über Effekte ihrer Aktionen auf den Zustand der Umwelt) erfolgen.

- Charakter („character“): Der Agent kann Züge menschlichen Charakters und eventuell auch Emotionen zeigen. Dem Benutzer soll dabei ein Gefühl der Glaubhaftigkeit vermittelt werden. Die Hauptmerkmale sind dabei: Charakter zu zeigen, Überzeugungen und Rechte zu haben, Gefühle ausdrücken zu können und ethische Verantwortung zu tragen.

Nicht alle Agenten müssen zwangsweise über alle angeführten Eigenschaften verfügen. Je nach Einsatzgebiet und Ausprägung (Typ des Agenten) rücken unterschiedliche Eigenschaften in den Vordergrund bzw. sind stärker ausgeprägt als andere. Aus diesem Grund sollen im folgenden kurz die unterschiedlichen Typen von Agenten angeführt werden.

2.2.2. Agententypen

Nwana [NWA96a] unterteilt die Agenten in sieben Agententypen:

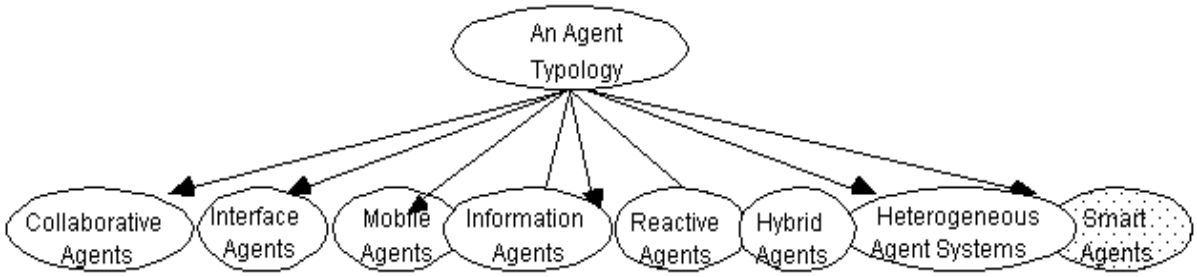


Abbildung 4: Klassifikation von Agenten [NWA96a]

Kollaborative Agenten

Kollaborative Agenten sind Agenten, die in der Lage sind selbständig (autonom) zu verhandeln, mit anderen Agenten zu kooperieren und auf Ereignisse ihrer Umwelt zu reagieren um ihre Aufgabe zu erfüllen. Daher muss es ihnen möglich sein, autonom ohne direkte Intervention durch ihre Benutzer oder andere Agenten zu agieren und dabei auch rational in einer offenen Umgebung, in der sich auch andere (möglicherweise konkurrierende) Agenten finden können, zu arbeiten.

Interface-Agenten

Das Wesen eines Interface-Agenten ist es einen Benutzer zu unterstützen. Ein Beispiel wäre, dass der Agent die Verwendung einer bestimmten Anwendung z.B. einer Kalkulationstabelle

oder eines Betriebssystems erlernt. Dabei beobachtet und überwacht er die Maßnahmen, die vom Benutzer getätigt werden (d.h. er schaut dem Benutzer quasi über der Schulter). Er lernt dabei Neues und schlägt gegebenenfalls dem Benutzer eine bessere Vorgehensweise zur Bewältigung der Aufgabe vor. Ein Interface-Agent ist somit ein autonomer persönlicher Assistent, der mit dem Benutzer bei der Erfüllung einer Aufgabe zusammenarbeitet und ihn unterstützt. Dabei ist die Lernfähigkeit des Agenten ein wesentliches Kriterium. Das Lernen kann üblicherweise durch Beobachtung und Nachahmung des Benutzers, durch Empfangen positiven und negativen Feedbacks des Benutzers, durch ausdrückliche Anweisungen vom Benutzer erfolgen oder dadurch, dass der Agent Fragen an andere Agenten richtet (eingeschränkte Form der Kooperation).

Mobile Agenten

Mobile Agenten sind in der Lage sich in einem Netzwerk zu bewegen und somit Aufgaben auf verschiedenen Rechnern bearbeiten zu können. Dies birgt den Vorteil in sich, dass dadurch vor allem die Netzlast reduziert wird und auch Ressourcen genützt werden können, die nicht zentral, sondern nur lokal auf bestimmten Rechner vorhanden sind. Um ihre Aufgabe zu lösen, können mobile Agenten im Allgemeinen ihren Weg durch die Netze selbständig (autonom) bestimmen, ohne damit den Benutzer zu belasten.

Informations- und Internet-Agenten

Bei der immer größer werdenden Flut an Information, die im Internet verfügbar ist, werden im zunehmenden Maße Werkzeuge benötigt, um gewünschte Information zu finden. Dabei ist es oft sehr aufwendig diese aus verschiedenen Quellen stammende Information zu finden, zu filtern und zu verwalten. Die Aufgabe von Informationsagenten besteht somit in der Entlastung der Benutzer bei der Suche nach Information. Der Informationsagent soll die gewünschte Information suchen, sammeln und aufbereiten. Dabei sollen auch persönliche Präferenzen der Benutzer, wie etwa, welche der gefundenen Ergebnisse für einen Benutzer tatsächlich relevant sind, oder wie diese Ergebnisse zu präsentieren sind, berücksichtigt werden. Informationsagenten handeln bei ihrer Suche weitgehend autonom. D.h. die Agenten setzen selbständig Aktionen, die zum Erreichen der Ziele führen sollen und reagieren entsprechend auf ihre Umgebung. Informationsagenten haben dabei auch im zunehmenden Maße die Fähigkeit der Kooperation mit anderen Agenten, um auch gegenseitig

Informationen austauschen zu können. Im Gegensatz dazu besitzen Informationsagenten in der Regel nur einen verhältnismäßig geringen Grad an Intelligenz [BRE98].

Reaktive Agenten

Reaktive oder reagierende Agenten besitzen kein fixes Modell ihrer Umwelt, sondern sie reagieren auf Reize, die von der Umgebung auf die Agenten ausgeübt werden. Diese Agenten sind deshalb einfach aufgebaut. Folgedessen haben sie ein sehr eingeschränktes Einsatzgebiet und werden spezifisch etwa in Simulationen oder in Spielen eingesetzt.

Hybride Agenten

Hybride Agenten vereinen die Eigenschaften mehrerer Agententypen. Das kann von Vorteil sein, da hier der Agent unterschiedliche Strategien anwenden kann, um ans Ziel zu kommen, sprich um seine Aufgabe zu erfüllen.

Smarte Agenten

Ein smarter Agent stellt die Idealform eines Agenten dar. Der smarte Agent ist autonom, kooperiert mit anderen Agenten, ist mobil und in der Lage selbständig zu lernen. Jedoch existiert ein solcher in der Realität noch nicht.

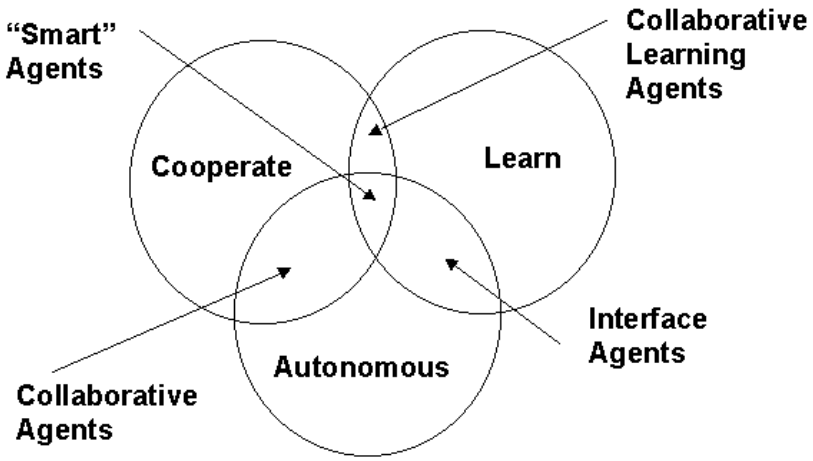


Abbildung 5: Teilansicht der Agententopologie nach [NWA96a]

2.2.3. Agentenfunktionen

Eine weitere Möglichkeit Agenten zu klassifizieren ist sie im Kontext von Agency, Intelligenz und Mobilität zu betrachten.

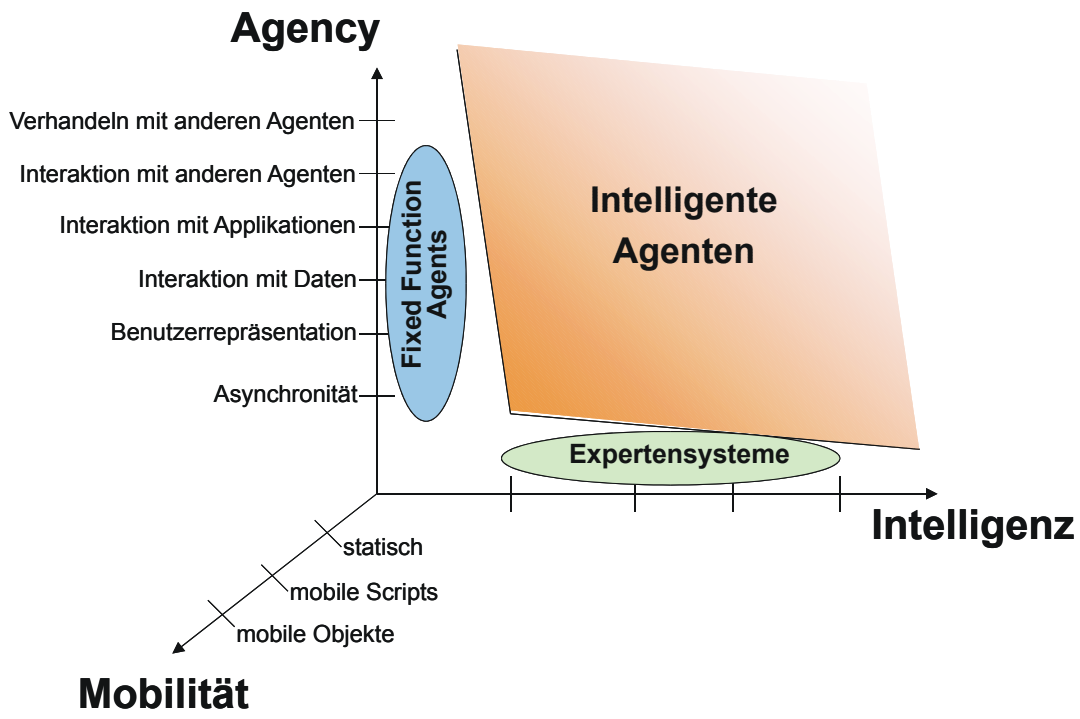


Abbildung 6: Dreidimensionale Charakterisierung von Agenten nach [IBM96]

Unter **Agency** versteht man den Grad an Autonomie, die ein Agent besitzt. Der Agent handelt dabei für den Benutzer, repräsentiert diesen also gegenüber Computersystemen, Applikationen und anderen Agenten. Der Agent soll Aufgaben für den Benutzer erfüllen, diesen helfen, unterstützen und zu möglichen Lösungen hinführen. Die Bandbreite an Agenten geht dabei von hart-codierten Agenten bis hin zu Agenten als intelligente Werkzeuge.

Intelligenz weist darauf hin, in wie weit ein Agent fähig ist zu lernen, Schlussfolgerungen zu treffen und mittels Wissensverarbeitung zu Lösungen zu kommen. Die Brandbreite bei der Agentenintelligenz ist auch hier enorm und reicht von einfacher Intelligenz, die über einfache logische Kodierung gelöst ist, bis hin zu komplexen Lern- und Adaptionsmethoden basierend auf künstlicher Intelligenz.

Mobilität ist dann gegeben, wenn der Agent in der Lage ist, sich zwischen Systemen eines Netzwerks zu bewegen. Gerade Mobilität erhöht die Komplexität eines Agenten, da damit auch Aspekte, wie die Sicherheit des Agenten und der Daten, der Schutz vor feindlichen Agenten und vor Datenmanipulation, etc. zu bedenken sind. Die Bandbreite in diesem Bereich reicht hier von statischen Agenten, Agenten die von System zu System wechseln können und mit den jeweiligen Systemdaten arbeiten können, bis hin zu Agenten, die in der Lage sind Daten beim Systemwechsel mit zu transportieren.

2.3. Geschichte der Agenten

Die Idee eines Agenten, der Aufträge für den Menschen übernimmt und ausführt gibt es schon lange. So hat der Prager Autor Karel Capek im Jahre 1921 bereits ein Theaterstück "Rossum´s Universal Roboter“ verfasst. Er wollte dabei den Unterschied zu einem gewöhnlichen Automaten betonen. Sein Roboter – ein künstliches Wesen, dass nicht im biologischen und chemischen Sinn, aber im mechanischen Sinn einem Menschen nachempfunden ist. In seinem Theaterstück zeigt Capek den Aufstand der Maschinenwesen, die die Welt übernehmen und die Menschheit zerstören.

Die Ansicht, dass man Probleme durch intelligente Maschinen lösen könne, geht auf Alan Turing zurück. Er hat im Jahre 1948 vorhergesagt, wie künstliche Intelligenz künftig aussehen könnte. Er meinte, dass die Intelligenz von Maschinen auf formaler Logik basieren würde, auf der Anwendung logischer Deduktionen. Aber gleichzeitig sprach sich Turing dafür aus Wissen als Grundlage für die Maschinenintelligenz zu benützen. Heute stellt genetisches Programmieren eine dritte Kategorie dar.

Aber zurück zur Künstlichen Intelligenz. Alan Turing dachte nicht nur in seinem berühmten Artikel „Computing Machinery and Intelligence“ [TUR50] darüber nach, ob eine Maschine denken kann, sondern ihn beschäftigte vor allem die Frage „wie können wir feststellen, dass sie es tut?“ und ob man in der Zukunft noch synthetische von menschlicher Intelligenz unterscheiden könne. Mit seinen Überlegungen legte er den Grundstein für die Künstliche Intelligenz, die als Begriff 2 Jahre nach seinem Tod geprägt wurde. Turing selbst meinte: „My contention is that machines can be constructed which will simulate the behaviour of the human mind very closely. They will make mistakes at times, and at times they may make new

and very interesting statements, and in the whole the output of them will be worth attention to the same sort of the extend as the output of a human mind.“ [TUR51].

Aus dem Bereich der Künstlichen Intelligenz (KI), genauer aus dem Bereich der „Verteilten Künstlichen Intelligenz (Distributed Artificial Intelligence DAI)“ entstammen ursprünglich Agenten. Dieser Bereich der Künstlichen Intelligenz umfasst wiederum drei Teilbereiche: Multi-Agenten-Systeme (Multi-Agent-Systems, MAS), Verteiltes Problemlösen (Distributed Problem Solving) und Parallele Künstliche Intelligenz (Parallel Artificial Intelligence). Agenten entstammen, wie es der Name bereits ausdrückt, dem Bereich der Multi-Agenten-Systeme. Obwohl Agency in der Forschung der Künstlichen Intelligenz als zentral angesehen werden kann, gab es innerhalb der KI-Gemeinschaft bis in die 80er Jahre vergleichsweise wenige Bemühungen in Richtung der intelligenten Agenten [JEN98].

Alan Kay, ein großer Befürworter der Agententechnologie, führt den Ursprung der Agententechnologie auf eine Idee von John McCarthy aus der Mitte der (19)50iger Jahre zurück [KAY84]. Einige Jahre später wurde der Begriff „Agent“ von Oliver G. Selfridge geprägt. Beide Forscher, McCarthy und Selfridge, arbeiteten am Massachusetts Institute of Technologie (MIT), wo sie ein System andachten, welches zu einem gegebenen Ziel die entsprechenden Schritte ermitteln und dabei vom Benutzer Hinweise erfragen kann – ein „Soft Robot“. Dieser agiert im Gegensatz zum Roboter nicht in der „realen“ Welt, sondern im Computer.

Carl Hewitt´s „Actor“ Modell (1977) [HEW77] beschreibt „a self-contained, interactive and concurrently-executing object, with some encapsulated internal state and which could respond to messages from other similar objects”. Zu dieser Zeit (Ende der 1970iger, Beginn der 1980iger Jahre) beschäftigte sich die Forschung mit deliberativen bzw. smarten Agenten [END]. Später gingen die Arbeiten in Richtung der reaktiven Agenten. Diese Agenten arbeiten mit symbolischen, internen Modellen. Dabei konzentrierte man sich auch vorwiegend auf die Zusammenarbeit und Kommunikation zwischen einzelnen Agenten, auf die Aufgabenverteilung, auf die Interaktion, Koordination und Konfliktlösung durch Argumentation. Es besteht also eine Konzentration auf das Ganze – die Makro-Aspekte – der „Gesellschaft der Agenten“.

Seit etwa 1990 hat sich eine weitere Forschungsrichtung entwickelt, die sich stärker auf die einzelnen Agenten – Agententypen, Architekturen, Agententheorien und Sprachen –

konzentriert. Auch werden breitere Anwendungsbereiche (Web, etc.) betrachtet und es wird verstärkt im Bereich der mobilen Agenten und der Agenten als „intelligente“ Assistenten geforscht.

Seit 1993 führte die rasante Verbreitung des Internets und des World-Wide-Web (WWW) als globales, für jeden verfügbares Informationssystem zum Entstehen eines neuen Agententyps – dem Internetagenten. Dieser findet seine konkrete Ausprägung als Suchagenten, Informationsagenten, Shoppingagenten, etc.

In Zukunft werden Agenten nicht nur aus unserer Arbeitswelt, sondern auch aus unserer Gesellschaft nicht mehr wegzudenken sein. Besonders durch die mobilen Agenten und durch die Verbreitung des Internet, ist auch die Forschung in diesen Bereichen intensiver geworden und auch immer mehr Unternehmen entwickeln Systeme, die mit Agenten arbeiten. So wird zur Zeit an der Stanford University in Kalifornien die Kommunikation von Agenten untersucht und es werden bereits Agentenfunktionen in einer Java-Anwendung oder einem Applet bereitgestellt [CDR]. Die Artificial Intelligence Research Group am Worcester Polytechnic Institute in England arbeitet verstärkt an lernfähigen Maschinen, wissensbasierten Systemen und Multi-Agenten-Systemen. Am Massachusetts Institute of Technologie (MIT) wird daran gearbeitet die Agententechnologie ins Fernsehen bzw. ins Internet zu integrieren. Dies betrifft vor allem elektronische Marktplätze und Agenten auf mobilen Endgeräten. Dabei werden aber auch soziale Aspekte untersucht. [SOF].

Last, but not least arbeiten Organisationen, wie die Foundation for Intelligent Physical Agents [FIP] oder die Object Management Group (OMG) [OMGa] daran, Standards im Bereich der Agententechnologie zu schaffen.

Die englische Beratungsfirma OVUM schrieb bereits 1997 in einer Studie [GUI97]: “Agententechnologie verspricht, das Web zu einem effektiveren Medium für E-Commerce zu machen. Persönliche Agenten werden Käufern helfen, die Produktauswahl, sowie erhaltene Inhalte und Dienste an ihre Bedürfnisse anzupassen. „Data Mining“-Agenten werden Verkäufern helfen, ihre Kunden zielgenauer anzusprechen.“ und [NOR94] meint “Smart agents are here to stay. Once unleashed technologies do not disappear”

2.4. Aktuelle Forschungsbereiche und Anwendungsgebiete

2.4.1. Agententheorien

Aufgabe einer Agententheorie ist es alle Teilaspekte eines Agenten in ein logisches System zusammen zu fassen. Dieses Thema ist sehr komplex, da Agenten ein sehr vielschichtiges Forschungsgebiet darstellen. Dabei geht es um das Bewusstsein, Ziele und Wünsche von Agenten, den damit verbundenen Modellen, Konfliktanalysen und Kommunikation, Sicherheitsaspekten, etc. Momentan zeigt sich, dass sich die einzelnen Forscher und Forschergruppen bei ihren Agententheorien jeweils auf unterschiedliche Schwerpunkte konzentrieren. Dies und die Vielschichtigkeit und Komplexität des Themas mag der Grund sein, dass es bis heute noch keine allumfassende Agententheorie gibt. Im folgenden sollen einige Agententheorien kurz geschildert werden. Es sei allerdings erwähnt, dass gerade im Bereich der Agenten verstärkt geforscht wird und deshalb dienen diese vier Beispiele als Repräsentanten ohne dabei einen Anspruch auf Vollständigkeit erheben zu wollen.

Robert C. Moore: Moore beschäftigt sich in seiner Forschung mit der Frage, was Agenten wissen müssen, um handeln zu können. Dafür stellt er „Vorbedingung für Handlungen (preconditions for actions), ein Modell der Fähigkeit und ein dynamisches System zur Modellierung von Aktionen auf. Dies erlaubt es den Agenten mit nur wenig Information durch bestimmte Aktionen mehr über ihre Ziele herauszufinden. Die Agenten können also mit unvollständigen Zielen umgehen [MOO90].

Paul R. Cohen and Hector J. Levesque: haben eine sehr bekannte Agententheorie entwickelt. Dabei haben sie ein Modell entwickelt, dass sich auf Glaube und Ziele stützt und wobei andere Eigenschaften, wie etwa Absichten über Glaube und Ziele definiert werden. Das Hauptaugenmerk liegt dabei vorwiegend auf Multiagentensystemen und der Konfliktlösung und Kooperation in Multiagentensystemen ([COH90a], [COH90b]).

Anand S. Rao and Michael P. Georgeff: Das Modell von Rao und Georgeff basiert auf einem System, das nach der Zeit strukturiert ist. Ihr System weist im Gegensatz zu Cohen und Levesque nicht nur zwei Eigenschaften als Hauptmerkmale auf, sondern ihr Modell stützt sich auf drei wesentliche geistige Eigenschaften, wie Glaube, Ziele und Absichten (Belief, Desire, Intention – BDI-Agents). Dabei steht Belief für den Glauben, das Gedächtnis, Desire für die

Ziele, die Motivation des Systems und Intention für die aktuellen Absichten des Systems, also die Änderungen der Ziele, die aus der Veränderung der Umgebung resultieren [RAO95].

Michael J. Wooldridge: Wooldridge konzentrierte sich auf die Entwicklung einer Familie von logischen Systemen, welche die Eigenschaften von Multiagentensystemen repräsentieren. Ziel dabei war es einige Formalismen für die Spezifikation und Verifikation realistischer Multiagentensysteme zu schaffen ([WOO92a], [WOO92b]).

2.4.2. Anwendungsgebiete

Der Einsatz von Agenten führt zu einem Paradigmenwechsel von der direkten Manipulation hin zur indirekten Manipulation.

Bei der derzeit hauptsächlich vorherrschenden direkten Manipulation werden Aktionen explizit vom Benutzer gestartet und alle entstehenden Ergebnisse werden ebenfalls durch den Benutzer selbst betrachtet und ausgewertet. Der Einsatz von Agenten führt zu einer indirekten Manipulation [MAE94]. Sprich, es findet ein kooperativer Prozess zwischen Benutzer und Agent statt. Beide können Aktionen starten, Ergebnisse betrachten und auswerten, Aufgaben ausführen und kommunizieren. Durch den Einsatz von Agenten besteht somit beispielsweise die Möglichkeit ungeübten Benutzern einen effektiven Zugriff auf das Internet zu erleichtern und Geübten eine schnellere Methode zur Informationenbeschaffung zur Verfügung zu stellen. Dabei ist der Agent als Partner zu verstehen, der mit einer gewissen Eigenständigkeit Aktionen ausführen und Aufgaben bewältigen kann.

2.4.2.1. System- und Netzwerkmanagement

Aufgrund der immer größer werdenden Komplexität des Bereichs des System- und Netzwerkmanagements tauchten hier Agenten bereits zu einem frühen Zeitpunkt in der Forschung auf und es daher auch weitreichende Expertise und Anwendungsfälle. Im System- und Netzwerkmanagement werden Agenten eingesetzt, um die ständige Verfügbarkeit der Systeme sicher zu stellen. Agentenfunktionen reichen dabei von der Überwachung ein oder mehrerer Elemente der Netzwerke, über das Erkennen und Aufspüren von Fehlersituationen bis hin zum selbständigen Beheben von Fehlern oder Störungen. Nötigenfalls, wenn eine Reparatur nicht möglich ist, sollen diese Agenten übergeordnete Instanzen benachrichtigen.

Die Agenten sollten dabei in der Lage sein aus früheren Fehlern zu lernen und künftige ähnliche Situationen behandeln zu können [GUE96].

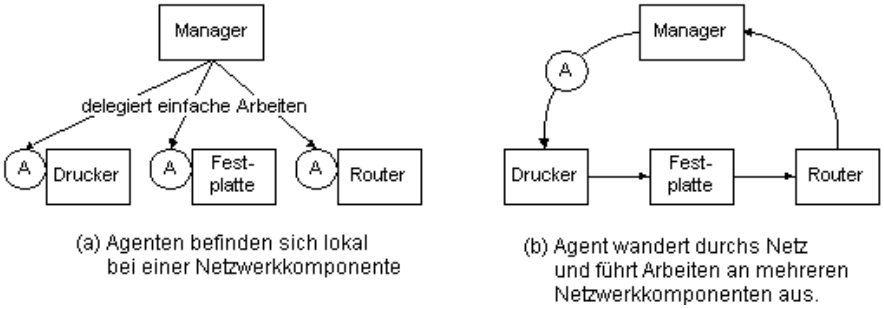


Abbildung 7: Einsatz von Agenten beim Netzwerkmanagement [MAG96]

Agenten finden jedoch nicht nur in Netzwerken allgemein Anwendung, sondern besonders in Telekommunikationsnetzwerken ([FRI01]). Dabei sind sie eine notwendige Unterstützung im Management komplexer Telekommunikationsdienste, die durch dynamische Netzwerk-Rekonfigurationen charakterisiert sind. Die physische Größe dieser Netzwerke und die strengen Regeln für deren Operationen können durch mobile Agenten, die miteinander als Repräsentanzen der einzelnen Netzwerk-Ressourcen kommunizieren, flexibel und trotzdem effektiv verwaltet werden.

2.4.2.2. Intelligente Benutzerschnittstellen

Schon einige Zeit werden Agenten in Betriebssystemen und im Bereich von Anwendungsprogrammen eingesetzt. Durch die Vielfalt von Anwendungen und graphischen Benutzerschnittstellen (Graphical User Interface, GUIs) eröffnet sich ein weiterer Anwendungsbereich für Agenten. Diese Interface-Agenten sollen das Verhalten der einzelnen Benutzer beobachten und automatisch Hilfe anbieten, wenn Probleme auftreten. Manche Agenten arbeiten dabei im Hintergrund ohne direkte Benutzerintervention. Sie beobachten das Benutzerverhalten und ihre Hauptaufgabe liegt in der Fehleranalyse bzw. in der Anleitung zu einer effizienteren Benutzung (z.B. Office-Assistent von Microsoft). Es werden aber auch Agenten eingesetzt, die durch Beobachtung der Benutzeraktionen lernen, wie der Benutzer vorgeht und dabei dem Benutzer anbietet immer wiederkehrende Tätigkeiten für ihn zu übernehmen (z.B. Open Sesame! von Apple [CAG97]).

2.4.2.3. Informationsmanagement

Ein weiterer Bereich in dem Agenten schon länger ein Thema sind, ist der Bereich des Informationsmanagements. Dies umfasst Bereiche, wie Informationssuche (Beschaffung von Expertise), Informationsfilterung, -aufbereitung und –speicherung.

Dabei geht es primär darum den Benutzern Arbeit abzunehmen und ihnen Zeit zu sparen. Gerade durch das Internet nehmen mögliche Informationsquellen täglich zu und die Datenformate werden immer vielfältiger (Texte, Audio, Video, etc.). Die aufwendige Informationssuche im Internet soll durch den Einsatz von Agenten für den Benutzer entfallen und seine Aufgabe wird auf die Auftragserteilung und Abfrage von Resultaten verkürzt. Vor allem haben die Agenten aber auch die Aufgabe auf die Präferenzen (Interessen, Vorlieben, Gewohnheiten, etc.) der Benutzer Rücksicht zu nehmen und eventuell durch Kooperation mit anderen Agenten ([HER01]) zu besseren Ergebnissen zu kommen.

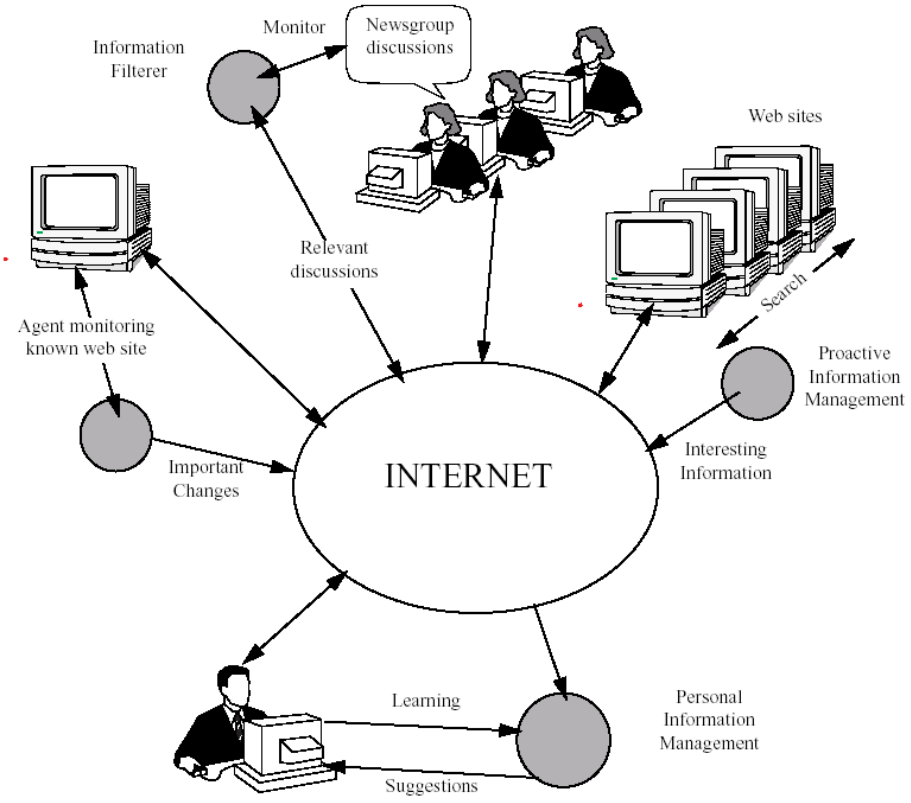


Abbildung 8: Informationsmanagement [JEN96]

2.4.2.4. "Workflow Management"/Arbeitsgruppen

Im Bereich des Workflow-Managements können Agenten sehr effektiv eingesetzt werden, da besonders in Workflow-Systemen viele Arbeitsprozesse koordiniert werden müssen und oft automatisierte Tätigkeiten vorliegen. Chang und Scott [CHA96] beschreiben ein System, wobei mobile Agenten den Transport von Dokumenten übernehmen, Überprüfen, ob Informationen fehlen und diese gegebenenfalls auch von anderen Stellen beschaffen. Der Sachbearbeiter soll dabei die vollständigen Informationen vom Agenten erhalten. Bei dringlichen Aufgaben kann der Agent feststellen, wer die Vorgängeraktivität bearbeitet. Ist diese Aktivität noch nicht bearbeitet, kann der Agent anfragen, wer diese Aufgabe übernehmen kann und die nötigen Informationen dafür weiterleiten.

In Arbeitsgruppen können Agenten ebenfalls zahlreiche Aufgaben für ihre Benutzer übernehmen. Sie können die Kalender pflegen und gemeinsame Termine abstimmen. Sie sind imstande die Verteilung von Unterlagen durchzuführen, Benachrichtigungen vorzunehmen und etwa Dienstreisen (Unterkünfte, Verkehrsmittel, etc.) zu organisieren und koordinieren.

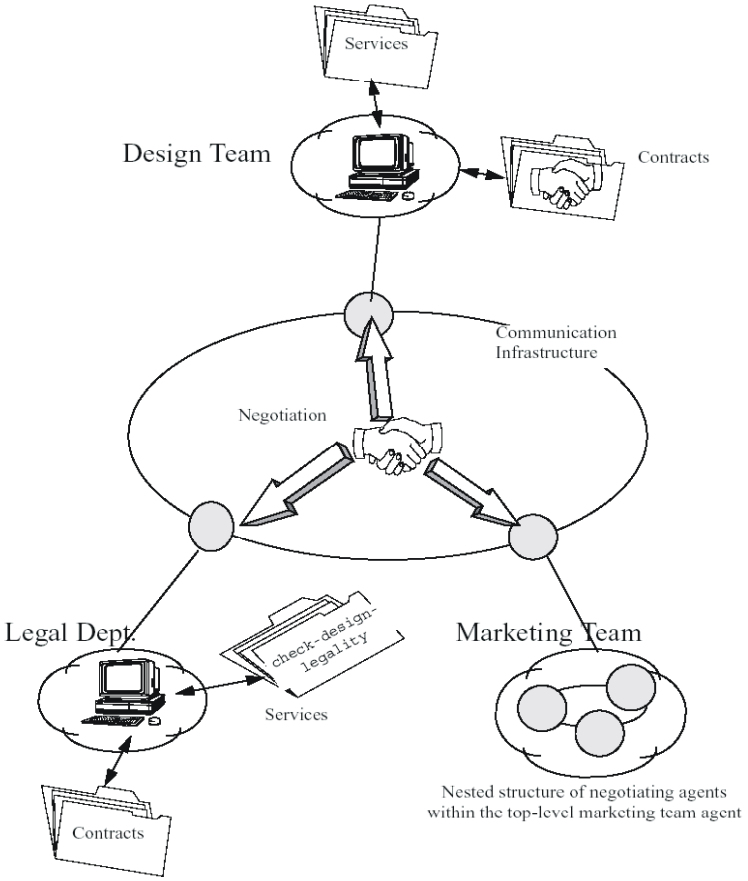


Abbildung 9: Business Process Management [JEN96]

2.4.2.5. Electronic Commerce

Agenten können in allen Geschäftsbereichen unterstützend wirken. Dabei können sie nicht nur Produkte und Anbieter für die Konsumenten finden, sondern auch verhandeln und Käufe abschließen. Auch die Anbieter können etwa durch Verkaufsagenten unterstützt werden. Im E-Commerce kommen bereits heute unterschiedlichste Agenten ([MAA01], [WUR01]) zum Einsatz. Die Bandbreite der Agenten reicht dabei von einfachen Kaufagenten, die Preisvergleiche und Produktübersichten liefern (z.B. BargainBot [BOT]), über komplexere Kaufagenten, die bereits Kaufvorgänge unterstützen (z.B. Jango) bis hin zu agentenorientierten Marktplätzen (z.B. Kasbah, Tete-a-Tete), die Repräsentations-möglichkeiten für unterschiedliche Anbieter offerieren und die auch Verhandlungen ermöglichen (siehe auch Abbildung 10).

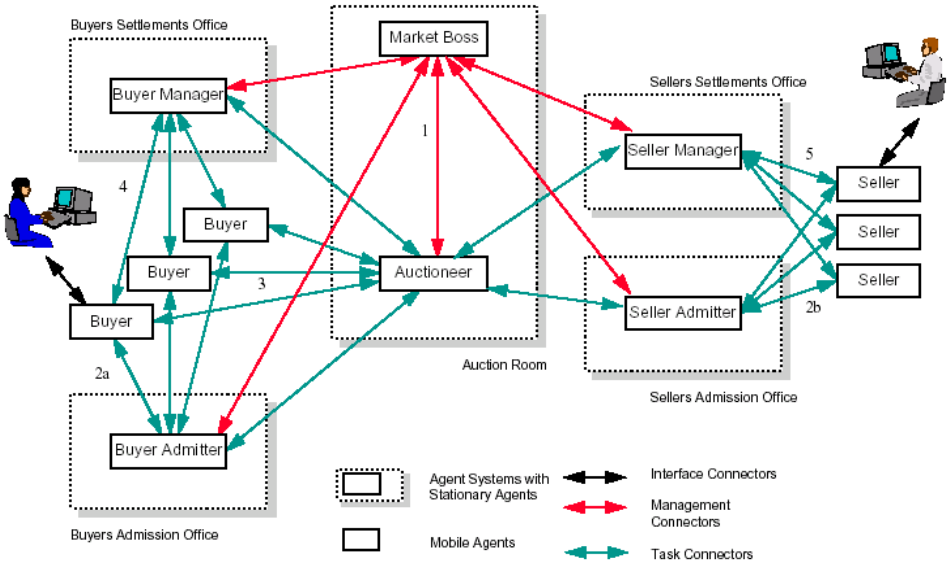


Abbildung 10: Beispiel eines elektronischen Marktplatzes [HOR]

2.4.2.6. Mobile Computing - Persönliche Digitale Assistenten

Persönliche Digitale Assistenten (PDA) finden immer größere Verbreitung. Zur Zeit sind dies vorwiegend Geräte, wie Palm-Organizer oder PocketPC-Geräte, die klein und handlich sind und als neue Terminkalender, Adress-, Aufgaben- und Emailverwaltung dienen. Darüber hinaus bieten sie aber auch die Möglichkeit der Nutzung des Internet und des Einsatzes von Multimedia (Radio, Video, etc.). Die Fähigkeit von mobilen Agenten ([LIN95]), auf entfernten Rechnern Operationen auszuführen, eröffnet dem Benutzer die Möglichkeit, den

mobilen Agenten als „Persönlichen Assistenten“ Aufgaben im Namen des Benutzers durchführen zu lassen.

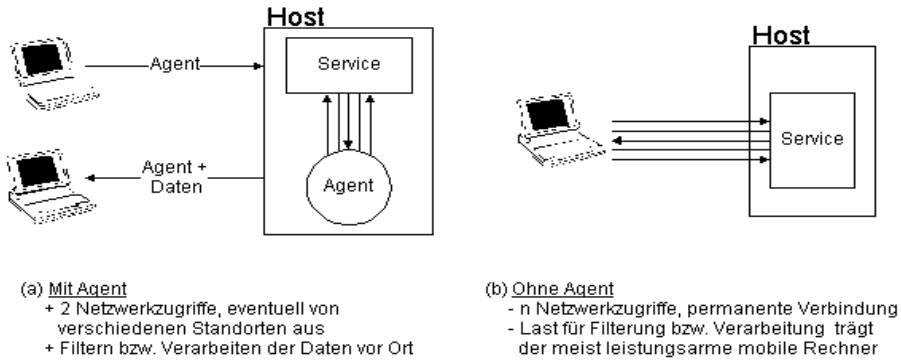


Abbildung 11: Einsatz von Agenten beim Mobile Computing [ROT97]

Der entfernte „Assistent“ kann unabhängig von einer begrenzten Verfügbarkeit des Netzwerkes arbeiten, und der Benutzer braucht in dieser Zeit nicht „online“ zu sein. Die Aufgaben dieser Assistenten sind vielzählig und umfassen z.B. Recommendation Agents (Bücher, CDs, Filme), Mail-Filtering ([MAE97]), Terminkalenderverwaltung, etc. Diese Option wird in naher Zukunft umso wichtiger werden, da diese PDA´s mit dem Handy verschmelzen und dabei besonders die Möglichkeit des Zugriffs auf externe Ressourcen, wie Speicher, Rechenkapazität, etc. eine wichtige Rolle spielt.

Zur besseren Übersicht werden in der folgenden Tabelle die erwähnten Einsatzgebiete und die möglichen Aufgaben der Agenten gegenübergestellt:

EINSATZGEBIETE	AUFGABEN						
	Überwachung	Protokollierung	Analyse	Ressourcen-verwaltung	Individualisierung	Informations-filterung	Koordination
System- und Netzwerk-management	X	X	X	X			
Intelligente Benutzerschnittstellen					X		
Informationsmanagement			X	X	X	X	X
“Workflow Management”/		X	X	X	X		X

Arbeitsgruppen							
Electronic Commerce	X	X	X		X	X	X
Mobile Computing – PDA´s				X	X	X	

Tabelle 1: Aufgaben von Agenten in den einzelnen Einsatzgebieten

2.4.2.7. Weitere Einsatzgebiete

Die Liste der Einsatzgebiete lässt sich noch endlos fortsetzen. Hier sollen stichwortartig ([ALB01], [LIE01b], [PAP01]) noch einige wiedergegeben werden:

- Industrielle Anwendungen, wie Prozesssteuerung (Agenten fungieren als Controller), Fertigung (Agenten stellen Fertigungszellen dar), Flugleitsysteme (Agenten werden in Flugkontrollsystemen und Instrumentenüberwachung eingesetzt).
- Medizinische Anwendungen, wie Patientenüberwachung und Patientenmanagement.
- Unterhaltung, wie der Einsatz in interaktiven Filmen oder Spielen bzw. innerhalb der Forschung in der Simulation komplexer Systeme.
- Spezialisierte Agenten, die einander Helfen und miteinander in Kombination einen Mehrnutzen ergeben, wie etwa im Büro der Zukunft:

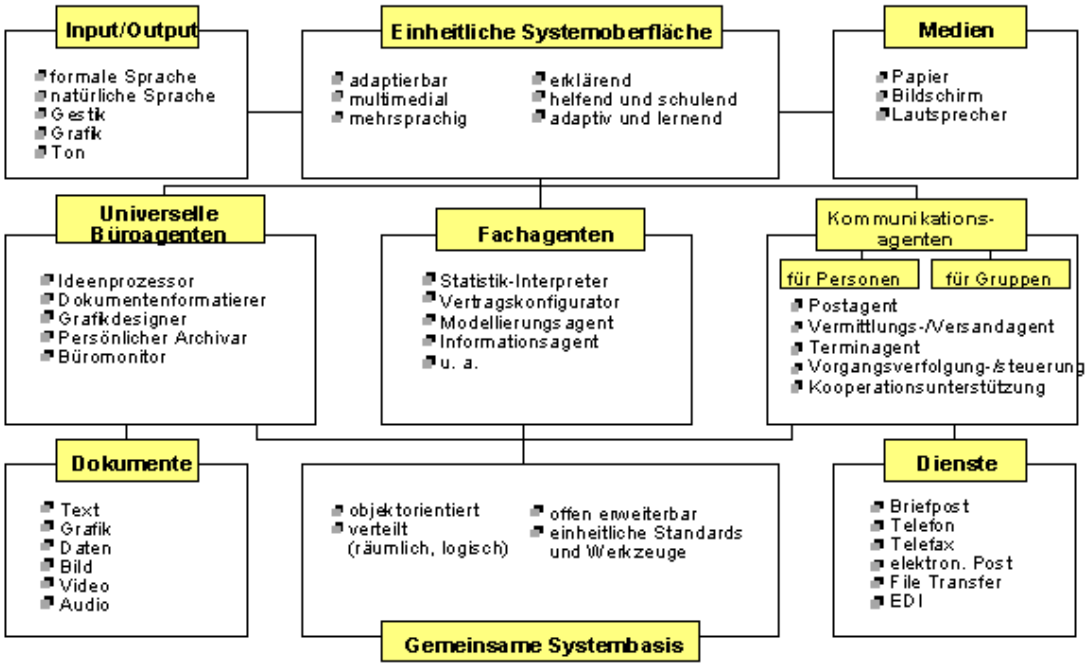


Abbildung 12: Büro der Zukunft [BOD]

2.5. Agenten: Möglicher Nutzen, Vorteile, Probleme und ungelöste Fragen

Agenten können den Benutzern einige Vorteile bzw. Nutzen bieten. Dies reicht je nach Anwendungsgebiet von einfacher Benutzerunterstützung durch Hinweise, die der Agent dem Benutzer gibt, bis hin zur vollständigen Übernahme und Durchführung von Aufgaben.

Dennoch darf man nicht verschweigen, dass es auch im Bereich der Agenten noch ungelöste Fragen und Probleme gibt. Das heißt selbstverständlich nicht, dass jeder Benutzer und Agent mit allen Problemen konfrontiert sein muss. Wichtig ist es jedoch im Umgang mit Agenten ein Problembewusstsein zu haben und zu wissen, welche Schwierigkeiten einem begegnen können, und wie man darauf reagieren kann.

Im folgenden sollen daher einige Aspekte für den Nutzen von Agenten, aber auch mögliche ungelöste Fragen angesprochen werden.

2.5.1. Mögliche Vorteile und Nutzen

Da Agenten in der Regel autonom sind, sind sie in der Lage selbständig unterschiedliche Aufgaben für ihre Benutzer auszuführen. Dabei setzen sie Aktionen, die sie eigenständig ausführen und auch kontrollieren. Agenten kommen dabei mit asynchroner Kommunikation aus, außer eine Verbindung mit dem Agenten wird vom Benutzer ausdrücklich gewünscht.

Je nach dem Grad ihrer Intelligenz und Kollaborationsfähigkeit können sie bei der Ausführung ihrer Aufgabe in dynamischen Umgebungen arbeiten. Dies ist bedingt durch die Tatsache, dass sie ihre Umgebung wahrnehmen und auf Umweltänderungen reagieren können. Beispielsweise können sie auch basierend auf ihrem Wissen Benutzeranfragen ergänzen oder modifizieren und somit bessere Ergebnisse für den Benutzer erzielen. Intelligente Agenten unterstützen somit die Produktivität ihrer Benutzer.

Mobile Agenten haben die Fähigkeit ihren Ausführungsort zu wechseln, d.h. sie sind nicht an einen Rechner und dessen Ressourcen gebunden, sondern können alle Ressourcen innerhalb eines Netzwerkes nutzen. Dabei können sie auch in der Lage sein, den Umgang mit unterschiedlichen Umgebungen zu lernen und sich automatisch an Änderungen ihre Umwelt zu adaptieren. Durch die Mobilität wird der Agent in seiner Ausführung flexibler. Die

Komplexität wird reduziert, aber auch die Zuverlässigkeit steigt, dadurch dass die Agenten auf andere Plattformen und Ressourcen ausweichen können.

Mobile Agenten bieten eine wichtige Unterstützung von „Mobile Computing“. Anwendungen auf der Basis von mobilen Agenten können die Restriktionen von Mobile Computing (schmale Bandbreiten, geringe Rechen- und Speicherleistung der mobilen Geräte, begrenzte Benutzerschnittstelle, begrenzte Netzpräsenz der Geräte) umgehen und externe Ressourcen nutzen.

Andere Ressourcen können aber auch andere, verschiedene, heterogene Datenquellen sein. Die Daten müssen nicht an einer einzigen Stelle zugänglich sein, sondern können verteilt sein. Somit können auch die Aufgaben dezentralisiert, die Funktionalität auf mehrere Agenten verteilt und die einzelnen Ergebnisse durch Kooperation wieder zusammengeführt werden.

Folgedessen ergibt sich auch eine Reduzierung der Kommunikationskosten. Durch Migration kann ein Lastausgleich zwischen den Komponenten hergestellt werden. Auch die Anforderungen an die Clients werden durch eine stärkere Nutzung der Server-Ressourcen reduziert.

Die Möglichkeit der asynchronen Ausführung bietet den Vorteil, dass der Benutzer nicht ständig mit dem Netzwerk verbunden sein muss. Der Agent wird zum Server transferiert, der Client kann sich vom Netz trennen und nach erfolgter Abarbeitung, kehrt der Agent zum Client zurück.

Agenten sind somit autonome Einheiten, die eventuell miteinander kooperieren und auch die Möglichkeit zur Migration haben. Sie dienen ihren Benutzern als persönliche digitale Assistenten, die ihnen helfen, Aufgaben abnehmen und zusätzlich die Netzlast, die Ressourcenauslastung und die Kommunikationskosten verringern können.

2.5.2. Mögliche Probleme und ungelöste Fragen

Obwohl Agenten nutzbringend sein können, so gibt es natürlich auch Grenzen für Agenten. Agenten werden entworfen und implementiert von Menschen. Dabei kann es zu Fehlern kommen. Fehler können sich dabei nicht nur in den Regeln und in der Logik einschleichen,

sondern auch in der Wissensdatenbank der Agenten. Demgemäss kann es dazu kommen, dass Agenten falsche Entscheidungen treffen oder dem Benutzer schlecht beraten.

Menschen setzen oft auch übertriebene Erwartungen in die Fähigkeiten von Agenten. Sie sehen bei den Agenten menschliche Eigenschaften und menschliches Verhalten und neigen daher dazu die Fähigkeiten der Agenten zu überschätzen.

Manche „mobile“ Agenten sind in ihrer Migration auf bestimmten Plattformen eingeschränkt oder auf bestimmte Schnittstellen angewiesen. Agenten mit derartigen Merkmalen liefern somit nur Ergebnisse aus diesem eingeschränkten Bereich, ohne dass dies dem Benutzer bewusst wird. Wenn der Benutzer nach dem besten Preis eines Produkts sucht, bekommt es folglich möglicherweise nicht den besten Preis aller im Internet präsenten Händler, sondern nur jener die eine spezielle Funktionalität oder Schnittstelle zur Verfügung stellen.

Besonders heikel ist der Bereich der Privatsphäre. Vor allem bei mobilen Agenten ergibt sich das Problem der Wahrung der Privatsphäre. Agenten haben Zugriff oder migrieren mitsamt relevanten Benutzerinformationen. Solche Informationen können persönliche Daten, wie Adresse, Familienstand, Glaubenbekenntnis, Passwörter, Kreditkartennummern, Daten über finanzielle Aktivitäten, Korrespondenz und vieles mehr sein. Bei diesen „wissenden“ Agenten bedarf es daher besonderer Schutzmaßnahmen für die Daten, da diese zusätzliche Angriffspunke für andere Systeme oder Agenten darstellen, um diese Informationen unrechtmäßig zu erlangen. Es besteht aber auch die Möglichkeit, dass Systeme, auf denen mobile Agenten laufen, so manipuliert sind, dass das Gedächtnis des Agenten vollständig ausgelesen wird und das Wissen der Agenten manipuliert werden kann, sodass der Agent nicht mehr im Sinne seines Benutzers arbeitet. Beispielsweise könnte ein Agent „überzeugt“ werden, elektronisches Geld des Benutzers an einen anderen zu überweisen.

Damit verbunden ist auch eine Reihe von rechtlichen Problemen. Wenn ein Agent einmal eine Fehlfunktion aufweist und daraus z.B. finanzielle Verluste entstehen, wer ist dann dafür verantwortlich zu machen? Trägt der Benutzer die Haftung, weil er die Aufgabe vielleicht zu undeutlich beschrieben hat, ist der Hersteller verantwortlich, weil das Programm eventuell eine Fehlfunktion aufweist oder ist der Betreiber des Systems zur Rechenschaft zu ziehen, weil das System nicht sicher genug ist? Sind Vertragsabschlüsse zwischen Agenten bindend?

Das Delegieren von Aufgaben an Agenten erfordert Vertrauen, auch Vertrauen in selbständige Entscheidungen. Personen möchten das Gefühl haben ihr Computersystem, also auch ihre Agenten zu kontrollieren. Wieviel Autorität und Verantwortung sind wir daher bereit (oder müssen wir bereit sein) intelligenten Agenten zu übertragen?

Darüber hinaus sind viele andere Probleme zu lösen: Wie wird z.B. der Ressourcenverbrauch auf fremden Systemen geregelt, müssen Agenten „ehrlich“ gegenüber anderen sein. Wer kontrolliert und reguliert Agenten? Welche Bedrohungen können noch von ihnen ausgehen? - Agenten sollten nicht destruktiv die Welt ändern. Wie kann man das Problem des Zauberlehrlings: „Die ich rief, die Geister, Werd ich nun nicht los?“ [GOE] vermeiden.

Eichmann fordert in [EIC94]: Agenten müssen sich identifizieren, Agenten müssen die Geschwindigkeit ihrer „Requests“ anpassen, Agenten müssen Information mit Agenten teilen, Agenten müssen die Autorität von Systemoperatoren respektieren.

3. AGENTEN-PORTALE/PLATTFORMEN

3.1. Definition eines Portals/einer Plattform

Im Bezug auf das Internet ist oft von Agentenportalen oder Agentenplattformen die Rede. Diese beiden Begriffe werden oftmals synonym und ohne klare Definition verwendet. Deshalb wird im folgenden eine kurze Begriffsdefinition gegeben um die Unterschiede dieser Begriffe herauszustreichen.

Der Begriff des **Portal**s nach [MTC01]: „Ein Portal ist eine stark frequentierte Site im Internet, die meist von einer der großen High-Tech-Firmen betrieben wird. Portal deswegen, weil die Site zum einen für viele Surfer als Eintrittspunkt in das Internet dient, zum anderen einen gebündelten Ausgangspunkt zu anderen Sites bietet. Da ein Portal sozusagen einen Knotenpunkt im Internet darstellt, finden hier zahlreiche Firmen potentielle Zielgruppen und Kunden. Eine der Haupteinnahmequellen eines Portals ist aus diesem Grund der Verkauf von Werbefläche auf der Site."

Der Begriff der **Plattform** nach [MTC01]: „Als Rechnerplattform bezeichnet man die Arbeitsumgebung, in der sich der Anwender bewegt, also das Betriebssystem und die darauf ablaufenden Anwendungsprogramme."

Ein Portal ist daher eine Einstiegsseite, auf der der Benutzer wichtige Informationen findet und die oftmals individuell gestaltbar oder personalisiert (auf die Bedürfnisse und Wünsche des Benutzers abgestimmt) ist.

Unter einer Plattform hingegen ist der Bereich zu Verstehen auf dem das Portal aufgebaut ist, d.h. die entsprechende Hardware, das Betriebssystem, die Dienste und somit auch allfällig ein Agentensystem, das auf einer Plattform angeboten wird.

3.2. Portale

3.2.1. Zweck

Portale haben eine wichtige Funktion. Sie stellen eine Einstiegsplattform dar, helfen dem Benutzer beim Auffinden von wichtigen Informationen. Dabei wird oftmals eine

Personalisierung angeboten, bei der die Interessen des Benutzers erfasst werden und versucht wird nur für den Benutzer relevante Informationen darzubieten. Dabei bündeln sie ansonsten autonome Informationen und ergänzen diese um teilweise selbst erstellte Dienste, wie Kommunikationsdienste (Email, Chat, etc.), etc. ([POR]). Ein Portal sollte dem Benutzer das Gefühl geben wohlorganisiert zu sein, sprich für den Benutzer soll ein hoher Nutzwert entstehen durch ein vielfältiges Angebot an Diensten und einer hohen Aktualität der Informationen. Beziehungsweise sollte sich der Benutzer als Teil einer Gemeinschaft (community) fühlen können und das Gefühl haben, dass der Fokus auf ihn und seine Bedürfnisse ausgerichtet ist.

In Zukunft wird die Kombination von Portalen und intelligenten Agenten eine wichtige Rolle spielen. Mehr dazu in Kapitel „Beziehung Portal und Agentenplattform".

Portal ist aber nicht gleich Portal. Portale unterscheiden sich nicht nur dadurch, dass es sich bei manchen um Internetportale und bei anderen um Intranetportalen handelt, sondern vielmehr durch ihre Zielgruppe.

3.2.2. Unterschiedliche Ausprägungen- Taxonomie von Portaltypen

3.2.2.1. Horizontale Portale – Mega-Portale

Unter horizontalen Portalen werden Portale verstanden, die die Allgemeinheit ansprechen. Sie sind meist der erste Anlaufpunkt für die Mehrzahl der Internetbenutzer und zeichnen sich daher auch durch hohe Besucherzahlen aus. Beispiele dafür sind etwa Yahoo.com, Excite.com, Lycos.com, etc. Diese Portale bieten derzeit größtenteils kostenlose Dienste an, da sich die Portale hauptsächlich über Werbung finanzieren. Um die „Kundenbindung" zu verstärken und somit ihre hohen Besucherzahlen halten zu können, setzen diese Portale sehr stark auf Personalisierung (z.B. My Yahoo, My Excite, etc.) [FLE01]. Damit geben sie ihren Benutzern Unterstützung bei ihren individuellen Interessen. Die Personalisierung sieht dabei oftmals auch vor, dass nicht nur die favorisierten Seiten des Benutzers auf der Einstiegsseite präsentiert werden, sondern dass die Benutzer die eigene Einstiegsseite individuell in Aussehen und Diensten gestalten, ihre Termine oder ihre Korrespondenz über das Portal verwalten können, etc.

Typische Bestandteile eines horizontalen Portals sind somit:

- Aktuelle allgemeine Nachrichten
- Aktuelle spezielle Nachrichten, wie Sport, Wetter, TV, Kino, etc.
- Shoppingmöglichkeiten bzw. Auktionen
- Suchmaschinen, Linksammlungen
- Chat/Foren
- Registrierung - Kostenlose Accounts mit E-Mail, Kalender-Service, etc.
- Personalisierung
- Werbung

3.2.2.2. Vertikale Portale - Vortale

Vertikale Portale oder auch Vortale [WEB] genannt, sind nicht auf die Allgemeinheit ausgerichtet, wie horizontale Portale sind durch einen Fokus auf bestimmte Benutzergruppen bestimmt. Vertreter dieser Portale sind etwa Unternehmensportale, E-Commerce Portale, Nischenportale etc. Sie alle sind auf bestimmte Benutzergruppen mit spezifischen Interessen ausgerichtet.

Unternehmensportale (Enterprise Information Portal - EIP, Enterprise Resource Portals – ERP, Business to Employee – B2E, Corporate Portals)

Gerade bei den Unternehmensportalen handelt es sich oftmals um hybride Portale. Diese stellen eine Verzahnung aus traditionellen Webportalen und den Intranetportalen dar. Dabei wird das Unternehmenswissen verbunden mit der Präsentation des Unternehmens nach außen. Die Vorteile, die sich daraus ergeben, spiegeln sich vor allem in einer Nutzungsmöglichkeit von bisher ungenutzten Daten wider. Nunmehr ist der Zugriff auf alle Unternehmens-informationen möglich. Ziel ist es den Mitarbeitern Informationen und Dienste online zur Verfügung zu stellen, damit sie effizienter und produktiver arbeiten können. Durch die Nutzung des Portals haben die Mitarbeiter Zugang auf eine Reihe von Diensten,

Informationen, Neuigkeiten, etc. Um etwa Datensicherheit zu gewährleisten, kann der Zugang mittels Zugriffsrechten individuell gestaltet werden kann.

Auch Kunden, Lieferanten und Partner profitieren von diesen zusätzlichen Unternehmens-informationen. Einerseits können direkt Daten mit Lieferanten und Partnern ausgetauscht und infolgedessen auch die Abläufe von Geschäftsprozessen beschleunigt werden, andererseits besteht auch die Möglichkeit die Kunden direkter als bisher anzusprechen. Dabei wird oftmals auch Online-Verkauf genutzt und durch die Anbindung der Unternehmensdaten können die Kunden beispielsweise bereits zum Bestellzeitpunkt über den Lagerstand und über die Lieferzeit informiert werden. Es gibt daher meist keine klare Trennlinie zu Business-to-Business (B2B) oder Business-to-Consumer (B2C) Portalen, sondern der Übergang gestaltet sich hier eher fließend.

Unternehmensportale können dabei sowohl horizontal (Informationen, Dienste, Prozesse, etc.) als auch vertikal (Geschäftsbereiche, Funktionen, etc.) ausgerichtet sein. Die Ausprägung als horizontal bzw. vertikal richtet sich dabei primär nach der Adressatengruppe (Kunden, Partner, Mitarbeiter, etc.).

E-Commerce-Portale, Business to Consumer Portale (B2C)

Diese Portale richten sich direkt an den Konsumenten, im Vordergrund steht die Werbung und der Verkauf von Produkten. Dabei dienen die E-Commerce-Portale nicht nur als virtuelle Schaufenster, sondern es wird vor allem die Gelegenheit zum Kauf geboten (z.B. Amazon.com). Dabei geht die Bandbreite dieser Portale von Einzelshops (z.B. Niedermeyer.at) bis hin zu Shopping Malls (z.B. Evita.de, Netzmarkt.de).

Nischenportale

Bei Nischenportalen steht das gemeinsame Interesse der Benutzer an einem bestimmten Thema im Vordergrund. Diese Portale beziehen sich auf ein spezifisches Themengebiet und versuchen alle relevanten Informationen zu diesen Gebieten aufzuzeigen und darzubieten. Vertreter dieser Nischenportale sind etwa Garten.at, das Informationen über Österreichs Gehölze, einen Leitfaden zur grünen Umweltgestaltung im Garten, Forst und Landwirtschaft, etc. anbietet oder auch Auto.at – der ersten Adresse für Autofreunde.

Demographische Portale

Unter demographischen Portalen sind Portale zu Verstehen, die von einer bestimmten Gemeinschaft für ihre Mitglieder und Interessenten angeboten werden. Bei solchen Gemeinschaften handelt es sich vorwiegend um Berufsverbände, Vereine, Klubs, etc.

Da vertikale Portale stets themenspezifisch sind, sind auch die typischen Bestandteile recht unterschiedlich. Daher wird hierbei je nach der Ebene auf der die Information angeboten wird unterschieden ([CRE], [SOM]):

- Global:
  - Verweise auf untergeordnete Einrichtungen nach verschiedenen Kriterien:
    - Regional
    - Verschiedene Gruppen von Inhalten
  - Allgemeine Informationen, wie Wetter, etc.
  - Informationen über die Unternehmensstruktur (Organigramme) bzw. über den Aufbau des Vereins, des Berufsverbands, etc.
  - Informationen über Aktienkurse, Geschäftsberichte, etc.
  - Freie Arbeitsstellen
  - Online-Shopping, Verweise auf Händler
- Mittlere Ebenen:
  - Verweise auf übergeordnete, untergeordnete und benachbarte Einrichtungen
  - Aktuelle Meldungen und Nachrichten, Veranstaltungshinweise
  - Chat, Foren
  - Zusammengefasste Inhalte untergeordneter Instanzen
  - E-Mail, Adressbuch, Kalender, Aufgabenlisten, etc.
  - Informationen über Weiterbildungsmöglichkeiten
  - Workflow- bzw. Groupware-Anwendungen, Prozessinformationen, Anwendungen, die den Zugriff auf Datenbanken einschließen

- Interne Ebene:
  - Verweise auf übergeordnete und benachbarte Einrichtungen
  - Lokale Inhaltsangebote
  - IP Telefonie
- Auf allen Ebenen: Suchmaschinen und Linksammlungen.

3.2.3. Aufbau eines Portals

Beim Aufbau eines Portals stehen vor allem zwei Aspekte im Vordergrund - der geschäftliche und der technische Aspekt.

Geschäftlicher Aspekt: Ein Portal ist eine Website mit dem wesentlichen Ziel, Zugang zu anderen Sites bzw. zu für den Benutzer wichtigen Informationen zu verschaffen. Dabei spielen natürlich auch Verträge über die Verweise auf eventuelle dritte Parteien auf der Site, Verkaufsförderung und Werbung eine wichtige Rolle. Selbst wenn man nur Informationen des eigenen Systems anbietet, so stellt sich doch die Frage, der Einnahmen durch Werbung bzw. des Marketings durch Werbebanner. Bei beiden Aspekten sollte darauf geachtet werden, nur Werbung für Produkte zu erlauben, mit denen sich das Portal auch identifizieren kann und es sollte auch nur Werbung für das Portal auf Partnerseiten geschaltet werden, die mit dem Konzept und dem gewünschten Image des Portals im Einklang zu bringen sind. Durch die graphische Gestaltung und durch die Inhalte (Hinweise auf interessante Ereignisse, Nachrichten, Artikel, Wissenswertes, Unternehmen und URL-Adressen, etc.), die am Portal für alle potenziellen Benutzer präsentiert werden, kann die Akzeptanz und vor allem der Ruf eines Portals wesentlich gesteuert werden, weshalb gerade auch dem technischen Aspekt viel Bedeutung zukommt.

Technischer Aspekt: Für den Aufbau jedes Portal bedarf es einer Infrastruktur - Hardware, Software und Kommunikationstechnik, die entweder durch die Unternehmen selbst oder durch Drittanbieters solcher Dienstleistungen, etwa einem ISP (Internet Service Provider, Anbieter von Internetdienstleistungen) aufgebaut bzw. gewartet werden kann. Als essentiell hat sich die Wahl eines (oder mehrerer) passenden Domänen-Namens herausgestellt. Diese Namen müssen registriert werden und stellen zukünftig die Grundlage für das öffentliche Auftreten und die Werbung dar. Wichtig ist es, einen einprägsamen gut klingenden Namen zu

wählen, der direkt mit dem Unternehmen, dem Produkt, etc. in Verbindung gebracht werden kann. Bei der technischen Umsetzung von Portalseiten ist es besonders wichtig darauf zu achten, dass die Seiten schnell geladen werden können (so viele Bilder als nötig, aber nicht mehr, kleine Bilder, viel Text, etc.). Jede zusätzliche Verzögerung könnte die potentiellen Benutzer verärgern und somit das Portal unattraktiv machen. Durch das Webdesign soll der Benutzer höchstmögliche Bequemlichkeit in der Benutzung haben. Wichtige Informationen im Blickfeld des Benutzers, wie auch einfaches An- bzw. Abmelden sollte unterstützt werden. Es sollte auch eine Personalisierung der Benutzer erfolgen.

Dies kann in mehreren Stufen erfolgen:

- Erste Stufe: Die Benutzer geben über ein Formular ihre Interessen, Gewohnheiten, Hobbies, etc. bekannt und künftig werden die Informationen für sie unter Berücksichtigung ihrer Daten zusammen gestellt.

- Zweite Stufe: Die Benutzer können diverse Einstellungen, wie Farben, Hintergrund, Schriftarten, etc. selbst vornehmen und somit individuell die Einstiegsseite gestalten.

- Dritte Stufe: Die Benutzer werden beobachtet, sprich ihr Verhalten wird im Hintergrund mitgespeichert und entsprechend ausgewertet. Dienste, die durch den Benutzer immer bzw. oft in Anspruch genommen werden, werden dann bereits in der Einstiegsseite platziert, andere hingegen werden hintan gestellt.

3.3. Agentensysteme/Agentenplattformen

Agentenplattformen ermöglichen einerseits das Entwickeln von Agenten und dienen andererseits als Laufzeitumgebung, wobei die Agenten bestimmte Dienste für die Benutzer ausführen. In der Literatur gibt es zahlreiche Beispiele für Agentenplattformen ([DIK01]), Aglets (mittlerweile open source [AGL]), Grasshopper ([GRA], [MAG]), Hive ([MIT]), etc.

Wenn sich mehrere Agenten auf einer Plattform befinden, gibt es die Möglichkeit, dass die Agenten unabhängig voneinander arbeiten ohne miteinander interagieren. Es besteht jedoch auch die Möglichkeit, dass die Agenten miteinander kommunizieren, miteinander bzw. füreinander Arbeiten und allenfalls an einem gemeinsamen Ziel arbeiten. Diese Aspekte werden im folgenden Kapitel genauer betrachtet.

3.3.1. Multiagentensysteme (MAS)

3.3.1.1. Definition

Ein Multiagentensystem ist ein System, das sich aus mehreren interagierenden Agenten zusammensetzt. Die Agenten sollen dabei ihr Wissen, ihre Ziele, Fähigkeiten und Pläne miteinander koordinieren und somit ihr Zusammenleben und ihr gemeinsames Handeln regeln. Dadurch können sie mehr erreichen, als jeder Agent für sich alleine erreichen könnte.

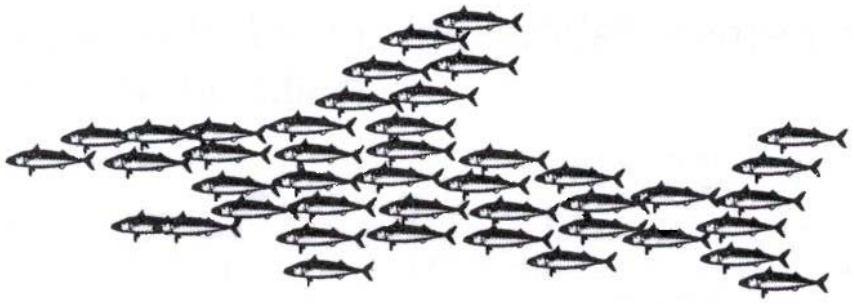


Abbildung 13: Multiagentensystem – das Ganze ist mehr als die Summe seiner Teile [JAC01]

Multiagentensysteme lassen sich wie folgt charakterisieren:

- Im Allgemeinen handelt es sich bei Agentensystemen um offene Systeme, d.h. die Anzahl der Agenten im System ist variabel, Agenten können hinzugefügt bzw. entfernt werden.

- Die einzelnen Agenten haben dabei unterschiedliche Fähigkeiten und keine globale Sicht. Kein Agent verfügt über das gesamte Wissen des Systems. Jeder Agent kennt nur die für ihn relevanten Informationen und die Handlungen der Agenten mit denen er kommuniziert.

- Auch die Kontrolle ist verteilt. Jeder Agent kann autonom Handlungen anstoßen und entscheiden, welche Ziele er verfolgt.

3.3.1.2. Agentenkommunikation

Unerlässlich für ein Multiagentensystem ([WOO02]) sind Kommunikation und Interaktion.

Die Interaktion definiert den Kommunikationsablauf, sprich die Struktur der Kommunikation. Interaktion kann sowohl zwischen Agenten (Agent-Agent-Interaktion) als auch zwischen Agenten und ihrer Umwelt (Agent-Umwelt-Interaktion) stattfinden. Interaktion kann nicht nur direkt über Kommunikation erfolgen, sondern auch indirekt durch die Beobachtung von Umweltveränderungen geschehen.

Kommunikation erfolgt normalerweise asynchron über Nachrichten. Dabei wird ein gemeinsames Protokoll, eine gemeinsame Sprache (Syntax und Semantik) verwendet. Damit der Agent durch Kommunikation nicht in seiner Arbeit unterbrochen wird, sind die Kommunikations- und die Funktionseinheit meist als getrennte Prozesse realisiert.

Auf die konkrete Umsetzung und auf diverse Werkzeuge die für die Kommunikation zur Verfügung stehen, wird ausführlich in Kapitel „4.3 Werkzeuge und Methoden“ eingegangen.

Interaktion oder Agentenkommunikation kann je nachdem, wie die Agenten miteinander in Beziehung stehen, verschiedene Ausprägungen haben. Die folgende Graphik soll die Grundtypen und ihre Zusammenhänge verdeutlichen:

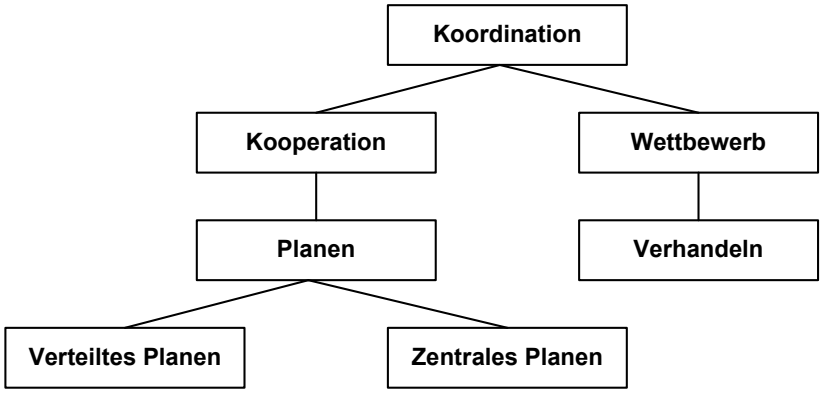


Abbildung 14: Interaktionstypen in Multiagentensystemen

Von Koordination spricht man dann, wenn von der gegenseitigen Abstimmung von Aktivitäten die Rede ist. Ob dies in Kooperation oder im Wettbewerb erfolgt, hängt von der jeweiligen Situation ab und ist somit von Fall zu Fall unterschiedlich. Die Notwendigkeit von Koordination wird in [NWA96b] wie folgt geschildert:

Verhindern von Anarchie und Chaos

Bei dezentralen Multiagentensystemen in denen jeder Agent nur eine eingeschränkte Sicht des Gesamtsystems hat, verfügen die Agenten nur über eine lokale Sicht, Ziele und Wissen. Folglich können sie auch in Konflikt mit anderen Agenten stehen. Wenn die Agenten die Freiheit hätten, jede Art von Verbindung und Arrangement einzugehen, dann kann dies zu Anarchie und Chaos führen. Um daher gemeinsame Ziele verfolgen zu können, müssen die Agenten unbedingt untereinander koordiniert werden.

Einhalten von globalen Bedingungen

Damit ein System erfolgreich arbeiten kann, müssen bestimmte Bedingungen und Regeln eingehalten werden. Beispielsweise erfordert die Einhaltung eines gemeinsamen Budgets oder der exklusive Zugriff auf bestimmte Ressourcen ebenfalls Koordination.

Verteiltes Wissen, verteilte Ressourcen oder verteilte Information

Agenten können unterschiedliche Fähigkeiten aufweisen und über spezialisiertes Wissen verfügen. Möglich wäre auch, dass sie Zugriff auf unterschiedliche Informationsquellen, auf differente Ressourcen (wie Speicher, Prozessoren, Ausgabegeräte, etc.) haben oder über verschiedene Zugriffsrechte, ... verfügen. Unter Einbeziehung solcher Szenarien ist Koordination unabdingbar. Dies ist vergleichbar mit einem Krankenhaus, in dem die Rettung, die Pfleger, Krankenschwestern und Ärzte untereinander koordiniert werden müssen. Jeder verfügt über Spezialwissen in einem bestimmten Bereich, keiner kann alles wissen und keiner kann alles alleine machen.

Abhängigkeiten zwischen Agentenaktionen

Die Ziele der Agenten sind meist voneinander unabhängig. Jedoch ist auch hier Koordination erforderlich, wenn die Ergebnisse in einer bestimmten Reihenfolge benötigt werden, um ein Gesamtziel zu erreichen. Man stelle sich vor zwei Agenten sollen das Blocks World Problem lösen.

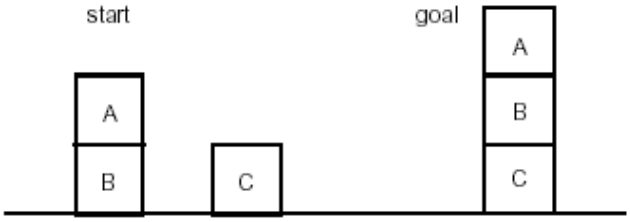


Abbildung 15: Anordnung von Blöcken (Blocks World Problem) [NWA96b]

Die einfachste Weise, dies zu lösen wäre, dass der erste Agent B auf C stapelt. Der zweite Agent muss warten, und danach A auf B-C stapeln um ABC zu erzielen. Offenbar sind die beiden Agenten voneinander abhängig; der zweite Agent muss warten, dass der erste Agent sein Sub-Ziel durchführt, bevor es sein Sub-Ziel durchführen kann. Wo solche Abhängigkeiten bestehen müssen die Tätigkeiten der Agenten koordiniert werden.

Effizienz

Selbst, wenn Agenten alle Fähigkeiten und Möglichkeiten haben, um allein zu arbeiten, kann Koordination ([EXC02]) sinnvoll sein. Information, die bereits von einem Agenten gewonnen wurde, kann auch für andere interessant sein und vice versa. Koordination kann hier dazu führen, dass die Agenten schneller und effizienter arbeiten können.

Koordination kann ohne Kooperation auskommen, wenn Agenten im Wettbewerb miteinander stehen, unterschiedliche Ziele verfolgt werden und somit nur auf die Tätigkeiten anderer Agenten reagiert wird bzw. wenn es zwingend notwendig ist zu verhandeln. Kooperation hingegen bedeutet, dass Agenten ein gemeinsames Ziel vereinbaren und verfolgen. Kooperation ohne Koordination ist somit nicht sinnvoll, ist uneffizient und zu fehlerträchtig.

Um Kooperation zwischen den einzelnen Agenten gewährleisten zu können, müssen die Aktivitäten geplant werden. Planen hilft Inkonsistenzen und Konflikte zu vermeiden ([DUR01]). Es gibt zwei Arten von Multiagentenplanung: zentrales Planen und dezentrales Planen. Auf diese Planungsarten wird im Detail im Kapitel „Interaktionsprotokolle - Multiagentenplanung“ eingegangen.

Abbildung 16 zeigt ein Beispiel für ein Informationsmultiagentensystem.

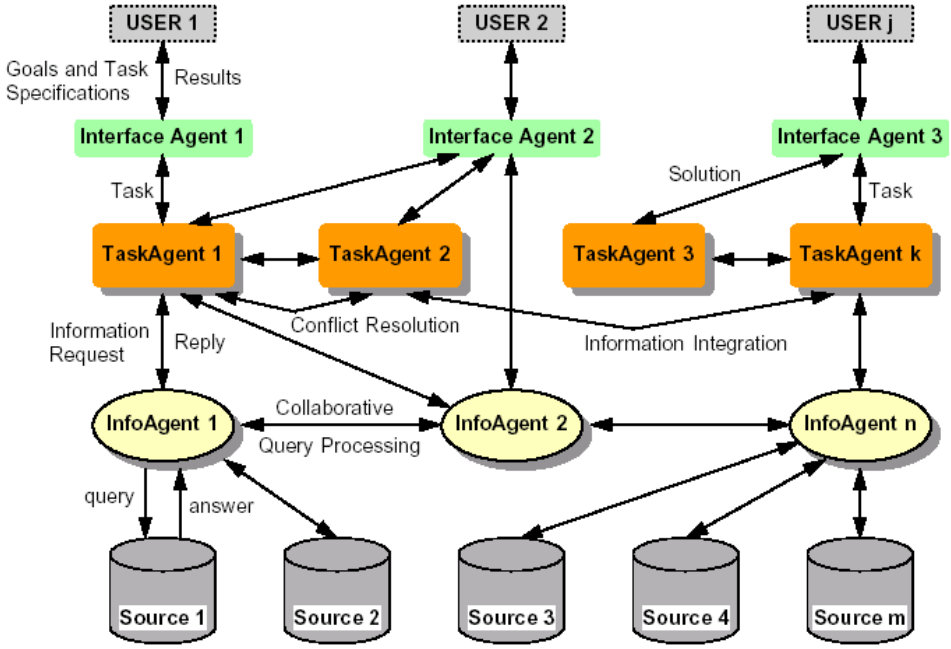


Abbildung 16: Beispiel eines Informationsmultiagentensystems [SYC96]

Dabei hat jeder Benutzer eine eigene Ansprechstelle, sprich jeweils einen Interface-Agenten mit dem er kommuniziert. Die Interface-Agenten leiten die Anfrage entweder direkt an Informationsagenten bzw. indirekt über Task-Agenten weiter. Die Task-Agenten versuchen die ihnen übertragene Aufgabe zu lösen. Dabei entscheiden sie, welche Informationsagenten sie beauftragen. Bei manchen Anfragen kooperieren sie miteinander, bei anderen Anfragen stehen sie miteinander im Konflikt und müssen diesen durch Kooperation lösen. Auch die Informationsagenten kommunizieren miteinander und tauschen Informationen aus. Der erste Informationsagent (InfoAgent1) bezieht sein Wissen aus den Informationsquellen 1 und 2 (Source1, Source2). InfoAgentN sucht Source3 bis SourceM auf. Beide zusammen haben genügend Wissen, sodass InfoAgent2 auf keine Informationsquelle mehr zugreifen muss, ihm genügt das Wissen, dass er durch Kooperation mit InfoAgent1 und InfoAgent3 bekommt. InfoAgent2 leitet sein Wissen direkt an InterfaceAgent2 und an TaskAgent1 weiter. User2 bekommt somit seine Frage unter Mithilfe von InfoAgent1 und InfoAgentN beantwortet ohne dass der beauftragte InfoAgent2 zusätzlich recherchieren musste. Die Beantwortung wurde durch die Kooperation daher wesentlich effizienter. User1 profitiert ebenso durch die Koordination. Auch er nützt das Wissen aller Informationsagenten und somit wird seine Lösung präziser. Der letzte User bekommt zwar nur direkt Ergebnisse des InfoAgentN, sein

TaskAgent kooperiert allerdings mit anderen Task-Agenten. Somit verdichtet sich auch hier die Information, sodass auch dieser Benutzer vom Multiagentensystem profitiert.

3.3.1.3. Interaktionsprotokolle

Die Interaktion der einzelnen Agenten muss organisiert werden. Dafür gibt es unterschiedliche Interaktionsprotokolle bzw. Organisationsstrukturen.

3.3.1.3.1. Organisatorische Strukturierung

Die einfachste Form der Koordination liegt dann vor, wenn die Organisation die Rechte, die Verantwortlichkeiten, die Fähigkeiten, den Kontrollfluss definiert. Dabei werden Rollen, Kommunikationspfade und eventuell eine hierarchische Rangordnung festgelegt.

Die **Master-Slave Technik** ist eine hierarchische Organisationsform. Sie gleicht der Client-Server Technik. Es gibt einen Masteragenten, der die Planung übernimmt und Aufträge an die Slaveagenten vergibt. Die Slaveagenten haben einen festen Tätigkeitsbereich. Sie verfügen nur eine partielle Autonomie in der Hinsicht, dass sie ihre Ergebnisse dem Master berichten müssen. Ob sie untereinander kommunizieren oder nicht, obliegt ihrer Entscheidung. Diese Form der Organisation widerspricht jedoch dem Grundprinzip, dass Multiagentensysteme dezentral organisiert sein sollen. Auch ist kein offenes System mehr möglich, da beim Hinzufügen und Entfernen von Agenten die hierarchische Struktur angepasst werden müsste und dadurch zu aufwendig wird. Die Master-Slave Technik ist demzufolge unflexibel und störungsanfällig. Falls ein Agent funktionsuntüchtig wird, können in der Regel auch die hierarchisch untergeordneten Agenten nicht mehr genützt werden.

Eine andere Organisationsform ist die **Blackboard-Architektur** [HAY85]. Die “knowledge sources” des Systems werden durch Agenten ersetzt, die Beiträge an das Blackboard liefern und andere Beiträge lesen, solange bis eine Lösung für die Aufgabe gefunden wird. Die Agenten kommunizieren somit nicht direkt miteinander, sondern nur über das Blackboard. Ein Masteragent regelt dabei den Ablauf der Lese- und Schreibzugriffe. Er stellt dabei einen wechselseitigen Ausschluss (mutual exclusion) der Agentenzugriffe auf die Ressourcen sicher ([JEN98], [DIX92]) und steuert dadurch den Problemlösungsprozess. Der Vorteil des Blackboard-Systems liegt darin, dass es keine Einschränkung der zu lösenden Probleme gibt. Die Agenten sind unabhängig voneinander und nicht auf die Zusammenarbeit angewiesen.

Problematisch ist jedoch, dass alle Agenten eine gemeinsame Sprache benötigen, um die Beiträge auch richtig interpretieren zu können. Bei einer großen Anzahl von Agenten kann sich das Blackboard als Flaschenhals (bottleneck) herausstellen, da der Masteragent gleichzeitiges Lesen und Schreiben verhindert.

3.3.1.3.2. Vertragsnetzprotokoll

Beim Vertragsnetzprotokoll (Contract-Net Protocol) [DAV83, SMI80] kann ein Agent zwei Rollen annehmen. Er kann als Manager oder als Anbieter auftreten.

Ein Manager unterteilt ein Problem in mehrere Teilprobleme und sucht Anbieter dafür, die die Ausführung der Teilprobleme übernehmen. Schließlich verarbeitet der Manager alle Teilergebnisse zu einem Gesamtergebnis.

Die Anbieter übernehmen die Ausführung einer Teilaufgabe und liefern dem Manager ihre Ergebnisse. Dabei können sie selbst rekursiv zum Manager werden und andere Anbieter mit Teilaufgaben beauftragen.

Der Ablauf der Aufgabenausführung hat daher folgendes Aussehen:

Schritt 1: Der Manager unterteilt das Problem, beschreibt die Teilaufgaben und sendet eine Nachricht mit der Aufgabenspezifikation an die in Frage kommenden Agenten.

Schritt 2: Die Agenten prüfen, ob sie in der Lage sind den Auftrag zu übernehmen (ausreichend Ressourcen, Fähigkeiten, etc.).

Schritt 3: Die Agenten, die den Auftrag übernehmen können senden dem Manager eine Notifikation, in der sie auch die erwarte Dauer oder Qualität beschreiben können.

Schritt 4: Der Manager empfängt die Angebote, wertet diese aus und beauftragt einen Agenten. Dabei muss er nicht alle Angebote abwarten, sondern entscheidet autonom, welches Angebot er wählt.

Ein wesentlicher Vorteil des Vertragsnetzprotokolls ist, dass es dezentral aufgebaut und selbstorganisierend ist. Agenten können beliebig hinzugefügt und entfernt werden und durch die freie Wahl der übernommenen Aufträge entsteht ein natürlicher Lastausgleich. Die Nachteile sind darin zu sehen, dass Aufträge an Agenten vergeben werden, die weniger qualifiziert

sind als andere, beispielsweise, wenn die qualifizierten Agenten zum Zeitpunkt der Ausschreibung beschäftigt waren und den Auftrag nicht übernehmen konnten. Zusätzlich muss das System einen starken Nachrichtenverkehr bewältigen, der durch die Anfragen und Antworten entsteht. Das Vertragsnetzprotokoll ist dann sinnvoll, wenn die Agenten vielseitig sind und daher unterschiedliche Aufgaben bearbeiten können und die Aufgaben so beschaffen sind, dass sie leicht unterteilt werden können.

3.3.1.3.3. Multiagentenplanung

Die Multiagentenplanung ist in zwei Typen eingeteilt: zentrales Planen und verteiltes Planen.

Zentrales Planen: Hier wird der Plan von einem Agenten konzipiert. Dieser koordiniert alle partiellen Agentenpläne und versucht Konflikte zu eliminieren. Der Koordinationsagent versucht die einzelnen Pläne entsprechend dem Gesamtziel zu modifizieren und erstellt einen Multiagentenplan. Leider weist zentrales Planen aber ähnliche Nachteile, wie auch die Master-Slave Technik auf. Außerdem müssen die Agenten bereit sein, wesentliche Informationen miteinander zu teilen, bzw. den anderen Agenten zugänglich zu machen. Beispiele für zentrales Planen sind in [GEO83], [GEO84], [CAM83] und [JIN90] zu finden.

Verteiltes Planen: Hier ist die Grundidee die, jedem Agenten ein Model der Pläne der anderen Agenten zu geben. Die Agenten kommunizieren danach untereinander und adaptieren ihre Pläne, solange bis alle Konflikte beseitigt sind. Auch hier müssen die Agenten wichtige Informationen miteinander teilen, auch der Kommunikationsbedarf ist hier sehr hoch und der Planungsprozess ist wesentlich komplexer gegenüber dem zentralen Planen. Beispiele für verteiltes Planen sind etwa in [LES81] und in [DUR87] zu finden.

Zusammenfassend zeichnen sich Multiagentensysteme daher durch folgende Eigenschaften aus ([NWA96b], [WEI00]):

- Ein Multiagentensystem ist gekennzeichnet durch eine gewisse Intelligenz. Die Agenten in einem Multiagentensystem interagieren über definierte Schnittstellen und Protokolle miteinander. Die Wiederverwendbarkeit der Agenten steigt daher. Sie können auch in anderen Systemen, die die gleichen Schnittstellen und Protokolle unterstützen problemlos eingesetzt werden. Durch die Standardisierung können die Agenten vollkommen unabhängig voneinander entwickelt werden.

- Jeder Agent hat bestimmte Fähigkeiten. In einem Multiagentensystem können die Agenten miteinander kommunizieren und ihre Informationen und Ergebnisse miteinander teilen. Die Agenten können dadurch mehr Wissen gewinnen und Informationen müssen nicht mehrfach eingeholt werden.

- Gibt es mehrere Agenten mit gleichen oder ähnlichen Fähigkeiten im System, so erhöht dies die Fehlertoleranz und die Ausfallssicherheit.

- Ist eine Aufgabe teilbar, so können mehrere Agenten mit Teilaufgaben beauftragt werden, die sie parallel durchführen. Dadurch sinkt die Problemlösungsdauer.

3.4. Beziehung Portal und Agentenplattform

Portale, aber auch intelligente Agenten gewinnen in der Welt des Internets immer mehr an Bedeutung. Durch die steigende Zahl von Informationen und Möglichkeiten im Internet und wird es immer schwieriger für den Einzelnen die Übersicht zu bewahren, die gewünschten Informationen zu bekommen bzw. Systeme zu überwachen oder immer wiederkehrende Tätigkeiten zu erledigen, da auch hier die Vielfalt zunimmt. Zukünftig werden daher intelligente Agenten und spezifische Portale in Kombination angeboten.

Intelligente Agenten werden dabei sehr stark in den Bereichen:

- Informationsmanagement
- Personalisierung
- Ereignisüberwachung und Automatisierung

zum Einsatz kommen.

3.4.1. Informationsmanagement

Agenten können den Benutzer wesentlich dabei unterstützen Informationen aus der Fülle der im Internet verfügbaren Daten herauszufiltern. Dabei gestaltet sich diese Suche für die Benutzer oftmals sehr aufwendig. Angenommen, ein Benutzer sucht Information über ein bestimmtes Interessensgebiet, so bedient er sich bislang einer Suchmaschine. Je nachdem, wie weit der Benutzer seine Suche durch Schlüsselwörter einschränkt, bekommt er eine mehr oder minder große Anzahl von gefundenen Seiten. Aber auch bei sehr starker Einschränkung der

Suchbegriffe kann es sein, dass hunderte oder gar tausende gefundene Webseiten aufgelistet werden. Anschließend muss der Benutzer abermals versuchen, die Suche einzuschränken, oder aber es wird, meist wahllos, ein Subset dieser gefunden Webseiten ausgewählt und direkt angesehen.

Damit ist die Suche jedoch nicht abgeschlossen: Jetzt erst kann damit begonnen werden, die Informationen auf den Webseiten auszuwerten. Dabei muss analysiert werden, ob das Gesuchte in der Form, wie man die Information benötigt, auf den Seiten angeboten wird. Wenn nicht muss man seine Suche wieder fortsetzen.

Diese Vorgehensweise ist für die Benutzer sehr aufwendig und nimmt viel Zeit in Anspruch. Auch gibt es keine Garantie, wirklich die Information zu bekommen, die man benötigt.

Noch aufwendiger und zeitintensiver kann eine Suche nach einem bestimmten Produkt werden, besonders dann, wenn man dieses Produkt auch gleich bestellen möchte.

Ein Beispiel: Ein Benutzer sucht das beste Angebot für ein bestimmtes Produkt und möchte dieses auch gleich bestellen. Auf herkömmliche Weise erfolgt die Suche nach dem Produkt analog wie die beschriebene Suche nach Informationen üblicherweise über Suchmaschinen. Anschließend müssen aus diesen gefundenen Webseiten durch den Benutzer erst die benötigten Informationen über das Produkt selbst, aber meist auch andere Informationen, wie Lieferbedingungen, Zahlungsarten usw. heraus gefiltert werden und die Resultate analysiert werden. Weiters müssen die verschiedenen Angebote untereinander verglichen werden. Dabei werden abermals Informationen wie Preis, Services, Versand, usw. gegenübergestellt und gegeneinander abgewogen. Abschließend muss der Benutzer zur Webseite mit dem besten Angebot zurückkehren und ein Formular für die Bestellung (Produkt, Anzahl, Versand-adresse, Kreditkartennummer, Art der Anlieferung, etc.) ausfüllen.

All dies erfordert enormen Zeitaufwand und kann je nach Anbindung (Modem, ISDN-Verbindung, ADSL, etc.) des Benutzers kostenintensiv werden.

Es ist daher zweckmäßig, die Suche nach Information und eventuell auch die Bestellung durch Agenten durchführen zu lassen. Die Aufgabe des Benutzers beschränkt sich dabei auf die „Auftragsvergabe“, die Spezifikation des Geforderten und auf die Entgegennahme der Resultate beschränken. Die zeitraubende Aufgabe der Suche, des Filterns und der Analyse der

Webseiten und auch der Aufbereitung der Ergebnisse wird dabei vollständig durch andere übernommen und durchgeführt. Dabei kann man auch Bestellungen abwickeln lassen, indem man dem Beauftragten z.B. eine maximale Summe bekannt gibt und die gewünschte Zahlungsart spezifiziert.

Wenn Agenten diese Aufgabe für den Benutzer übernehmen sollen, müssen diese bestimmte Fähigkeiten aufweisen. Die zum Einsatz kommenden intelligenten Agenten sollten mobile Agenten sein, d.h., diese Agenten finden ihren Weg durch das Internet selbständig zu den benötigten Websites und filtern diese nach den benötigten Informationen. Manche dieser Agenten müssen auch in der Lage sein, Verhandlungen zu führen und Käufe abzuschließen ([MUE01a], [LOM01], [VET01]). Deshalb führt dieser Bereich bereits in die Automation von Aufgaben (siehe Kapitel 3.4.3 Ereignisüberwachung und Automatisierung).

3.4.2. Personalisierung

Hand in Hand mit einem verbesserten Informationsmanagement geht die Personalisierung. Durch Personalisierung wird der Benutzer und dessen Interessen besser bekannt und die bereitgestellte Information kann maßgeschneidert werden. Eine perfekt personalisierte Seite sollte demnach dem Benutzer genau die Informationen zeigen, die er benötigt.

Diese Personalisierung kann abermals durch Agenten ([LIE01a]) erfolgen. Dabei überwachen die Personalisierungsagenten die Verhaltensweisen der Benutzer. Sie achten auf die relative Verweildauer auf bestimmten Webseiten, auf die Klickfrequenz, wie häufig bestimmte Webseiten aufgerufen werden, etc.

Mit diesen Informationen und den allgemeinen Benutzerinformationen (Alter, Geschlecht, Familienstand, etc.), die durch Fragebögen erfragt werden, kann ein individuelles Benutzerprofil erstellt werden, das die Interessen und das Verhalten des Benutzers berücksichtigt. Durch Personalisierung wird somit die Komplexität der Informationsaufbereitung und der individuellen Ansprache des Benutzers reduziert ([SON02a]).

3.4.3. Ereignisüberwachung und Automatisierung

Agenten können vor allem auch zur Ereignisüberwachung eingesetzt werden und somit dem Benutzer hilfreich zur Seite stehen.

Wenn der Benutzer beispielsweise seinen Emailverkehr über ein Portal abwickelt, Teilnehmer eines Forums ist bzw. wenn mehrere Benutzer Zugriff auf die Kalender anderer Benutzer haben, so können Agenten eingesetzt werden um Ereignisse (Hinzufügen, Ändern, Löschen von Emails, Beiträgen, Inhalten, Terminen, etc.) zu überwachen und die jeweiligen Benutzer zu verständigen. Der Einsatz der Agenten bietet den Vorteil, dass die Betreffenden nicht selbst periodisch (stunden- bzw. tageweise) ihre Arbeitsumgebung kontrollieren müssen, sondern beim Auftreten eines Ereignisses durch den Agenten verständigt werden.

Darüber hinaus können Aufgaben automatisiert werden durch den Einsatz von Agenten. Besonders, indem einfache wiederkehrende bzw. lang dauernde Tätigkeiten durch Agenten übernommen werden, können Benutzer unterstützt und entlastet werden.

Es ist abzusehen, dass zukünftig auch immer öfter anspruchsvolle Tätigkeiten an Agenten übertragen werden können. Schon jetzt gibt es einige Shopping-Agenten (z.B. mySimon [SIM]), die die Fähigkeit haben nach bestimmten Produkten zu suchen und Preise zu vergleichen. In Zukunft werden die Agenten immer mehr zu unseren persönlichen Assistenten werden, die die unterschiedlichsten Tätigkeiten für uns ausführen werden und somit für uns als Email-Melder und –Vorleser, als Adress-, Termin-, Routen- und Reiseplaner, als Passwortmanager, als Immobilienmakler, als Einkäufer oder Ratgeber, als Aktienbroker, als Buchführer, als Jobsucher, etc. agieren werden.

4. AGENTEN ALS NEUES SOFTWAREENGINEERING-PARADIGMA

Agenten stellen durch ihre Komplexität neue Anforderungen an die Entwickler. Durch ihr autonomes Verhalten, sprich aufgrund der Tatsache, dass Agenten selbst Entscheidungen treffen, mit anderen kooperieren sollen, aber auch mitunter mit anderen Agenten in Konkurrenz stehen können und auf ihre Umwelt reagieren und einwirken können sollen, bedarf es auch anderer bzw. adaptierter Methodiken und Werkzeuge als sie derzeit im Bereich des Softwareengineering im Einsatz sind.

Im folgenden sollen daher einige Methodiken bzw. Programmiermodelle gegenübergestellt werden und ihre Einsatzmöglichkeit für Agentenprogrammierung aufgezeigt werden.

4.1. Gegenüberstellung von Modellen

4.1.1. Client-Server-Modell

Beim Client Server Modell ist der Client derjenige, der einen Dienst benötigt und der Server, derjenige, der einen Dienst erbringt.

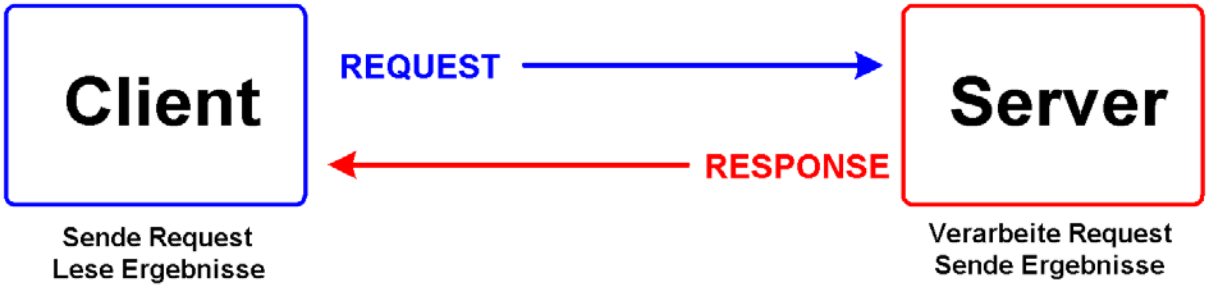


Abbildung 17: Client-Server-Modell

Die Interaktion zwischen Client und Server kann sowohl synchron als auch asynchron erfolgen. Es herrscht jedoch eine feste Rollenverteilung und wie in Abbildung 17 gezeigt, muss der Client einen Dienst anfordern und der Server antwortet ihm (Request – Response). Dies wird häufig über entfernte Prozeduraufrufe (Remote Procedure Calls, RPC) realisiert.

Der Client ist im wesentlichen dadurch charakterisiert, dass er bei Bedarf eine Anforderung an den Server in geeigneter Weise aufbereitet und anschließend initiiert. Daraus leitet sich ein punktueller Koordinationsbedarf ab und der Client bleibt jeweils anonym.

Der Server hingegen weißt einen kontinuierlich ablaufenden Prozess auf (permanenter Koordinationsbedarf), da er Operationen netzwerkweit bereitstellt. Dabei gibt es auch die Möglichkeit, dass ein Dienst nicht nur von einem, sondern von mehreren Servern erbracht wird.

4.1.2. Objektmodell

Ein Objekt ist eine Einheit, die einen Zustand (Daten) und eine Anzahl von Methoden kapselt. Die Methoden stehen dabei für Operationen die auf den Zustand wirken (diesen Abfragen, Verändern, etc.). Die Interaktion zwischen mehreren Objekten erfolgt über Methodenaufrufe. Die Objekte haben eine wohldefinierte Rolle in einem bestimmten Problembereich. Objekte können private Daten haben (d.h. Attribute, auf die nur das Objekt selbst zugreifen kann) und öffentliche Daten, die für beliebige Objekte zugreifbar und veränderbar sind ([RUM91]).

Objekte können unterschiedliche Beziehungen zueinander haben:

- **Äquivalenzrelation:** Im Sinne der Wiederverwendbarkeit werden sehr oft Objekte gleichen Typs benötigt. Daher wurden Klassen eingeführt, die die Eigenschaften abstrakt beschreiben. Klassen sind praktisch Vorlagen nach denen Objekte erstellt werden. Ein Objekt wird als Instanz einer Klasse bezeichnet. Instanzen der selben Klasse weisen somit die gleiche Struktur auf.

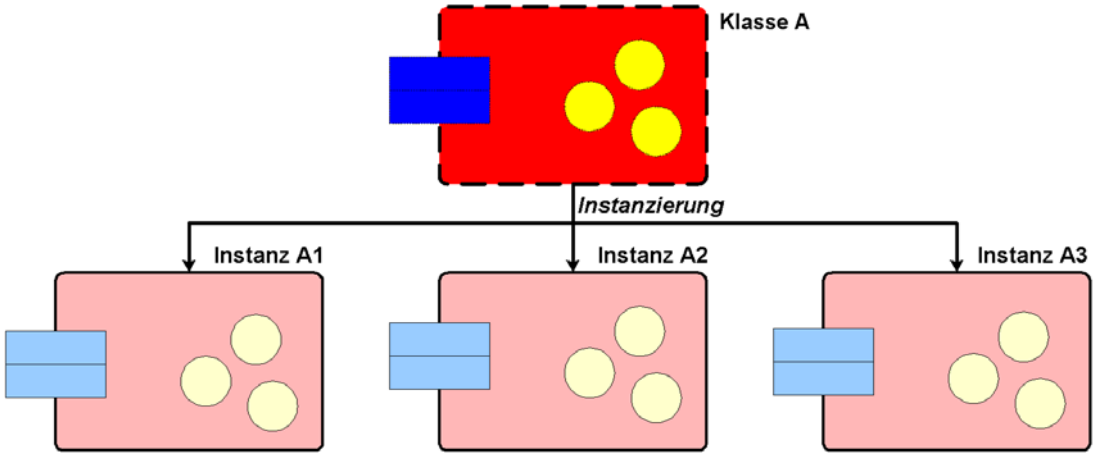


Abbildung 18: Objekte als Instanzen einer Klasse

- **Vererbung:** Dabei kann es einfache (eine Klasse kann nur von einer anderen Klasse erben) und mehrfache Vererbung (eine Klasse kann von mehr als einer anderen Klasse erben) geben. Bei der Vererbung bekommt eine abgeleitete Klasse die Datenstrukturen und Methoden der Basisklasse vererbt und kann diese erweitern.

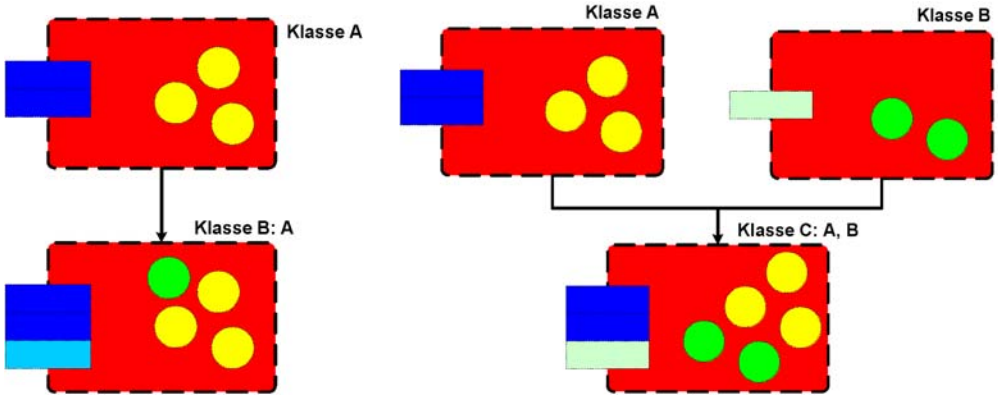


Abbildung 19: Einfache Vererbung und Mehrfachvererbung

- **Polymorphismus:** In der Klassenhierarchie kann es mehrere verschiedene Methoden mit gleichen Namen geben. Je nachdem, welche Klasse einem Objekt zugeordnet ist, verhält sich das Objekt anders, es ist polymorph. Nicht zur Übersetzungszeit wird entschieden, welche der möglichen Versionen ausgeführt wird, sondern erst zur Laufzeit des Programms fällt die Entscheidung über den Methodenaufruf.

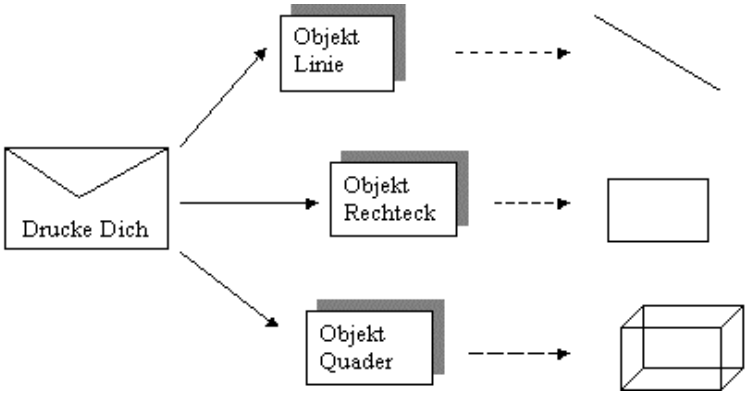


Abbildung 20: Polymorphismus

Auch bei der Interaktion zwischen Objekten gibt es eine feste Rollenverteilung. Prinzipiell ist die Interaktion ortsunabhängig und kann auch hier synchron oder asynchron erfolgen. Die Koordination erfolgt über Methodenaufrufe und im Gegensatz zum Client-Server-Modell kann hier jedes Objekt Methoden zur Verfügung stellen. Dabei korrespondieren die Methoden mit den möglichen Aktionen – die aufgerufenen Methoden müssen immer ausgeführt werden.

Objektorientierte Programmierung bietet insofern Vorteile, dass man eine Analogie zur realen Welt bilden kann. Darüber hinaus steigt durch objektorientierte Programmierung die Wiederverwendbarkeit und die Systeme sind durch mögliche Objektbeziehungen erweiterbar.

4.1.3. Agentenmodell

Ein Agent ist eine autonome Softwarekomponente. Sie sind meist reaktionsfähig, zielgerichtet, autonom und kooperationsfähig. Bei der Agentenkommunikation gibt es im Allgemeinen keine zentrale Instanz, es gibt beliebig viele Teilnehmer und somit auch beliebige Interaktionen. Folglich kann Wettbewerb um knappe Ressourcen entstehen und Zielkonflikte können durch unterschiedliche Prioritäten und divergierende Einschätzungen der Ziele aufbrechen.

Agenten entkoppeln Methoden und Aktionen. Agenten entscheiden selbst, ob sie Aktionen ausführen oder nicht.

4.1.4. Agenten versus Objekte

Agenten und Objekte haben unumstritten Gemeinsamkeiten. Jedoch unterscheiden sich Objekte und Agenten in einigen wesentlichen Punkten:

Bei der objektorientierten Programmierung (OOP) gibt es kein Konzept für autonomes und flexibles Verhalten von Objekten. Agenten hingegen haben oftmals einen hohen Grad an Autonomie und sind dadurch charakterisiert, dass sie eine explizite Kontrolle über ihren Zustand und ihr Verhalten haben.

Objekte können ihre Daten kapseln. Ein Objekt ist meist auf wenige Datentypen beschränkt und bietet eine Vielzahl von Methoden an. Dabei hat ein Objekt die Möglichkeit Datenelemente und Methoden als öffentlich oder als privat zu deklarieren. Private Datenelemente und Methoden haben den Vorteil, dass nur das Objekt selbst darauf zugreifen darf und kein anderes Objekt. Öffentliche hingegen, bieten allen Objekten die Möglichkeit diese aufzurufen. Objekte haben keine Kontrolle darüber, welche anderen Objekte ihre Methoden aufrufen. Sie haben somit keine Kontrolle über ihr eigenes Verhalten.

Agenten hingegen können sich in einem „feindlichen“ Umfeld befinden. Sie müssen darüber entscheiden können, wem sie ihre Dienste zur Verfügung stellen und in welchem Ausmaß. Sie müssen nicht nur ihre Daten, sondern auch ihr Verhalten kapseln können. Im Gegensatz zu Objekten verfügen sie meist über eine große Anzahl von Datenelementen (Wissen) und stellen wenige Methoden zur Verfügung.

Bei Objektorientierung entscheiden die aufrufenden Objekte, ob und wann ein Methodenaufruf erfolgt. Bei Agentenorientierung entscheidet der aufgerufene Agent, ob und wann er den Methodenaufruf durchführt. Dies lässt sich einfach aus der Tatsache herleiten, dass bei Objekten, die gemeinsam ein System bilden und ein gemeinsames Ziel vorausgesetzt werden kann. Bei Multi-Agenten-Systemen ist es sehr wahrscheinlich, dass jeder Agent ein anderes Ziel verfolgt. Es ist also nicht sichergestellt, dass ein Agent *i* eine Aktion (Methode) a ausführt, nur weil ein anderer Agent *j* dieses möchte. Wenn *j i* bittet a auszuführen, dann kann Agent *i* die Aktion *a* ausführen oder nicht, die Entscheidung liegt bei Agent *i* [CIA01].

Agenten müssen daher auch fähig sein Verhandlungen zu führen, miteinander zu kooperieren, zu planen und zu lernen (siehe auch Kapitel 3.3.1 Multiagentensysteme (MAS)).

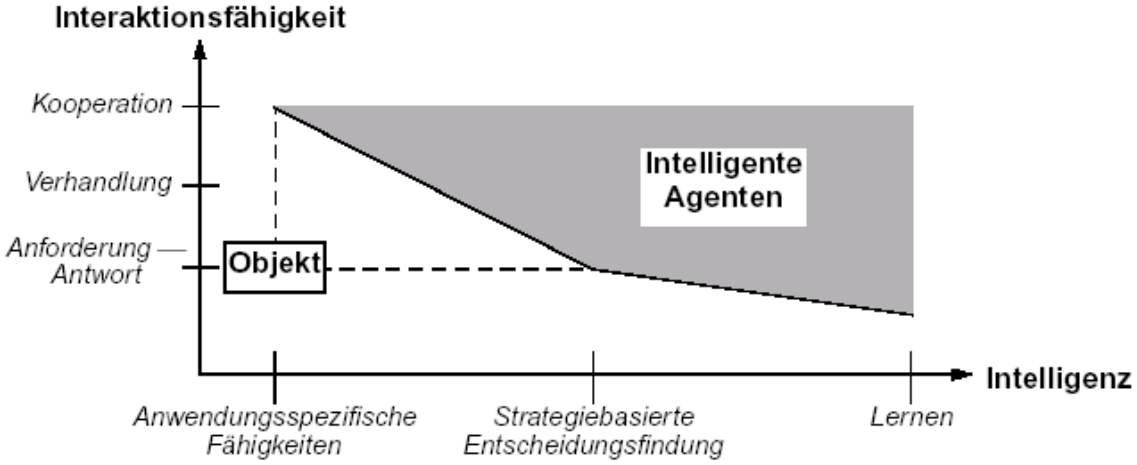


Abbildung 21: Gegenüberstellung der Fähigkeiten Objekt und Agent [BURa]

Zusätzlich ist im Allgemeinen beim objektorientierten Modell nur ein Thread vorgesehen. Hingegen hat jeder Agent einen eigenen Kontrollthread. Agenten sind von ihrem Wesen her kontinuierlich aktiv und aktualisieren typischerweise fortwährend ihren internen Zustand, beobachten ihre Umgebung, entscheiden über ihre weiteren Handlungen oder führen Aktivitäten aus. Demgegenüber sind Objekte meistens inaktiv. Nur wenn sie eine Nachricht von einem anderen Objekt empfangen führen sie die verlangte Methode aus. Ansätze dieses

Konzeptes findet man jedoch in der nebenläufigen Programmierung und beim Konzept der aktiven Objekte ([MUE01b], [DOU99]).

Wooldridge [CIA01] sieht die wesentlichen Unterschiede zwischen Objekten und Agenten so:

- Agenten sind autonomer als Objekte. Insbesondere entscheiden sie selbst ob sie eine Aktion ausführen, wenn sie ein anderer Agent darum bittet. Objekte müssen angeforderte Aktionen immer ausführen.
- Agenten sind zu einem flexiblen (reaktiven, pro-aktiven, sozialen) Verhalten fähig, im Standardobjektmodell hingegen ist ein derartiges Verhalten nicht vorgesehen.
- Ein Multiagentensystem ist immer multi-threaded, da jeder Agent zumindest einen eigenen Kontrollthread hat. Bei Standard-Objekten sind mehrere Threads nicht vorgesehen.

Zusammengefasst ergeben sich folgende Unterschiede zwischen Objekten und Agenten:

	OBJEKTE	AGENTEN
Eigenes Handeln	Keine eigene Kontrolle	Eigene autonome Entscheidungen
Handlungen erfolgen durch	Aufruf (muss Aufruf ausführen)	Anfrage (kann, aber muss der Anfrage nicht folge leisten)
Umgebung	Freundlich	Freundlich und Feindlich
Handlungsraum	(in der Regel) unflexibel	flexibel
Threads	Single-threaded	Multi-threaded

Tabelle 2: Unterschiede zwischen Objekten und Agenten

4.2. Agentenorientierte Softwareentwicklung (AOSE)

In den 90er Jahren haben mehrere Forscher ein neues Softwareentwicklungsparadigma aufgezeigt – die Agentenorientierte Softwareentwicklung (AOSE) ([JEN00], [WOO99], [WOO01], [TVE], [ZAM00]). Die Grundidee dabei ist der Aufbau eines Softwaresystems, bestehend aus einer Menge von Agenten (Multiagentensystem), die autonom, reaktiv, proaktiv und mit sozialen Fähigkeiten ausgestattet sein können. Diese Agenten sollen miteinander in Beziehung stehen und kommunizieren dabei über eine Agentensprache.

Dieser Ansatz ist durch die Komplexität von Agentensystemen entstanden. Dabei ist nicht die algorithmische Zeitkomplexität, im Sinne von NP-vollständig und Cook´s Theorem gemeint. Sondern darunter sei die Komplexität einer Systemspezifikation, eines Systementwurfs und der Konstruktion aus Sicht von Software-Engineering verstanden [WOO98].

Dabei sind vor allem folgende Systemeigenschaften ausschlaggebend [WOO98]:

Die Systemumgebung

Systemumgebungen sind charakterisiert dadurch, dass sie dynamisch oder statisch, zugänglich oder unzugänglich und deterministisch oder nicht-deterministisch sind. Dies beeinflusst natürlich auch den Entwicklungsprozess eines Systems. Im Allgemeinen ist die Entwicklung von Systemen, deren Umgebung dynamisch, unzugänglich und nicht-deterministisch ist komplexer als von Systemen deren Umgebung statisch, zugänglich und deterministisch ist.

Die Interaktion zwischen dem System und der Umgebung

In funktionellen Systemen (z.B. Compilern) ist die Interaktion mit der Umgebung einfach strukturiert. Das System nimmt die Eingaben, generiert daraus mittels Funktionen Ausgaben und terminiert. Komplexer ist die Interaktion mit Systemen (reaktiven Systemen), die eine fortwährende Beziehung mit ihrer Umgebung haben.

Offene Systeme

Oftmals kann sich die Struktur von Systemen dynamisch ändern und Systeme weisen eine starke Heterogenität der Komponenten ist. Agenten haben hier entgegen herkömmlicher Softwareentwicklungskonzepte den Vorteil, dass sie autonom und reaktiv sein können und deshalb ihre Ziele in einer unbekannten Umgebung leichter zu verfolgen vermögen.

Dezentrale Organisation

Systeme sind häufig verteilt in ihren Ressourcen und die Kontrolle von Abläufen ist dezentral organisiert. Für die Implementierung dieser Systeme bieten sich daher Agenten an.

AOSE stellt keine Abkehr, sondern eine Erweiterung des objektorientierten Paradigmas dar. Agenten können neben ihrem Zustand auch zielgerichtetes Verhalten kapseln, haben einen eigenen Kontrollthread und sind in der Lage mit anderen Agenten sozial zu interagieren.

AOSE baut dabei auf einer Kopplung von Interaktion und Kommunikation auf Wissensebene über eine Agentensprache auf.

Derzeit gibt es im Bereich von AOSE erst wenige Werkzeuge und deshalb ist eine breite industrielle Anwendung des Paradigmas noch nicht spruchreif. Eine Evaluierung von Werkzeugen und Theorien der AOSE findet sich in [ASS]. Allerdings haben sich mehrere Firmen und Universitäten zu unterschiedlichen Organisationen zusammen geschlossen und arbeiten an der Entwicklung von Standards. Sie versuchen Standards und Normen für den Entwurf und die Implementierung von interagierenden Agenten zu erarbeiten.

Um hier nur ein Beispiel für die Erweiterung von de facto Standards zu geben, sei Agent UML erwähnt. Unified Modelling Language (UML) ist eine Sprache, eine Notation für die Modellierung von objektorientierten Systemen. UML ist in der objektorientierten Analyse und im Design weit verbreitet und zum de facto Standard geworden ([OMGb]). Daraus haben sich Anstrengungen ergeben UML in Richtung der Agentensystemmodellierung zu erweitern. So wurden bereits Modifikationen vorgeschlagen [CRA01] und in [ODE00], [PAR01], [BAU01], [AUM] wird Agent UML (AUML) beschrieben und auf die darin enthaltenen Erweiterungen von UML eingegangen.

Die Erweiterung von UML wird unter anderen auch von Standardisierungsorganisationen, wie der Object Modelling Group (OMG) und der Foundation for Intelligent Physical Agents (FIPA, ([OBR98], [POS])) unterstützt.

Bevor auf Werkzeuge und Methoden eingegangen wird ein Zitat aus [JEN]:

„My guess is that object-oriented programming will be what structured programming was in the 1970s. Everybody will be in favour of it. Every manufacturer will promote his products as supporting it. Every manager will pay lip service to it. Every programmer will practice it (differently). And no one will know just what it is.” (Rentsch, 82)

„My guess is that agent-based computing will be what object-oriented programming was in the 1980s. Everybody will be in favour of it. Every manufacturer will promote his products as supporting it. Every manager will pay lip service to it. Every programmer will practice it (differently). And no one will know just what it is.” (Jennings, 99).

4.3. Werkzeuge und Methoden

AOSE basiert unter anderem auf wissensbasierten Agentenkommunikationssprachen und Protokollen. Standardisierungen in diesem Bereich sind besonders für Multi-Agenten Systeme wichtig, da diese keine isolierten Einheiten sind, die ihre Ziele allein erreichen, sondern mehrere Agenten müssen zusammenarbeiten und zusammenwirken können, um eigene und gemeinsame Ziele realisieren zu können.

Jeder Agent sollte dabei theoretisch in der Lage sein mit jedem anderen Agenten über ein beliebiges Gebiet, welches zur Designzeit des Agenten nicht bekannt ist, kommunizieren zu können. Dies kann nur dadurch gelöst werden, indem jeder Agent eine deklarative interne Wissensrepräsentation besitzt und die Kommunikation unter Einbeziehung gemeinsamer Ontologien (Ontologie der Informatik, Medizin, der Automobile, etc.) erfolgt.

XML (eXtensible Markup Language) gewinnt in diesem Bereich immer mehr an Bedeutung und Akzeptanz. In Zukunft könnte es das allgemeingültige Format für Nachrichten darstellen ([BIG01]). So bietet XML etwa Data Type Definition´s (DTDs) und Schemata, die für die Definition von Ontologien verwendet werden können (siehe Kapitel „4.3.3. XML-Familie“).

Wie bereits erwähnt gibt es mehrere Organisationen (Knowledge Sharing Effort – KSE, Foundation for Intelligent Physical Agents – FIPA, Agent Society, etc.), die versuchen Standards ([CHA00b]) zur Verfügung zu stellen. Diese stehen miteinander in Konkurrenz und die Zukunft wird weisen, welche Standards sich durchsetzen werden. Eine detaillierte Beschreibung der Organisationen und ihren Standards ist in [BOE99] zu finden. Im folgenden wird beispielhaft auf Knowledge Query and Manipulation Language (KQML) und Knowledge Interchange Format (KIF) als Repräsentanten für Agentensprachen eingegangen.

4.3.1. Agentenkommunikationssprachen (Agent Communication Languages – ACL)

Agentenkommunikationssprache ([REE02]) ist der Oberbegriff für Sprachen, in denen die Kommunikation von Agenten implementiert wird. Die Aufgabe ist es den Agenten die Möglichkeit zur Kommunikation, zur Zusammenarbeit und zum Wissensaustausch zu geben.

Eine Agentenkommunikationssprache sollte folgende Merkmale aufweisen:

- Sie sollte allgemein anwendbar, sprich unabhängig von der Semantik eines bestimmten Agenten sein.
- Sie sollte einfach verwendbar sein. Dafür eignet sich besonders ein Aufbau als Schichtenstruktur. D.h. es sollte eine Trennung geben, zwischen dem Teil der kommunikative Akte zwischen Agenten ausdrückt und dem Teil der den eigentlichen Inhalt der Kommunikation darstellt.
- Die Sprache sollte natürlich auch netzwerkfähig sein. Sie sollte dabei unabhängig von Transportmechanismen, wie TCP/IP, http, etc. sein.

Als Modell für die Kommunikation zwischen Agenten hat die Sprechakttheorie (speech act theory, [AUS62]) an Bedeutung gewonnen.

Die Philosophen John Austin und John R. Searle ([AUS62], [SEA69]) prägten in den 60er Jahren die Idee, dass Sprechende nicht nur Aussagen tätigen, sondern auch Aktionen durchführen bzw. auslösen.

Beispielsweise ist die Aussage „Kannst Du mir das Buch geben?“ nicht nur eine einfache Frage, sondern impliziert für den Gefragten die Auslösung einer Aktion. Der Gefragte gibt dem Fragenden das Buch oder reagiert mit einer entsprechenden Antwort (Gegenfrage), wie etwa „Welches Buch meinst Du?“

Ein Sprechakt besteht aus drei Komponenten:

- Lokutionärer Akt beschreibt das „Aussprechen“ einer Nachricht, hiermit ist rein der Übermittlungsvorgang gemeint – Syntax.
- Illokutionärer Akt beschreibt die, mit der Nachricht implizit verknüpften Absichten (Intentionen) des Senders (Bitte, Befehl, Frage, etc.) – Semantik.
- Perlokutionärer Akt beschreibt die, aus der Aussage resultierenden Auswirkungen auf die Umwelt, die Änderungen der Umwelt – Pragmatik.

Ein Sprechakt kann mehrere illokutionäre Akte umfassen, die Aussagen können mehrdeutig sein. Deshalb wird im Rahmen der Kommunikationssprachen eine Kategorisierung der Sprechakte vorgenommen – es werden Performative (performatives), wie report, insist, tell, request, etc. definiert, die eine explizite Bedeutung haben.

4.3.1.1. Knowledge Query and Manipulation Language – KQML

Knowledge Query and Manipulation Language (KQML) ist aus Knowledge Sharing Effort (KSE) hervorgegangen ([FIN94], [LAB97]). KSE wurde Anfang der 90er Jahre als Initiative von Firmen und Universitäten gegründet.

KQML ist eine Sprache die Nachrichtenformate definiert und ein Protokoll, für die Nachrichtenübermittlung zwischen Agenten. Dabei wird der Inhalt von Wissensbasen beschrieben. Es ist möglich Informationen zu erfragen, Informationen weiterzugeben und auch zwischen Agenten zu vermitteln. KQML basiert dabei auf der bereits erwähnten Sprechakttheorie.

KQML ist aus drei Schichten aufgebaut:

Abbildung 22: Schichtaufbau einer KQML-Nachricht

Die innerste Schicht – der „content layer“ - enthält die Nachricht selbst. Die Nachricht kann in einer beliebigen Sprache formuliert sein, die nur einigen syntaktischen Regeln folgen muss. Das Format der Nachricht ist für KQML also unerheblich, der Empfänger muss diese allerdings interpretieren können.

Die Mittelschicht – der „message layer“ - definiert verschiedene Kategorien von Nachrichten (siehe Tabelle 3: Kategorien von KQML-Nachrichten).

Kategorie	Nachrichtentyp/Kommunikationsperformativ
grundlegende Anfragen	evaluate, ask-if, ask-about, ask-one, ask-all
Anfragen mit multiplen Antworten	stream-about, stream-all, eos
Antworten	reply, sorry
Multiple Antworten	standby, ready, next, rest, discard
Informationen	tell, achieve, cancel, untell, unachieved
Fähigkeiten	advertise, subscribe, monitor, import, export
Netzwerke	register, unregister, forward, broadcast, route

Tabelle 3: Kategorien von KQML-Nachrichten

Durch die Kategorie wird mitgeteilt, um welche Art von Nachricht (Anfrage, Antwort, Fehlermeldung, etc.) es sich handelt – der Sprechakttyp wird festgelegt. Dabei können auch zusätzliche Informationen für die Weiterleitung des Nachrichtenpakets gegeben werden, sodass der Inhalt für die Übermittlung nicht preisgegeben werden muss.

Die oberste Schicht – der „communication layer“ - fügt Angaben, wie z.B. die eindeutige Kennung des Senders und des Empfängers, eine Nachrichten-ID, etc. hinzu.

Die Syntax der Nachrichten ist dabei an die Programmiersprache Common Lisp angelehnt. Der Aufbau einer Nachricht hat dabei folgende Struktur:

```
(<performative>
      :content <statement/speechact>
      :sender <name>
      :receiver <name>
      :language <text>
      :ontology <text>
      ...
)
```

Die Menge der KQML-Performatives ist beschränkt und genau festgelegt. :content enthält die Nachricht. :sender bezeichnet den Absender der Nachricht und :receiver gibt den Empfänger der Nachricht wieder. :language bezeichnet die Sprache, in der der Inhalt der KQML-Nachricht geschrieben ist. Mögliche Sprachen sind etwa KIF (Knowledge Interchange Format), Prolog, SQL, etc. :ontology beschreibt schlussendlich die Ontologie (das Vokabular), die dem Inhalt der Nachricht zugrunde liegt.

Ein konkretes Beispiel einer KQML Nachricht:

```
(ask-one
     :content (float price = sun.price())
     :sender sue
     :receiver stock-server
     :reply-with sun-stock
     :language java
     :ontology NYSE-TICKS
)
```

Der KQML Performativ hier ist „ask-one“. Agent „sue“ fragt dabei Agent „stock-server“ nach dem Kurs der SUN-Aktie.

Eine mögliche Antwort könnte in Folge so aussehen:

```
(tell
     :content (float price = 41)
     :sender stock-server
     :receiver sue
     :in-reply-to sun-stock
     :language java
     :ontology NYSE-TICKS
)
```

Auf KQML baut auch die Agentenkommunikationssprache (ACL) des FIPA-Standards auf. Dafür wurde KQML entsprechend erweitert und modifiziert.

4.3.1.2. Knowledge Interchange Format – KIF

Knowledge Interchange Format wurde ebenso wie KQML von KSE entwickelt. KIF ist, wie bereits erwähnt eine mögliche Contentsprache in einer KQML-Nachricht. KIF wurde für den Wissensaustausch zwischen Agenten entwickelt [GEN92] und basiert auf einer Präfix-Version der Prädikatenlogik (ähnlich wie LISP). KIF unterstützt die Definition von Objekten, Funktionen, Beziehungen, Regeln und von Metawissen (Wissen über Wissen).

Beispiele:

```
(farbe sonne gelb)
```

Beschreibt, dass die Farbe der Sonne mit gelb festgelegt ist.

```
(GetStockPrice ?price) (IntegerCompare ?price ">" 150)
```

Diese Regel besagt, dass der Kurs einer Aktie bestimmt werden soll und der Wert in der Variable price gespeichert wird. Anschließend wird geprüft, ob der Kurs über 150 liegt.

```
(> (* (width chip1) (length chip1))
   (* (width chip2) (length chip2)))
```

In dieser Funktion wird abgefragt, ob Chip1 größer als Chip2 ist.

Aus Effizienzgründen hat man sich zu einer Schichtenarchitektur entschieden. Grundsätzlich wäre es möglich die gesamte Kommunikation nur in KIF zu verfassen. KIF ist allerdings kontextunabhängig und so müsste jede Botschaft alle Angaben über Sender, Empfänger, etc. beinhalten. Effizienter ist es, den Inhalt und Sendeinformationen zu trennen und somit dient KIF als Inhalt und KQML stellt den Rahmen darum dar.

Beispiel einer KQML-Nachricht mit einem in KIF formuliertem Inhalt:

```
(ask-if
     :content (> (* (width chip1) (length chip1))
                 (* (width chip2) (length chip2)))
     :language KIF
     :ontology Chips
)
```

```
(reply
      :language KIF
      :ontology Chips
      :in-reply-to Agent1
      :content (true)
)
```

Agent1 fragt an, ob Chip1 größer als Chip2 ist und Agent2 antwortet darauf, dass dies zutrifft.

4.3.2. Sprachen zur Implementierung von Agenten

Generell können Agenten in jeder beliebigen Programmiersprache implementiert werden. Für viele herkömmliche Programmiersprachen gibt es auch Erweiterungen in Richtung Agentenentwicklung. Darüber hinaus sind jedoch auch Programmiersprachen entwickelt worden, mit denen spezielle Aspekte von Agenten einfacher umgesetzt werden können.

Programmiersprachen, die für die Implementierung von Agenten eingesetzt werden sollten jedoch bestimmten Anforderungen genügen:

Objektorientierung

Es wurde bereits erwähnt, dass Agenten Ähnlichkeiten mit Objekten aufweisen. Ein Agent besitzt ebenfalls Datenelemente und Methoden, wobei er eine Teilmenge seiner Methoden der Öffentlichkeit zur Verfügung stellen kann. Ein direkter Zugriff auf die internen Datenstrukturen eines Agenten soll nicht möglich sein.

Plattformunabhängigkeit

Da Agenten, die Möglichkeit haben sollen in unterschiedlichen Umgebungen zu operieren, soll die für die Implementierung verwendete Programmiersprache plattformunabhängig sein. Dabei soll sie die Portierbarkeit und die Migration von mobilen Agenten unterstützen.

Kommunikationsfähigkeit

Besonders Multiagentensysteme stellen die Forderung der Kommunikationsfähigkeit einer Programmiersprache. Die Programmiersprache muss die Implementierung von kommunikationsorientierten Komponenten unterstützen, sprich von Komponenten, die die

Kommunikation zwischen mehreren Agenten und die Kommunikation und Interaktion der Agenten mit ihrer Umwelt ermöglichen.

Sicherheit

Besonders mobile Agenten benötigen einen hohen Sicherheitsstandard ([HOH99]). Die verwendete Programmiersprache muss daher Sicherheitsfunktionen (z.B. Verschlüsselung, etc.) zur Verfügung stellen oder die Möglichkeit bieten diese bei Bedarf zu integrieren.

Code-Manipulation

Oftmals ist es notwenig Programmcode von Agenten zur Laufzeit zu manipulieren. Um dies zu ermöglichen muss die Programmiersprache Mechanismen zur Identifizierung von Agenten unterstützen – Code muss gegebenenfalls überprüft werden können. Auch die Übertragung (Senden, Empfangen, Ausführen) von Code muss möglich sein.

Darüber hinaus gibt es je nach Einsatzgebiet und gewünschten Agententypen zusätzliche wünschenswerte Merkmale. Darunter sind etwa Multitaskingfähigkeit, persistente Datenhaltung, Erweiterbarkeit, Reaktivität, etc. zu nennen.

Im folgenden wird beispielhaft auf vier Programmiersprachen und ihre Eignung für die Implementierung von Agenten eingegangen.

4.3.2.1. JAVA

Java ist eine objektorientierte Programmiersprache, die von Sun Microsystems ([JAV]) entwickelt und 1995 offiziell kommerziell vorgestellt wurde. Java verfügt über ein hierarchisches Klassenkonzept mit einfacher Vererbung. Einfache Datentypen, wie int, float, etc. sind jedoch nicht als Objekte implementiert. Java hat gegenüber von C++ eine automatische Garbage Collection und im Sprachkonzept sind keine expliziten Pointer vorgesehen. Dadurch sind mögliche Fehlerquellen und Missbrauch erschwert. Java unterstützt TCP/IP- und Socketkommunikation sowie Threads zur nebenläufigen Programmierung. Java-Programme sind in hohem Maße robust und portabel. Java wurde so plattformunabhängig, wie möglich gestaltet. Java-Sourcecode wird kompiliert in Java-Bytecode und dieser Bytecode wird in der Virtual Machine, einem Interpreter ausgeführt. Diese Virtual Machine kann somit auf nahezu allen Plattformen laufen. Auch wird die Übertragung von Programmen

in Form von Applets in gesicherten Umgebungen unterstützt. Durch den eingebauten Objekt-serialisierungsmechanismus (Speichern und Wiederherstellen von Objekten und ihren Beziehungen) wird die Implementierung von mobilen Agenten unterstützt.

Zur Zeit scheint es, dass sich Java unter der Vielzahl der zur Verfügung stehenden interpretierten Programmiersprachen als Programmiersprache für mobile Agenten etabliert hat, obwohl noch einige Konzepte für Agenten fehlen, wie beispielsweise das Auslösen einer Migration oder eine Einschränkung der Objektzugriffe auf nur lesenden Zugriff, etc.

Trotz dieser Nachteile ist durch die stetig fortschreitende Verbreitung von Java davon auszugehen, dass sich Java–basierte Agenten und Agentensysteme durchsetzen werden, da die Vorteile im Moment überwiegen.

Bei Java handelt es sich um eine weit verbreitete Programmiersprache, die objektorientiert ist, somit Wiederverwendbarkeit unterstützt. Sie ist robust, portabel und praktisch plattform-unabhängig. Sie erlaubt Multithreading, Parallelisierung und bietet Sicherheitsmechanismen (Sandbox), drei besonders wichtige Konzepte für die Umsetzung von Agenten. Im Bereich der Implementierung von Agenten mittels Java wird auch intensiv geforscht und der Sprach- bzw. Klassenumfang von Java wird fortwährend erweitert. So gibt es bereits eine Implementierung für die Integration von KQML ([MAS]) und von Sun ([SUNb]) wurde auch bereits XML eingebunden.

4.3.2.2. C#

C# ist eine modere, objektorientierte, plattformunabhängige Programmiersprache. C# wurde als Bestandteil des .NET Frameworks [LIB01] als Programmiersprache ([ROB02], [ARC01]) von Microsoft entwickelt. Bei C# kann man Einflüsse aus den Programmiersprachen C++, Java und auch Visual Basic erkennen. Von der Syntax und von der Mächtigkeit her ist sie sehr stark von C++ geprägt, wobei aber mögliche Fehlerquellen, wie das Pointerkonzept bzw. die Mehrfachvererbung nicht mehr im C#-Sprachumfang vorkommen. Hingegen ist C#, wie auch JAVA, plattformunabhängig konzipiert und unterstützt die strukturierte, komponenten-orientierte (Properties, Events, Attribute, etc.) und objektorientierte Programmierung (Klassenkonzept, Vererbung, Polymorphismus, Interfaces, etc.) gleichermaßen. Auch Inline-Dokumentation und Ausnahmebehandlungen sind in C# berücksichtigt.

4.3.2.3. Telescript, Odyssey

Telescript ([WHI94a], [WHI94b]) wurde von General Magic mit der Unterstützung verschiedener Firmen, wie Apple, Fujitsu, Motorola, Sony, etc., speziell für die Kommunikation innerhalb eines Netzwerks entwickelt. Telescript ist eine Architektur für mobile Agenten. Wie in Java werden Telescript-Programme interpretiert. Dafür gibt es einen Interpreter, die Telescript Engine, die als Ablaufumgebung dient, die Migration der Agenten steuert und etwa Kommunikationsdienste anbietet. Somit wird eine Portabilität der Programme erreicht. Telescript ist objektorientiert ([GEN95]), auch Typen wie Integer und String sind als Klassen implementiert. Die Syntax ist ähnlich zu C++ und Java.

In Telescript wird zwischen Orten (places) und Agenten (agents) unterschieden, die durch einen eindeutigen Namen (telename) identifiziert werden können. Ein Ort ist stationär, er verbleibt immer innerhalb einer Telescript Engine. Innerhalb eines Ortes können weitere Orte enthalten sein. Die Orte bieten in der Regel Dienste (Methoden) an die von Agenten in Anspruch genommen werden können. Agenten sind mobil, sie können von Ort zu Ort wandern mittels einer Methode go().

Telescript stellt verschiedene Sicherheitsmechanismen zur Verfügung:

- Agenten und Orten können bestimmte Rechte zugewiesen werden, wie das Recht bestimmte Befehle auszuführen oder das Recht eine bestimmte Ressourcenmenge zu verbrauchen.

- Über den telename können Agenten und Orte eindeutig identifiziert und authentifiziert werden und Orte können Agenten den Zutritt verweigern.

- Orte und Agenten können ausschließlich über die von der Telescript Engine zur Verfügung gestellten Schnittstelle auf Ressourcen des Systems zugreifen.

Das Telescript-System konnte sich nicht durchsetzen und wurde daher von General Magic vom Markt genommen und durch Odyssey (1997) ersetzt. Dabei wurden die vorhandenen Konzepte in Java implementiert. Odyssey unterstützt die Nutzung von beliebigen Transport-systemen zur Migration ([BURb]). Standardmäßig wird dabei RMI (Remote Method Invocation von Sun) verwendet, aber auch Module für DCOM (Distributed Object Model von Microsoft) oder IIOP (Internet Inter-ORB Protocol von CORBA) sind vorhanden.

4.3.2.4. Tcl, Safe Tcl, Agent Tcl, D´Agents

1987 hat John Ousterhout ([TCLa]) mit der Entwicklung von Tcl (Tool Command Language) begonnen. Tcl ist prozedural und nicht objektorientiert und hat nur einen Datentyp, die Zeichenkette (String). Die Syntax von Tcl lehnt sich sehr stark an die Unix-Shell und an LISP an. Die Sprache wird interpretiert und alle Ressourcenzugriffe laufen kontrolliert über einen Parser ab. Tcl ist für einfache kleine Anwendungen sehr populär, für komplexere Anwendungen eignet sich diese Skriptsprache jedoch nicht.

Um mehr Sicherheit zu gewährleisten wurde 1992 mit der Entwicklung von Safe Tcl ([TCLb]) begonnen. Dabei wurden alle Tcl Befehle aus dem Sprachumfang entfernt, die ein potentielles Sicherheitsrisiko darstellten. Gleichzeitig wurden Befehle zur Agentenkommunikation hinzugefügt.

Agent Tcl ([GRA96]) ist ein Agentensystem für mobile Agenten, dass in Tcl geschrieben ist. Agent Tcl besteht aus zwei Komponenten: einem modifizierten Tcl-Interpreter, der die Tcl Agenten ausführt und einem Server, der auf jedem Rechner läuft zu dem Agenten migrieren können. Wichtigstes Ziel bei Agent Tcl ist eine einfache und unabhängige Migration von Agenten und eine flexible transparente Agentenkommunikation zu unterstützen. Darüber hinaus soll die Agentenprogrammierung nicht nur in Tcl erfolgen, sondern komplexere Agenten sollen z.B. in Java implementiert werden können. Letztlich sollen auch Sicherheitskonzepte aus dem Safe Tcl-Ansatz umgesetzt werden.

Mittlerweile ist man von Agent Tcl nach D´Agents ([GRA98]) übergegangen, an dem ebenfalls am Dartmouth College geforscht wird. Ziel von D´Agents ist dabei Applikationen zu unterstützen, die in Netzwerken mit verteilter Information arbeiten, sprich diese gewinnen, organisieren und präsentieren müssen. Dabei wird intensiv geforscht in den Bereichen:

- Sicherheitsmechanismen,
- Unterstützung von mobilen und nur zeitweise miteinander in Verbindung stehenden Rechnern,
- Navigation, Netzwerkabtastung, Werkzeuge für die Suche nach Ressourcen und
- Automatische Indexierung, Gewinnungs- und Clustertechniken für Dokumente.

D´Agents wird derzeit in einigen Informationsgewinnungs- und Workflowapplikationen eingesetzt ([DAG]).

4.3.3. XML-Familie

XML (eXtensible Markup Language) ist das Universalformat für strukturierte Dokumente und Daten im Web ([W3Ca], [HOL98]). XML ist eine Teilmenge von SGML (Standard Generalized Markup Language) und wurde 1996 von einer Arbeitsgruppe entwickelt, die unter der Schirmherrschaft des World Wide Web Consortium (W3C) gegründet wurde. XML wurde entworfen, um eine einfache Implementierung und Zusammenarbeit sowohl mit SGML als auch mit HTML zu gewährleisten ([XML]).

HTML versucht, mit einer einzigen Definition die Struktur (Aufbau, Inhalt und Aussehen) von Dokumenten zu beschreiben. XML hingegen bietet zwar die Möglichkeit dies über ein Dokument zu lösen, wesentlich ist aber, dass es Möglichkeiten zur Verfügung stellt den Aufbau, den Inhalt und das Aussehen getrennt voneinander zu beschreiben. Dies wirkt sich positiv auf den Entwurf, die Wiederverwendbarkeit, die Robustheit, etc. aus.

Die Entwurfsziele für XML waren [W3Cb] daher:

- XML soll sich im Internet auf einfache Weise nutzen lassen.
- XML soll ein breites Spektrum von Anwendungen unterstützen.
- XML soll zu SGML kompatibel sein.
- Es soll einfach sein, Programme zu schreiben, die XML-Dokumente verarbeiten.
- Die Zahl optionaler Merkmale in XML soll minimal sein, idealerweise Null.
- XML-Dokumente sollten für Menschen lesbar und angemessen verständlich sein.
- Der XML-Entwurf sollte zügig abgefasst sein.
- Der Entwurf von XML soll formal und präzise sein.
- XML-Dokumente sollen leicht zu erstellen sein.
- Knappheit von XML-Markup ist von minimaler Bedeutung.

Ein Datenobjekt ist ein XML-Dokument, wenn es im Sinne dieser Spezifikation wohlgeformt (well-formed) ist, sprich entsprechend der XML-Syntax aufgebaut ist. Ein wohlgeformtes XML-Dokument kann darüber hinaus gültig (valid) sein, sofern es bestimmten weiteren Einschränkungen genügt, sprich einem gegebenen Schema folgt ([GOL98]).

Das „klassische“ Konstrukt für solche Schemata sind Dokumenttyp-Definitionen (DTD), die bereits in SGML bekannt sind. Anhand der Spezifikation einer DTD kann der Inhalt von XML-Dokumenten validiert werden. Durch DTDs wird die gewünschte Struktur eines XML-Dokuments beschrieben. Es handelt sich dabei um Meta-Informationen über die erlaubte Reihenfolge und Verschachtelungen der Elemente, über mögliche oder nötige Attribute, über Typen und Standardwerte ([STU00], [BOX00]; [EDE01]). Typdeklarationen gibt es in DTDs nur im eingeschränkten Maße und beschränken sich im wesentlichen auf nahezu beliebige Zeichenketten (CDATA) und auf die Definition von Attributen als ID (eindeutiger Attributwert) oder IDREF (Referenz auf einen eindeutigen Attributwert).

Im folgenden sei ein Beispiel für Mitgliedsdaten angeführt. Diese haben die folgende Struktur:

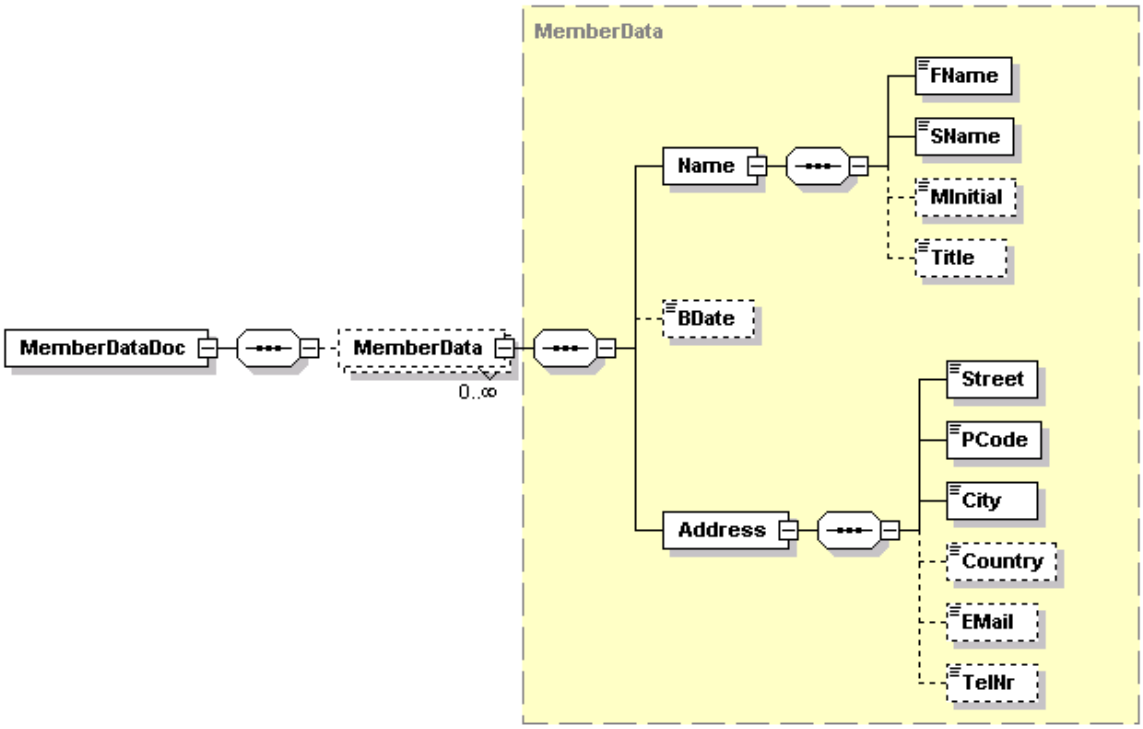


Abbildung 23: Strukturierter Aufbau von Mitgliedsdaten

Ein Element namens MemberDataDoc soll keine bis hin zu beliebig vielen MemberData-Elementen beinhalten können. Ein Element namens MemberData besteht aus den drei Elementen Name, BDate und Address, wobei das Element Name wieder unterteilt ist in FName (First Name), SName (Surname) und optional in MInitial (Middle Initial) und Title. Das Element Address besteht notwendigerweise aus Street, PCode, City und Country. Email und TelNr (telephone number) können noch optional zusätzlich angeführt werden. Darüber hinaus soll MemberData zwei Attribute haben: eine ID (eindeutige Kennzeichnung) und ein Attribut Gender, welches wahlweise mit Female oder Male belegt werden kann.

Als DTD umgesetzt hat dies folgendes Aussehen:

```
<?xml version="1.0" encoding="UTF-8"?>
<!--DTD generated by Reisinger Susanne)-->
<!ELEMENT MemberDataDoc (MemberData*)>
<!ELEMENT MemberData (Name, BDate?, Address)>
<!ATTLIST MemberData
      id CDATA #REQUIRED
      Gender (Female | Male) #REQUIRED
>
<!ELEMENT Name (FName, SName, MInitial?, Title?)>
<!ELEMENT BDate (#PCDATA)>
<!ELEMENT Address (Street, PCode, City, Country?, EMail?, TelNr?)>
<!ELEMENT FName (#PCDATA)>
<!ELEMENT SName (#PCDATA)>
<!ELEMENT MInitial (#PCDATA)>
<!ELEMENT Title (#PCDATA)>
<!ELEMENT Street (#PCDATA)>
<!ELEMENT PCode (#PCDATA)>
<!ELEMENT City (#PCDATA)>
<!ELEMENT Country (#PCDATA)>
<!ELEMENT EMail (#PCDATA)>
<!ELEMENT TelNr (#PCDATA)>
```

Innerhalb eines MemberDataDoc können mehrere MemberData angeführt sein, sprich innerhalb einer Datei können die Daten mehrerer Mitglieder angeführt werden. Dies wird durch den Stern im Ausdruck `<!ELEMENT MemberDataDoc (MemberData*)>` gekennzeichnet. Optionale Elemente werden durch Fragezeichen gekennzeichnet, wie in `<!ELEMENT Name (FName, SName, MInitial?, Title?)>`. MInitial und Title können daher angegeben werden, müssen jedoch nicht innerhalb eines XML-Dokuments aufgelistet sein. Die einzelnen Elemente sind alle vom Typ #PCDATA (`<!ELEMENT FName (#PCDATA)>`). Dies besagt, dass der Inhalt aus Zeichenfolgen bestehen darf. Die Attribute von MemberData wurden beide als #REQUIRED angegeben, d.h., dass diese unbedingt angegeben werden müssen. Dabei kann die aus einer beliebigen Zeichenfolge bestehen (als CDATA deklariert) und bei Gender kann zwischen Female und Male gewählt werden. Darüber hinaus sind keine anderen Werte möglich.

Ein gültiges XML-Dokument, sprich ein Dokument, dass sich an die XML-Syntax hält und dieser DTD entspricht, könnte folgendermaßen aussehen:

```
<MemberDataDoc>
 <MemberData MemberID="M1" Gender="Female">
        <Name>
            <FName>Reisinger</FName>
            <SName>Susanne</SName>
            <Title>Dipl.-Ing.</Title>
        </Name>
        <BDate>19.04.1974</BDate>
        <Address>
            <Street>Altenbergerstrasse 69</Street>
            <PCode>4040</PCode>
            <City>Linz</City>
            <Country>Austria</Country>
            <EMail>sreisinger@fim.uni-linz.ac.at</EMail>
            <TelNr>+43(70)2468-8437</TelNr>
        </Address>
 </MemberData>
</MemberDataDoc>
```

Hier werden die Daten eines Mitglieds wiedergegeben. Dabei wurden die Daten in der Reihenfolge angeführt, wie sie durch die DTD festgelegt sind. Alle Elemente von Address wurden ausgefüllt, bei Name wurde für MInitial keine Angabe gemacht und daher ausgelassen.

DTDs haben allerdings auch einige Nachteile. Mittels des XML Metadata Interchange Format (XMI) können aus objektorientierten Datenmodellen, beschrieben etwa durch die Unified Modelling Language (UML), oder auch direkt aus z.B. einer Javaklasse, passende XML Strukturen automatisch generiert werden, doch lassen DTDs weder die Beschreibung bestimmter semantischer Bedingungen noch die Festlegung von Wertebereichen zu [SCH].

Auch sind die Datentypen auf Zeichenketten und explizite Aufzählung von Werten beschränkt. Es gibt keine Möglichkeit numerische Datentypen oder eigene Datentypen zu definieren. Gerade im Zusammenhang mit Datenbankabfragen im Web ist dies problematisch ([CHA00a]).

Auch sind DTDs hinsichtlich begrenzt, dass sie keine Namensräume (namespaces) ([AND00], [HOL01]) unterstützen. Basiert daher ein XML Dokument auf mehreren DTDs so kann dies hinsichtlich der Benennung der Bezeichner problematisch sein, da das Auftreten von Namenskollisionen nicht ausgeschlossen werden kann.

Last, but not least ist die Erweiterbarkeit eingeschränkt. Auch gibt es keine Möglichkeit im Sinne der Objektorientierung eine Vererbung durchzuführen.

Diese Nachteile sollen durch den Nachfolger der DTDs, dem XML-Schema (XML Schema Definition Language, XSDL) bereinigt werden. Genau wie eine DTD stellt ein XML-Schema Regeln zur Verfügung, nach denen ein XML Dokument auf seine Gültigkeit überprüft wird. Wichtige Neuerungen von XML-Schema:

Wenn ein XML-Dokument gegen ein XML-Schema geprüft wird, kann dieses wohlgeformt sein oder gültig. Der Test auf Gültigkeit beinhaltet neben der Überprüfung auf korrekte Reihenfolge und Struktur der Elemente auch die Überprüfung, ob es sich um gültige Datentypen handelt und ob die, im XML-Schema definierten, Wertebereiche eingehalten wurden.

XML-Schema birgt folgende Vorteile in sich:

- Es umfasst eine Reihe von vordefinierten, einfachen Datentypen und bietet darüber hinaus die Möglichkeit explizit komplexe Elementtypen zu definieren.
- Wertebereiche können explizit angegeben werden.
- XML-Schema erlaubt Vererbung.
- XML-Schema unterstützt Namensräume. Dadurch werden Namenskollisionen vermieden. Elemente mit gleichen Namen, aber unterschiedlicher Struktur können dadurch getrennt und gemeinsam in einem XML-Dokument verwendet werden.

Im folgenden wird das Beispiel für Mitgliederdaten als XML-Schema wiedergegeben:

```
<?xml version="1.0" encoding="UTF-8"?>
<!--XML Schema generated by Reisinger Susanne)-->
<xsd:schema targetNamespace="http://www.fim.uni-linz.ac.at" xmlns:xsd="http://www.w3.org/2000/10/XMLSchema"
xmlns="http://www.fim.uni-linz.ac.at" elementFormDefault="qualified">
        <xsd:element name="MemberDataDoc">
            <xsd:complexType>
                <xsd:sequence>
                    <xsd:element name="MemberData" type="MemberData" minOccurs="0" maxOccurs="unbounded"/>
                </xsd:sequence>
            </xsd:complexType>
        </xsd:element>
<! – Complex Type MemberData -->
        <xsd:complexType name="MemberData">
            <xsd:sequence>
<! – Element Name -->
                <xsd:element name="Name">
                    <xsd:complexType>
                        <xsd:sequence>
                            <xsd:element name="FName" type="xsd:string"/>
                            <xsd:element name="SName" type="xsd:string"/>
                            <xsd:element name="MInitial" type="xsd:string" minOccurs="0"/>
                            <xsd:element name="Title" type="xsd:string" minOccurs="0"/>
                        </xsd:sequence>
                    </xsd:complexType>
                </xsd:element>
<! – Element Birth Date  -->
                <xsd:element name="BDate" type="xsd:date" minOccurs="0"/>
```

```
<! – Element Address  -->
            <xsd:element name="Address">
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element name="Street" type="xsd:string"/>
                        <xsd:element name="PCode" type="xsd:integer"/>
                        <xsd:element name="City" type="xsd:string"/>
                        <xsd:element name="Country" type="xsd:string" minOccurs="0"/>
                        <xsd:element name="EMail" type="xsd:string" minOccurs="0"/>
                        <xsd:element name="TelNr" type="xsd:string" minOccurs="0"/>
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:element>
        </xsd:sequence>
        <xsd:attribute name="id" use="required"/>
        <xsd:attribute name="Gender" use="required">
            <xsd:simpleType>
                <xsd:restriction base="xsd:string">
                    <xsd:enumeration value="Female"/>
                    <xsd:enumeration value="Male"/>
                </xsd:restriction>
            </xsd:simpleType>
        </xsd:attribute>
    </xsd:complexType>
</xsd:schema>
```

Das Beispiel sieht auf den ersten Blick viel komplexer aus, als das vorangegangene DTD. Das liegt vor allem aber daran, dass jetzt zusätzliche Informationen, wie ein Namensraum, Datentypen und Wertebereiche definiert sind. Beispielsweise ist nun das Element Street vom Datentyp string (`<xsd:element name="Street" type="xsd:string"/>`) PCode vom Typ integer (`<xsd:element name="PCode" type="xsd:integer"/>`) und BDate vom Typ date (`<xsd:element name="BDate" type="xsd:date" minOccurs="0"/>`), wobei zusätzlich angegeben wurde, dass BDate auch ausgelassen werden kann, also optional ist.

Ein Vorteil des XML-Schemas ist nun, dass hierfür einfacher eine äquivalente Datenbank aufgebaut werden kann und keine Probleme bezüglich der unterschiedlichen Datentypen zu erwarten sind, da diese im XML-Schema bereits klar und eindeutig spezifiziert sind. Somit können mehrere Agenten basierend auf dem XML-Schema Anfragen stellen und sowohl der Datenbank-Agent als auch die Anfrageagenten sind kompatibel zueinander.

Ein weiterer wesentlicher Vorteil wird sichtbar, wenn man MemberData als Klasse betrachtet, die als Basis für unterschiedliche Bereiche dient. Hier kommt die Vererbung zum Tragen. MemberData kann als Basisklasse dienen und dementsprechend für spezielle Ausprägungen erweitert werden. Etwa für Vereine um Elemente für den Mitgliedsbeitrag, für Portale um Elemente für einen Nickname und Passwort, usw.

Nunmehr haben wir den Aufbau (die Struktur) mittels DTDs bzw. XML-Schema beschrieben und den Inhalt in ein XML-Dokument verpackt, dass sich an die vorgegebene Struktur hält.

Abschließend fehlt uns noch die Möglichkeit den Inhalt anschaulich zu präsentieren. Hierfür sind in XML mehrere Möglichkeiten vorgesehen.

Zunächst wird empfohlen die XML-Daten zu transformieren. XML-Daten sollten aus mehreren Gründen umgewandelt werden ([POT]):

Unterschiedliche Dokumenttypen für die gleiche Art von Daten

Zum einen kann es vorkommen, dass alte Datenbestände umgewandelt werden müssen, oder man XML-Daten bekommt oder weitergibt und sich die Datentypen des Senders und des Empfängers unterscheiden. Z.B. wird einmal das Geschlecht einer Person als Attribut definiert und einmal als eigenes Element.

Dynamische Dokumente mit unterschiedlichen Sichten auf die Daten

Je nach Anwendungsgebiet ist es nötig Einträge in bestimmten Reihenfolgen zu sortieren oder nur einzelne Ausschnitte des Datenvolumens zu präsentieren. Dies betrifft vor allem auch vertrauliche Daten. So sollte ein Mitglied einen Überblick über seine Daten, wie seine Anschrift, seinen Mitgliedsbeitrag oder über die Veranstaltungen bei denen es angemeldet ist, abfragen können. Jedoch sollte ein einfaches Mitglied keine Informationen über die Daten und über offene Mitgliedsbeiträge anderer Mitglieder bekommen.

Transformation der Daten in ein Präsentationsformat

Vor allem aber sollen die Daten für die Präsentation aufbereitet werden können und dafür stehen ebenfalls verschiedene Formate, je nach Zielgruppe oder Zielsystem zur Verfügung. Denkbar wäre etwa HTML zur Darstellung in Web-Browsern oder WML (Wireless Markup Language) zur Darstellung am Handy, VoiceXML, XSL-FO (XSL Formatting Objects) etc.

Als Formatierungssprache steht hierfür XSL (Extensible Stylesheet Language) zur Verfügung. Diese besteht aus zwei wichtigen Komponenten:

- Einer Komponente zur Formatierung von XML-Daten, wie XSL-FO. Ein XSL-FO Dokument beschreibt wo, welche Inhalte in welcher Gestaltungsform erscheinen sollen (etwa PDF, RTF, etc.).

- Einer Komponente zur Transformation von XML-Daten. Hierfür hat sich XSLT (Extensible Stylesheet Language Transformations) als Bestandteil von XSL etabliert.

XSL wurde ursprünglich zur Formatierung von XML-Daten geschaffen. XSL stellt daher eine Vielzahl von Formatierungsmöglichkeiten zur Verfügung. XSLT übernimmt die Transformation der Daten. Ein XSLT Dokument beschreibt daher, wie ein XML Quell-dokument in ein Ergebnisdokument zu überführen ist. XSLT bedient sich dabei wiederum der Sprache XPath.

Die XML Path Language (XPath) ist eine Sprache für die Lokalisierung von Elementen in der Baumstruktur eines XML-Dokuments. XPath selektiert die benötigten Daten aus dem XML Quelldokument, die für das Ergebnisdokument benötigt werden. Nur relevante Daten werden somit für die Darstellung für den Benutzer oder den Agenten bereitgestellt.

Die zentrale Rolle für den Transformationsprozess spielen Template-Regeln, die in einem XSL-Dokument enthalten sind. Eine Template-Regel beinhaltet Pattern (Muster) die spezifizieren, welche Ausgangsinhalte gegen neue Inhalte ausgetauscht werden (altes Vokabular wird auf das neue abgebildet). Dabei scannt der XSL-Prozessor die gesamte Struktur eines XML-Quelldokuments, vergleicht sie mit den Mustern der Template-Regeln, ersetzt die Inhalte und gibt anschließend das neu erzeugte Dokument aus.

Im folgenden sollen zwei Beispiele zum besseren Verständnis gegeben werden. Beim ersten Beispiel wird eine Transformation von einem Vokabular in ein anderes Vokabular vorgenommen. Beim Zweiten werden die XML-Daten in eine HTML-Präsentation transformiert.

Beiden Beispielen liegt die gleiche XML-Quelldatei und die gleiche Strukturbeschreibung zugrunde:

XML-Quelldatei:

```
<?xml version="1.0" encoding="UTF-8"?>
<!--XML Quelldatei generated by Reisinger Susanne)-->
<!DOCTYPE member SYSTEM "member.dtd">
<memberDoc>
  <member gender="female">
      <name>Susanne</name>
      <address>Altenbergerstrasse 69, Linz</address>
  </member>
  <member gender="male">
      <name>Reinhard</name>
      <address>Freistaedterstrasse 315, Linz</address>
  </member>
</memberDoc>
```

XML-Strukturbeschreibung (DTD):

```
<?xml version="1.0" encoding="UTF-8"?>
<!--XML DTD generated by Reisinger Susanne)-->
<!ELEMENT memberDoc (member)*>
<!ELEMENT member (name, address)>
<!ATTLIST member
      gender (female | male) #REQUIRED
>
<!ELEMENT name (#PCDATA)>
<!ELEMENT address (#PCDATA)>
```

Hier wird die Struktur der Daten (Geschlecht, Name, Adresse) von Mitgliedern beschrieben. In einem XML-Dokument können dabei mehrere Mitgliederdaten gespeichert sein.

Beispiel 1: Transformation des Vokabulars

Hier wurde eine XSLT-Datei erstellt, die das Attribut gender in ein Element abbildet. Das Element name wird übernommen und das Element address wird nicht selektiert.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- XSLT generated by Reisinger Susanne)-->
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
 <xsl:output method="html" version="1.0" encoding="UTF-8" indent="yes"/>
 <xsl:template match = "member">
      <member>
           <gender><xsl:value-of select="@gender" /></gender>
           <xsl:apply-templates/>
      </member>
 </xsl:template>
 <xsl:template match = "name">
      <nickname><xsl:apply-templates /></nickname>
 </xsl:template>
 <xsl:template match = "address"></xsl:template>
</xsl:stylesheet>
```

xsl:template ist eine Elementvorlage. Das Attribut match gibt jeweils an, für welchen Knoten des Dokuments die Vorlage gilt. xsl:value-of liefert den Wert des Ausdrucks im select-Attribut (hier gender). xsl:apply-templates veranlasst den XSL-Prozessor die Verarbeitung mit den Kindknoten fortzusetzen und für diese nach Vorlagen zu suchen.

Eine Ausgabe der Quelldatei nach der Transformation würde folgendermassen aussehen:

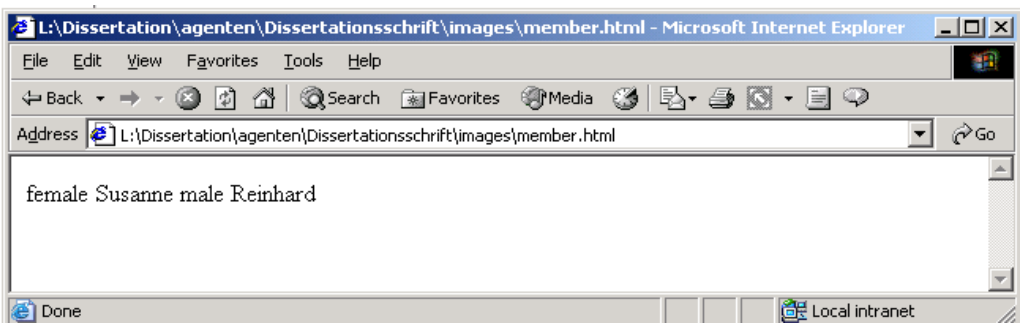


Abbildung 24: Anzeige der erzeugten HTML-Ausgabe der Inhalte (Version 1)

Für den Benutzer ist nicht ersichtlich, dass eine Transformation des Vokabulars stattgefunden hat und er bekommt nur die für ihn relevanten Informationen, wie Geschlecht und Name. Die Adresse jedoch bleibt für ihn verborgen.

Beispiel 2: Transformation in eine HTML-Präsentation

Ausgehend von derselben Strukturbeschreibung und desselben XML-Quelldokuments soll nun eine Präsentation der Datensätze in Form von HTML erfolgen.

Dabei sollen die einzelnen Datensätze jeweils zeilenweise in einer Tabelle ausgegeben werden und die Felder der Tabelle sollen eine Überschrift tragen.

Die XSLT-Datei besteht aus folgenden Anweisungen:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- XSLT generated by Reisinger Susanne)-->
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="html" version="1.0" encoding="UTF-8" indent="yes"/>

<xsl:template match = "/">
    <html>
        <body>
            <table border="1">
                <tr><th>Gender</th><th><font color="#0000FF">Name</font></th><th>Address</th></tr>
                <xsl:apply-templates/>
            </table>
        </body>
    </html>
</xsl:template>

<xsl:template match="member">
    <tr>
        <th><xsl:value-of select="@gender"/></th>
        <td><xsl:value-of select="name"/></td>
        <td><xsl:value-of select="address"/></td>
    </tr>
</xsl:template>
</xsl:stylesheet>
```

Durch `<xsl:template match = "/">` wird der Wurzelknoten des XML-Quelldokuments angesprochen. Danach wird der Tabellenkopf mit den Überschriften und Schriftdefinitionen (fett, teilweise in blau, etc.) mittels (`<tr><th>Gender</th><th><font color="#0000FF">Name</font></th><th>Address</th></tr>`) definiert. Für den Knoten member wird ebenfalls das Aussehen definiert und danach jeweils direkt mittels xsl:value-of und select auf die Werte des Attributs gender (`<xsl:value-of select="@gender"/>`) und der Elemente name (`<xsl:value-of select="name"/>`) und address (`<xsl:value-of select="address"/>`) zugegriffen.

Die HTML-Ausgabe hat daher folgendes Aussehen:

Abbildung 25: Anzeige der erzeugten HTML-Ausgabe der Inhalte (Version 2)

Durch die Kombination von XPath, XSLT und XML-FO stellt XSL sicher, dass nur die Daten präsentiert werden, die relevant sind und, dass diese Daten in beliebiger Form dargestellt werden können.

Um XML-Dokumente parsen zu können wurde von W3C das Document Object Model (DOM, [W3Cc]) geschaffen. DOM stellt ein plattform- und sprachneutrales API für den Umgang mit XML-Dokumenten dar. Geparste XML-Dokumente werden dabei als Baumstruktur im Hauptspeicher verfügbar gemacht. Ein DOM-Parser liest immer das ganze XML-Dokument ein und gibt ein Objekt vom Typ Document zurück. An diesem hängt der gesamte Parsebaum.

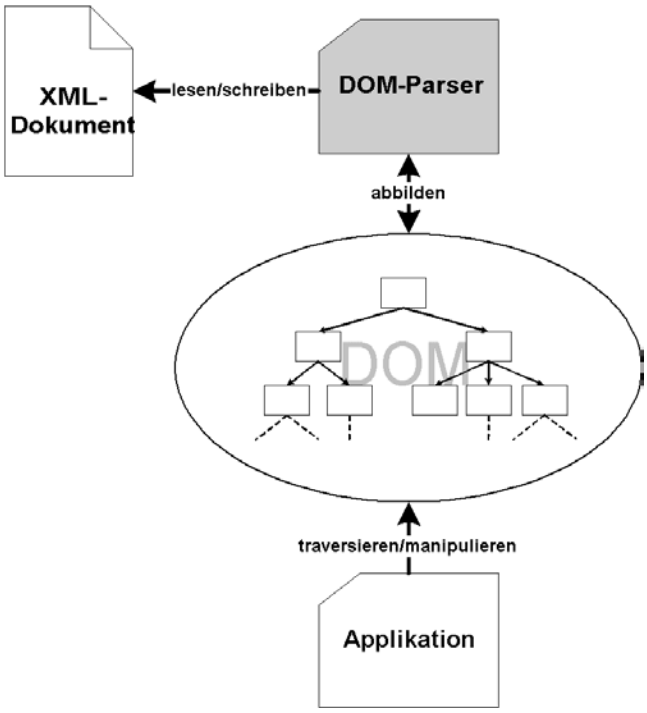


Abbildung 26: Funktionsweise eines DOM-Parsers

Mit DOM kann der gesamte Parsebaum nicht nur traversiert und evaluiert werden, sondern DOM bietet auch die Möglichkeit dem Baum zu manipulieren. Aufgrund dessen, dass alle Elemente eines XML-Dokuments Knoten mit Eigenschaften sind, können die einzelnen Konten beliebig adressiert, verändert, hinzugefügt und verschoben werden.

Alternativ zu DOM hat sich ein de-facto Standard Simple API for XML (SAX, [SAX]) entwickelt. Dieses einfachere API arbeitet Event-basiert, sprich während des Parsens wird die Applikation über gefundene Tags, etc. informiert.

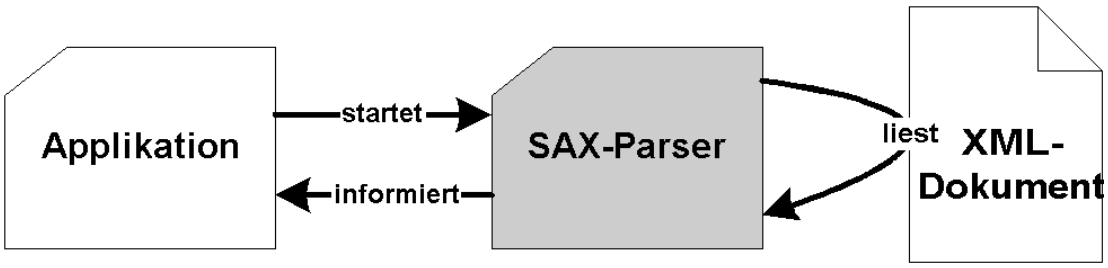


Abbildung 27: Funktionsweise eines SAX-Parsers

SAX ist einfach und leicht anwendbar, ein Vorteil ist, dass sehr große Dokumente schnell geparst werden können und kaum Speicher benötigt wird, dadurch dass kein Parsebaum aufgebaut wird. Aufgrund des rein sequentiellen Zugriffs, ist die Manipulation oder ein wahlfreier Zugriff (random access) auf Teile von XML-Dokumenten mit SAX nur erschwert möglich und DOM ist dafür besser geeignet.

Über die beschriebenen XML-Elemente hinaus, gibt es noch eine Reihe von XML-basierten Sprachen, Protokollen und Formaten auf die hier nicht genauer eingegangen wird. Die Entwicklung zeigt aber, dass man besonders im Bereich des Internet und bei der Entwicklung von Agenten nicht um XML vorbei kommt. Ganz im Gegenteil, gerade XML stellt in diesen Bereichen ein wichtiges Werkzeug dar. Insbesondere, wenn es darum geht eine gleiche Ausgangsbasis für Benutzer und Agenten zu schaffen (Ontologien, Kommunikation, Datenabgleich, etc.).

5. DISTANCE EDUCATION

Der Markt für Distance Education und Distance Education-Lösungen entwickelt sich rasant. „Nach einer Prognose des US-amerikanischen Marktforschungsinstituts IDC wird sich der Anteil des E-Learning an den gesamten Weiterbildungsaktivitäten von US-Unternehmen von 23 Prozent im Jahr 1998 auf 65 Prozent bis zum Jahr 2004 erhöhen. [CAV]“

Eine Studie ([UNI]) von Unicmind.com AG, die 2001 zum Thema E-Learning und Wissensmanagement in deutschen Grossunternehmen durchgeführt wurde, zeigt, dass bereits jetzt E-Learning bei diesen Top-350 Unternehmen der deutschen Wirtschaft eingesetzt wird (siehe Abbildung 28) und in der Zukunft noch stärker zum Tragen kommen wird.

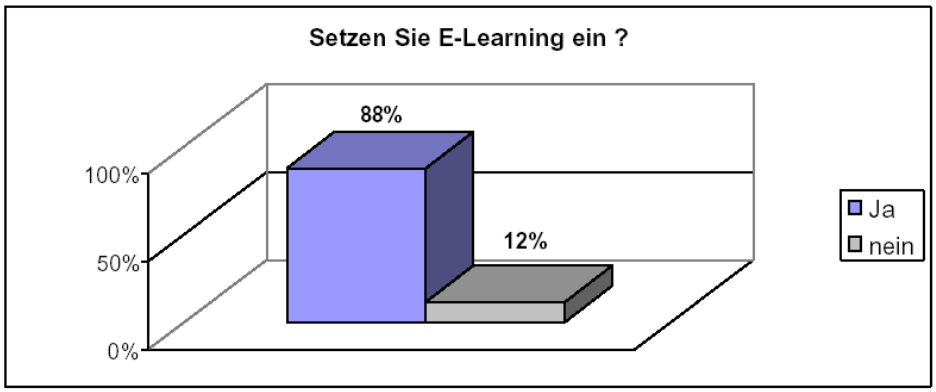


Abbildung 28: Einsatz von E-Learning derzeit (n=98) [UNI]

Auch ist der Prozentsatz der bereits mit E-Learning geschulten Mitarbeiter beachtlich und liegt teilweise bei über 50% (siehe Abbildung 29).

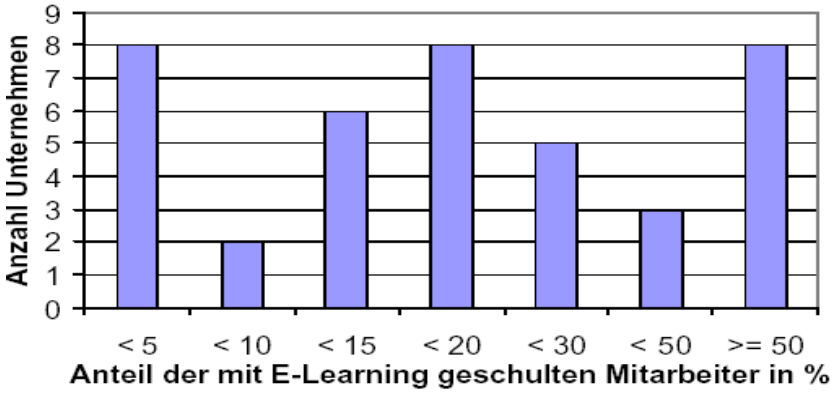


Abbildung 29: Anteile der Mitarbeiter in eLearning-Maßnahmen (n=40) [UNI]

Ebenso ist E-Learning teilweise auch mit über 20% im Budget bereits vorgesehen (siehe Abbildung 30)

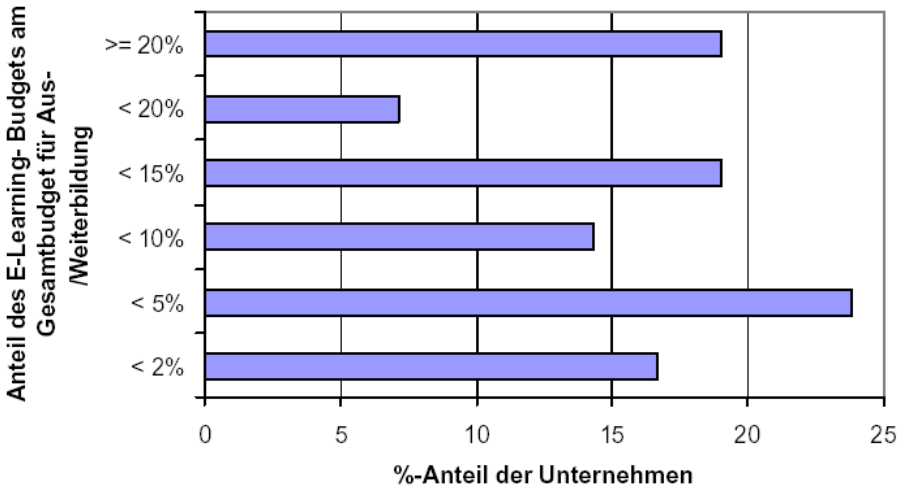


Abbildung 30: Anteil eLearning-Budget (n=42) [UNI]

Die befragten Firmen gaben auch Auskunft in welchen Bereichen sie bereits E-Learning einsetzen (siehe Abbildung 31) und in welchen Bereichen ihrer Meinung nach E-Learning zukünftig an Bedeutung gewinnen wird (siehe Abbildung 32).

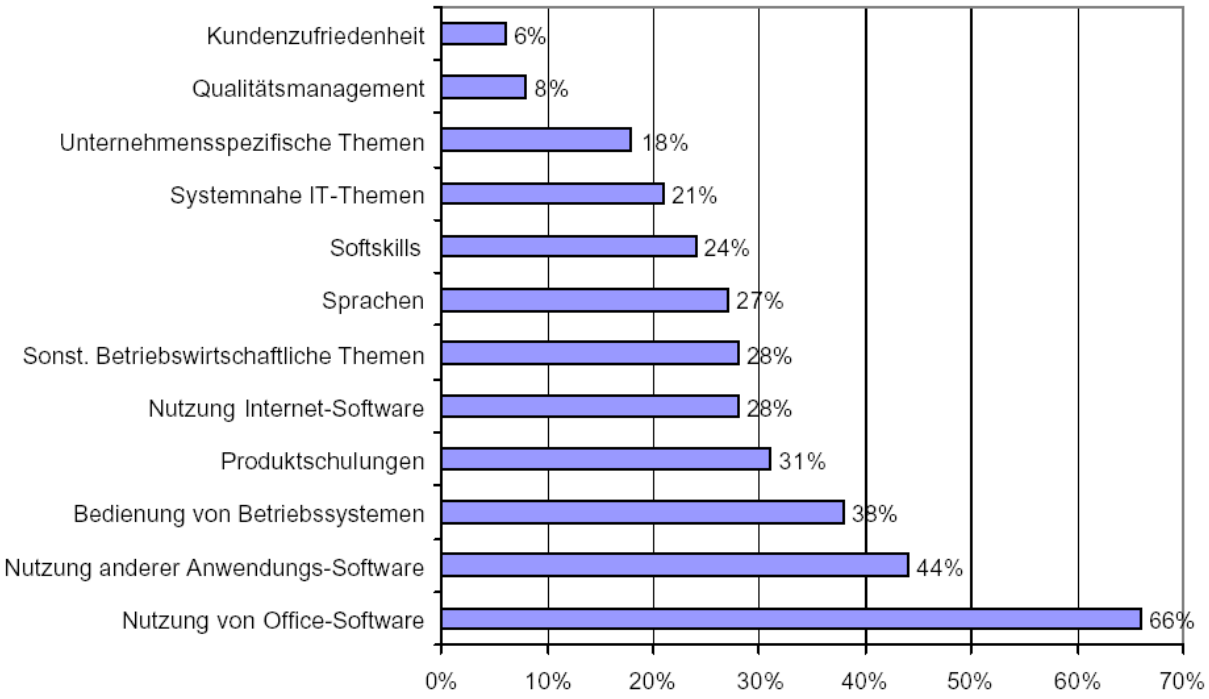


Abbildung 31: Schulungsthemen für eLearning heute (n=71) [UNI]

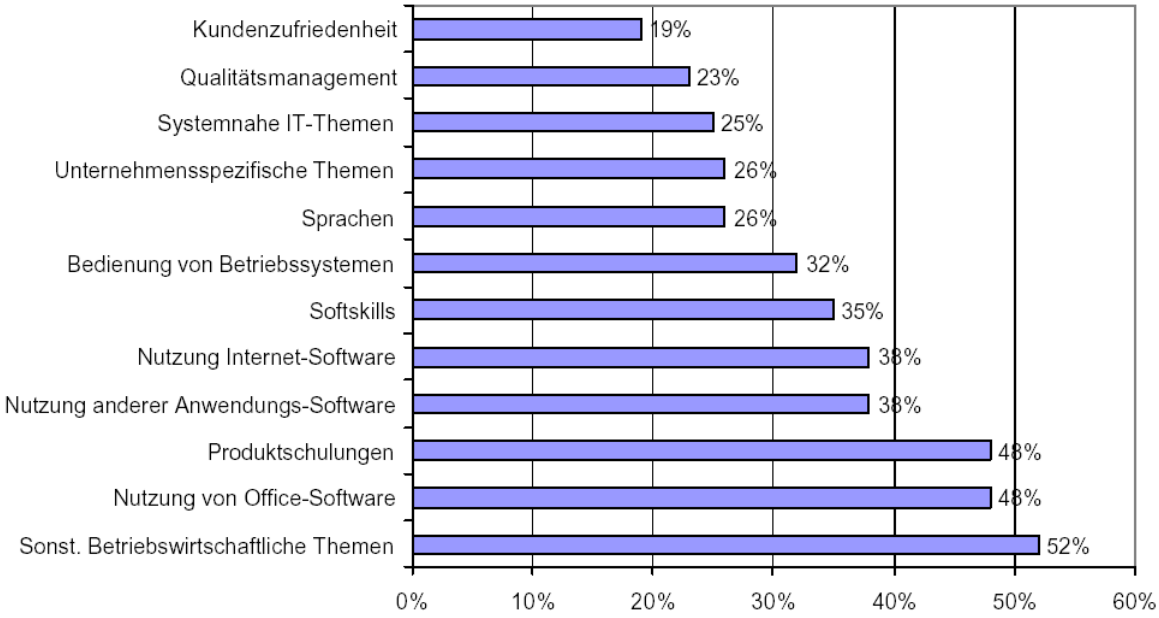


Abbildung 32: Schulungsthemen für eLearning zukünftig (n=69)

Die Ergebnisse zeigen, dass natürlich Bereiche, wie Produktschulung und die Nutzung von Internet-Software verstärkt genannt werden, aber auch besonders Bereiche, wie Kundenzufriedenheit, Qualitätsmanagement und Softskills eine Steigerung erfahren werden.

Abschließend soll noch aufgezeigt werden welche Hemmnisse, aber auch welche Chancen diese Firmen derzeit für E-Learning sehen.

Als Hemmnisse werden besonders die momentan fehlende Akzeptanz und der vermeintliche Wegfall des sozialen Umfelds und des sozialen Austauschs (social effects) genannt (siehe Abbildung 33).

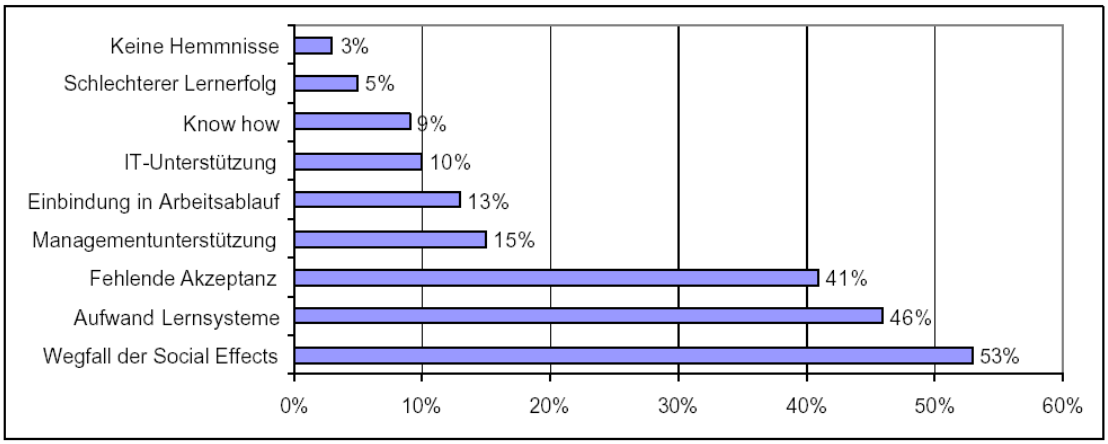


Abbildung 33: Hemmnisse von eLearning (n=78) [UNI]

Als Chancen werden jedoch die Möglichkeit des flexibleren Lernens, die hohe Aktualität und die Chance der Schulung aktueller Themen, aber besonders aus Unternehmenssicht auch die Aussicht auf Kostensenkung angeführt (siehe Abbildung 34).

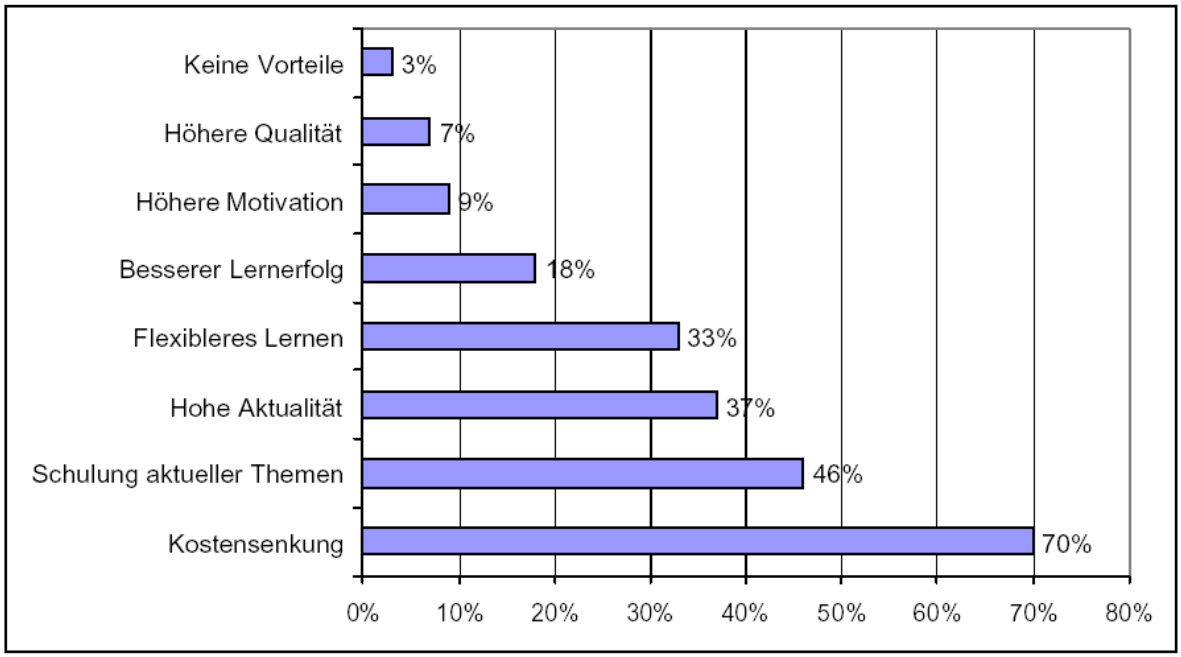


Abbildung 34: Chancen von eLearning (n=76) [UNI]

5.1. Formen von Distance Education

Was aber versteht man genau unter E-Learning, Distance Education, Open Learning etc. Im wesentlichen versteht man unter Distance Education die Anleitung zum selbstgesteuerten Lernen, nicht Face-to-Face, sondern vor allem auch mit den Mitteln der Informationstechnologie.

Jedoch gibt es auch in diesem Bereich, wie schon im Bereich der intelligenten Agenten unterschiedliche Sichtweisen und Auffassungen zu diesem Thema und ebenso unterschiedlich sind daher auch die Definitionen der Begriffe.

Eine Begriffsbestimmung über Telelernen, webbasiertes Distance Learning, E-Learning, etc. ist besonders in [MUE02b] zu finden. Seppo Tella geht in [TEL98] noch einen Schritt weiter in seiner Begriffswelt und spricht von Konzepten wie Distance Education, Open and Distance Learning, Flexi-Mode und Mixed-Mode Learning, wie in Abbildung 35 dargestellt.

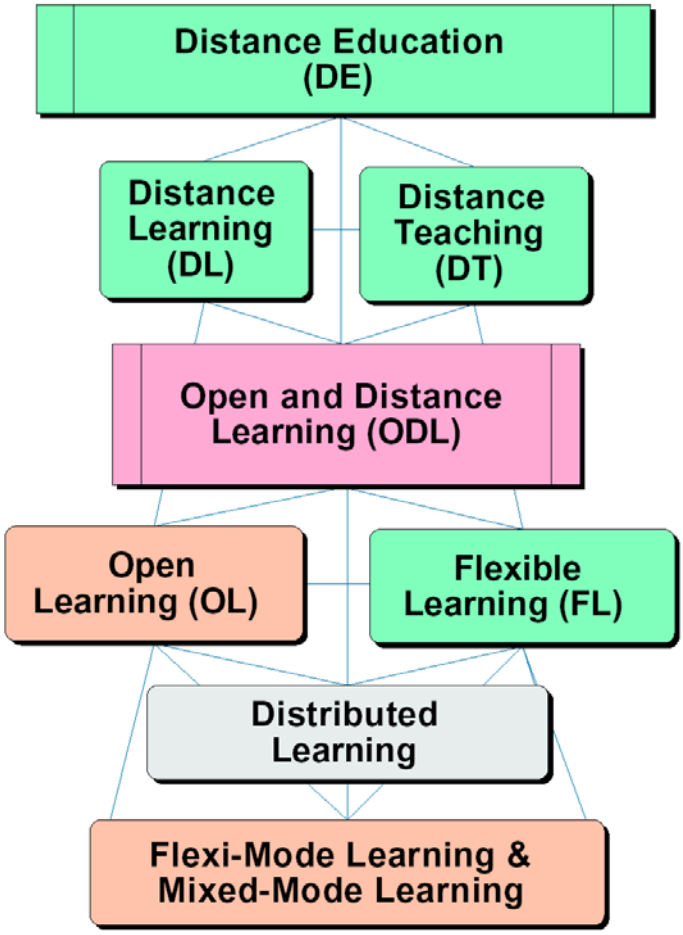


Abbildung 35: Unterschiedliche Formen von Unterrichtssystemen [TEL98]

Nach Tella wird Distance Education (DE) oftmals der „normalen" Unterrichtsform entgegengestellt, wobei er zwischen drei Generationen von DE unterscheidet. Die erste Generation, die durch eine einzelne Technologie charakterisiert ist und oftmals eine direkte Interaktion zwischen Lernenden und Lehrenden nicht möglich ist. Die zweite Generation die auf Multimedia basiert und in der bereits Lernmaterial zum Einsatz kommt, das speziell für Distance Learning ausgelegt ist. Weiters gibt es hier auch die Möglichkeit zur wechselseitigen Interaktion zwischen den Lernenden und Lehrenden bzw. dem Tutor. Die dritte Generation von Distance Education lässt nicht nur eine wechselseitige, sondern eine multidimensionale und auch eine direkte synchrone Kommunikation und Interaktion zwischen allen Beteiligten zu.

Den wesentlichen Unterschied zwischen Distance Education, Distance Teaching und Distance Learning sieht Tella darin, auf welche Personen bzw. Systeme sich das Hauptaugenmerk richtet. Bei Distance Education steht seiner Ansicht nach das verwendete Unterrichtssystem

im Vordergrund, wohingegen Distance Teaching lehrerzentriert und Distance Learning lernerzentriert ist.

Beim Begriff Open Learning (OL) sieht Tella vor allem drei Merkmale: Offenheit, Lernerzentriertheit und die Tatsache, dass es sich bei Open Learning mehr um eine Philosophie handelt bzw. die Grundüberlegung in Richtung eines flexiblen Lehr- und Lernprozesses geht.

Bates sieht den Unterschied folgendermaßen: „[..] open learning may include distance education, or it may depend on other flexible forms of learning, including a mix of independent study and face-to-face-teaching. It may also include other concepts, such as open access without prior requisite qualifications. Both open-ness [hyphenated in the original] and distance education are never found in their purest forms. No teaching system is completely open, and few students ever study in complete isolation. Thus there are degrees of open-ness and 'distance'—indeed, distance is more likely to be psychological or social, rather than geographical, in most cases " [BAT95]

Open Learning (OL) zusammen mit Flexible Learning (FL) und Distance Learning (DL) ergeben das Konzept des Open und Distance Learning (ODL), welches vor allem durch die ODL Programme der Europäischen Union als terminus technicus geprägt wurde. Die in ODL zum Einsatz kommenden Werkzeuge und Programme sind oftmals die selben wie die bei Distance Education. In ODL geht das Interesse aber weg von einer lehrerzentrierten Umgebung hin zu einer offenen lernerzentrierten und virtuellen Lernumgebung. Das Haupt-augenmerk geht dabei vor allem auch in Richtung Groupware.

Die Begriffe Distributed Learning und Distance Education werden oft synonym verwendet, jedoch besteht laut Bates [BAT96] der wesentliche Unterschied darin, dass reguläre Universitätsstudenten zwar teilweise Kurse über das Web oder CD-ROM belegen können, jedoch auch „physisch" an Kursen teilnehmen müssen. Daher lässt sich diese Form, sprich Distributed Learning, nicht mit Distance Education oder Open Learning vergleichen.

Die Begriffe Flexi-Mode Learning und Mixed-Mode Learning werden laut S. Tella verwendet um unterschiedliche Formen von Distance Learning und Face-to-Face Learning zu beschreiben. Flexi-Mode Learning wird bei Tella [TEL98] als eine Kombination zwischen

Distance und Face-to-Face Learning beschrieben, bei dem sowohl elektronisches als auch gedrucktes Lernmaterial Verwendung finden.

Bates kommentiert die Begriffs- und die Umsetzungssituation von Distance Education in [BAT95] wie folgt „the poor relatives of the mainstream educational system, whether we call them distance education, open and distance learning or flexi-mode learning, will gradually come to the center stage, and more and more high-profile teachers as well as students will come to realise their intrinsic value and start using their potential, physically and virtually."

5.2. Warum Distance Education?

Warum aber ist Distance Education überhaupt sinnvoll? Bereits das 20ste und vor allem das 21ste Jahrhundert sind Jahrhunderte des Wandels. Der technische Fortschritt schreitet immer schneller voran und die Dauer der Anwendbarkeit, sprich die Halbwertszeit von produkt-orientiertem Wissen wird immer kürzer. Fachwissen muss permanent erneuert, vertieft und erweitert werden. Das Wissen, das man im Laufe des Lebens benötigt, kann nicht mehr nur in der Jugend erlernt werden. Berufs- bzw. lebensbegleitendes Lernen (Aus- und Weiterbildung) wird immer wichtiger und zentraler Bestandteil des Lebens.

An eine berufsbegeleitende Aus- und Weiterbildung werden nach [SCH01] folgende Anforderungen gestellt:

- Sie muss **modular** sein um individuell unterschiedliche Defizite ausgleichen zu können.
- Sie muss **effizient** sein um Zeitbedarf und Kosten zu minimieren.
- Sie muss leicht **aktualisiert** und veränderten Zielsetzungen angepasst werden können.
- Sie soll weder **zeit- noch ortsgebunden** durchgeführt werden können.

Um diese Forderungen zu erfüllen wird Distance Education (DE) immer wichtiger. DE erlaubt eine freie Zeiteinteilung. Die elektronisch aufbereiteten und präsentierten Lerninhalte können von den Lernwilligen jederzeit (just-in-time, learning on demand) und von überall aus (Büro, Wohnung, von unterwegs) abgefragt und bearbeitet werden. Distance Education kann

dabei ausschließlich elektronisch, hybrid als Mischform aus elektronischem und face-to-face Learning und traditionell mit DE-Elementen stattfinden.

Immer mehr geht ein Paradigmenwechsel in der Art und Weise des Lernens und in der Vermittlung von Wissen vor sich. Die traditionelle Auffassung, dass der Lehrer die Information präsentiert und die Lernenden dieses Wissen passiv konsumieren (push-Prinzip) wird durch aktives selbstgesteuertes Lernen (pull-Prinzip) zunehmend abgewechselt. Die traditionelle Instruktion wird von Mandl [MAN01] wie folgt dargestellt:

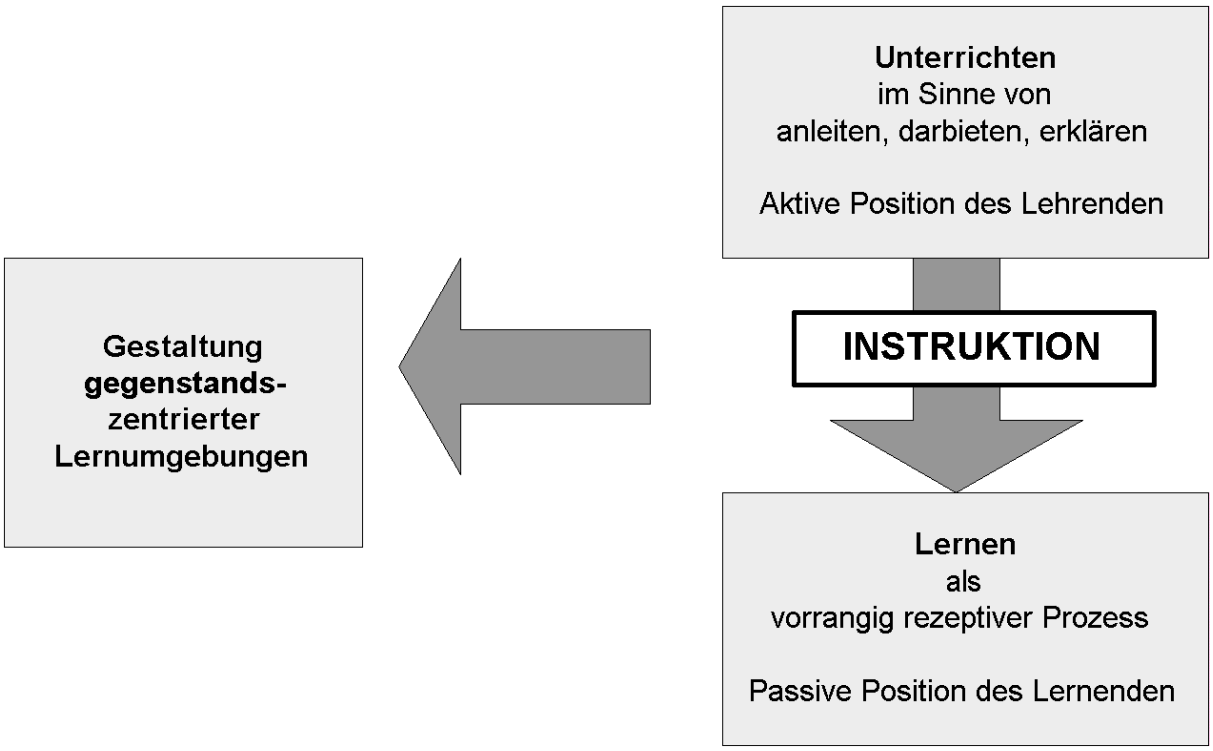


Abbildung 36: Traditionelle Auffassung vom Lehren und Lernen [MAN01]

Die Aneignung von Wissen wandelt sich jedoch nach der Vermittlung der 4 Kulturtechniken (Lesen, Schreiben, Rechnen, Umgang mit Information) zunehmend von einer Bringschuld zu einer Holschuld. Der Lernende ist aufgrund des immer schneller voranschreitenden Wandels gefordert aktiv zu sein und sich seinen Wünschen, Neigungen, Fähigkeiten und den jeweiligen Erfordernissen entsprechend Wissen aktiv anzueignen. In [LED01] wird dieser Wissens- und Kompetenzmanagementprozess durch folgende Graphik dargestellt:

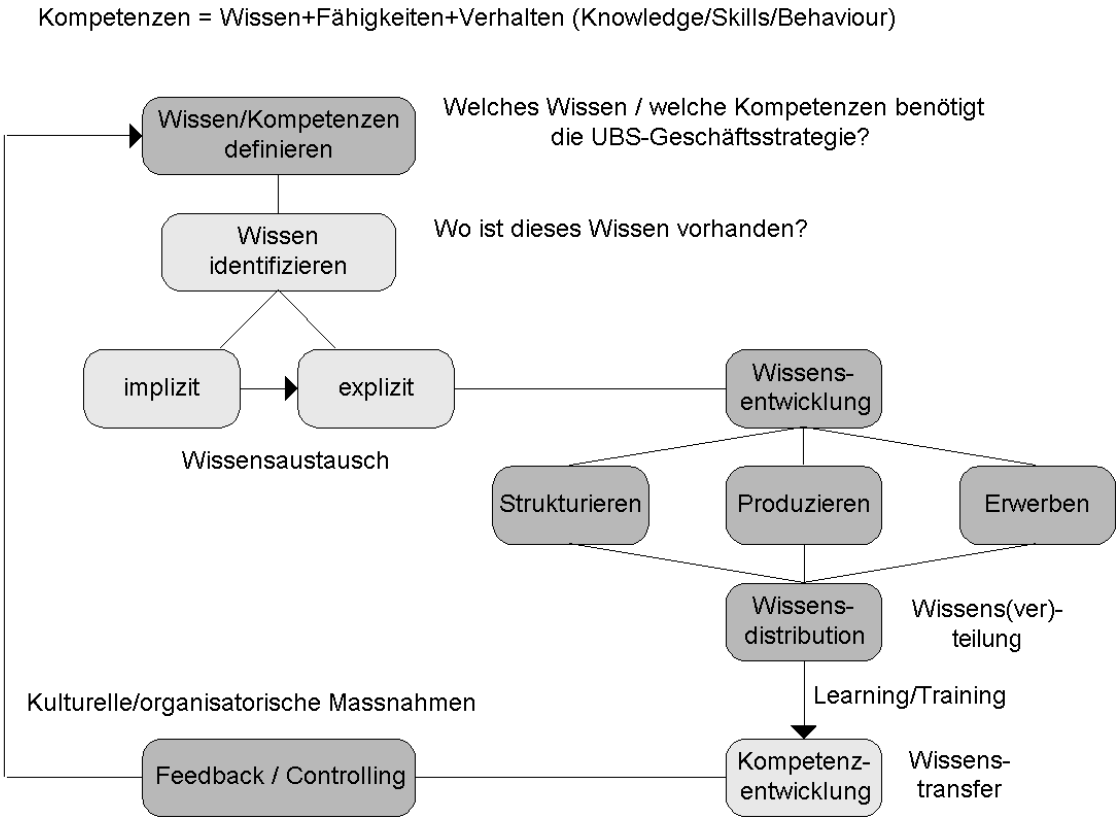


Abbildung 37: Wissens- und Kompetenzmanagementprozess [LED01]

Wobei Wissensmanagement in [LED01] definiert wird als: Organisieren, damit die richtigen Personen das richtige Wissen zur richtigen Zeit im optimalen Umfeld mit individuellen Methoden und effizienten Instrumenten entwickeln und dieses in den Arbeitsprozessen einsetzen können.

Dabei spielt Distance Education aufgrund seiner Eigenschaften wie Zeit- und Ortsunabhängigkeit, Modularität, Flexibilität und einer beliebigen Vervielfältigbarkeit (Globalisierung) eine bestimmende Rolle in der Aus- und Weiterbildung.

Distance Education wird berufs- und lebensbegleitend eingesetzt. DE ist modular und daher auch individualisierbar. Es kann global, just-in-time und on-the-spot eingesetzt werden. Eigenverantwortung (pull statt push) und die Selbstorganisation sind Hauptmerkmale von DE, weshalb es einen Paradigmenwechsel zum traditionellen Face-to-Face Unterricht bedeutet.

Distance Education soll dabei aber nicht als Ersatz des Lehrers verstanden werden. Der Lehrer spielt weiterhin eine zentrale Rolle im DE, aber seine Rolle wandelt sich hin zum Coach. Schauer schreibt in [SCH01]: „Die Kompetenz und Autorität des Lehrers darf nicht

auf dessen Wissensvorsprung gegenüber dem Schüler basieren, sondern muss aus der Didaktik, Methodik und Persönlichkeit des Lehrers hervorgehen.“ Der Coach ist derjenige, der die Lernenden im Lernprozess begleitet, ihnen Lernen lehrt, sie animiert, motiviert und Hinweise zur Problemlösung präsentiert. Dies wird als konstruktivistische Auffassung dargestellt als ([MAN01]):

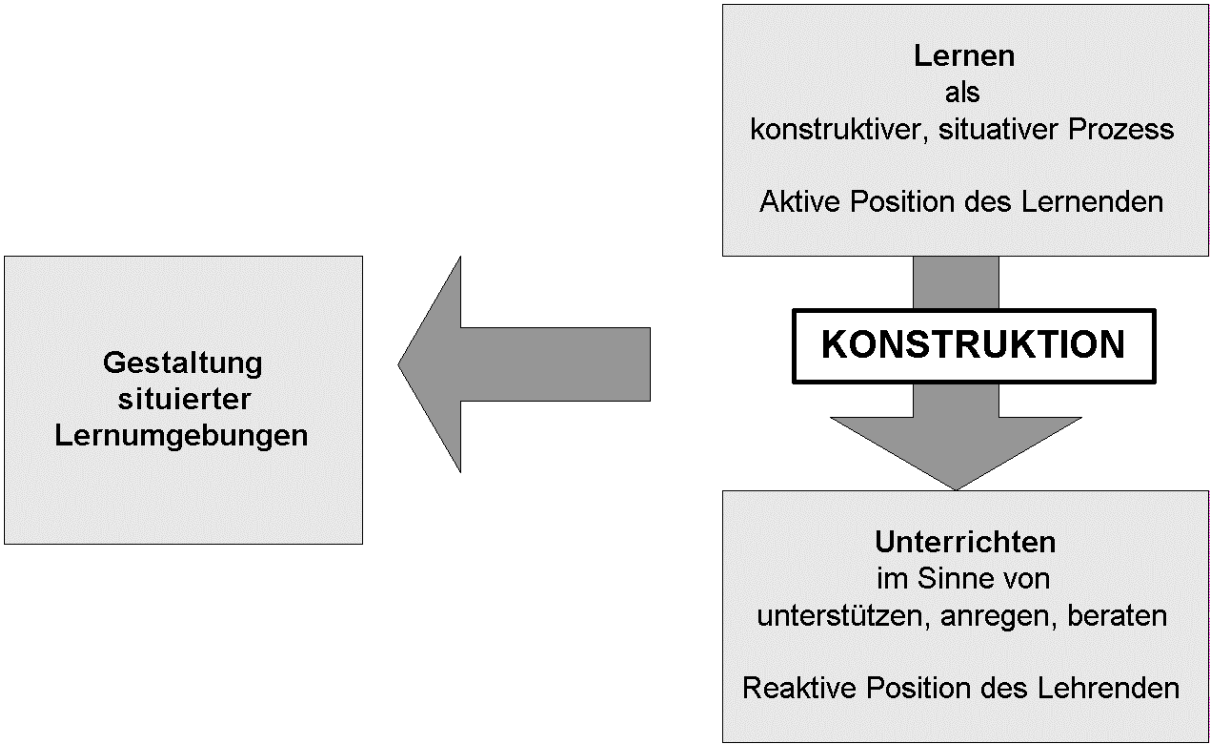


Abbildung 38: Konstruktivistische Auffassung vom Lernen [MAN01]

Die neue Lernkultur ist somit ein Wandel von der Instruktion hin zur Konstruktion. Weg vom Unterrichten im Sinne von anleiten, darbieten und erklären. Hin zum Lernen als aktiver, selbstgesteuerter, konstruktiver, situativer und sozialer Prozess ([MAN01]).

In [BRI92] werden zusätzlich tabellarisch die instruktionalen Ereignisse dargestellt, die von Robert Gagné beschrieben wurden. Sein Modell „Events of Instruction“ ist vor allem in den USA populär. Gagné beschreibt darin eine Folge von neun Ereignissen, die seiner Meinung nach, bei jedem Lehr- und Lernsystem stattfinden müssen. Die Aktivitäten der beiden Instanzen Lehrender und Lernender korrespondieren dabei miteinander.

	AKTIVITÄT LEHRSYSTEM	AKTIVITÄT LERNSYSTEM
1	Aufmerksamkeit erzielen	Konzentration mobilisieren
2	Lehrziele mitteilen	Realistische Erwartungen über Lernergebnis aufbauen
3	An Vorwissen anknüpfen	Langzeitgedächtnis aktivieren
4	Lernmaterial präsentieren	Lernmaterial wahrnehmen
5	Lernhilfe anbieten	Übernahme in Langzeitgedächtnis durch semantische Enkodierung fördern
6	Gelerntes anwenden	Rückschlüsse auf Lernergebnisse ermöglichen
7	Rückmeldung geben	Diagnostische Information und Verstärkung geben
8	Leistung testen	Hinweise zur Verfügung haben, die bei der Erinnerung benötigt werden
9	Behaltensleistung und Lern-transfer fördern	Leistung in neuen Situationen erproben

Tabelle 4: Gegenüberstellung von instruktionalen Ereignissen in Lehr- und Lernsystemen [BRI92]

5.3. Mögliche Vor- und Nachteile von Distance Education

Mögliche Vor- und Nachteile von Distance Education wurden in [MUE02b] von Mühlbacher ausführlich beleuchtet und aus dieser Quelle als Zitat übernommen:

5.3.1. Mögliche Vorteile von Distance Education [MUE02b]

- Zeitliche Unabhängigkeit

- Örtliche Unabhängigkeit: Giegerich und Füllen [GIE97] nennen die örtliche Ungebundenheit des Unterrichtenden „nomadic instruction“.

- Bessere Voraussetzungen für Studieren neben Beruf und/oder Familie: Durch die Orts- und Zeitungebundenheit wird Berufstätigen ein Studium ohne Karriereunterbrechung oder Verdienstausfall ermöglicht. Dies gilt auch für andere Verpflichtungen, wie z.B. der Pflege familiärer Strukturen.

- Individuelleres Lernen möglich: Baumgartner ([BAU97]) beschreibt den Vorteil, dass durch die zeitliche Unabhängigkeit und Offenheit hinsichtlich des Lernstoffes ein individuelleres Lernen möglich ist. Welche Stoffteile wie lange gelernt werden, bleibt dem Studierenden überlassen, was sich v.a. in heterogenen Gruppen positiv auswirkt.

- Höhere Aktualität des Studiums: Durch die relative Neuheit der Unterrichtsform ist meist auch ein starker Gegenwartsbezug des Materials gegeben.

- Hoher Praxisbezug: Durch die vermehrte Teilnahme berufstätiger Studierender ist häufig ein höherer Zusammenhang zwischen Theorie und Praxis gegeben.

- Meistens keine fixen Inskriptionszeiten: In vielen Distance Learning Angeboten kann ein Studium zu jeder Zeit begonnen werden.

- Leistungs- statt Zeitsemester: Das Studium ist (in den meisten Fällen) durch die Absolvierung bestimmter Leistungen geregelt und nicht eingeteilt in fixe (Zeit-)Semester.

- Zeitdruck verringert: Durch die oben beschriebe Einteilung in Leistungssemester sinkt auch der Zeitdruck auf die Lernenden. Ein Studium kann (meist) ohne Konsequenzen verlängert werden.

- Studieren in angenehmer Atmosphäre: Keine überfüllten Hörsäle und andere Stress-faktoren.

- Geringere Kosten für Bildungsinstitutionen und Studierende: Kosten der räumlichen Infrastruktur und Personalkosten (bezogen auf die größere Menge an Teilnehmern) auf institutioneller Seite, Reise- und Aufenthaltskosten von Studierenden und Lehrenden verringern sich, während Hard- und Softwarekosten (sowie das die Technik betreuende Personal), Telefongebühren etc. als zusätzliche Kosten dazukommen. Auch der Aufwand und die Kosten für die Erstellung der Kursmaterialien sollten berücksichtigt werden.

- Globales Angebot an Unterricht: Durch die örtliche Ungebundenheit ergibt sich eine höhere Auswahlmöglichkeit in den Bildungsangeboten.

- Mehr Chancengleichheit beim Zugang zur Bildung: Viele Distance Learning Angebote sind ohne Voraussetzungen hinsichtlich des persönlichen Ausbildungsgrades belegbar.

Auch der Unterschied zwischen „armen“ und „reichen“ Institutionen wird durch den Zugang zum selben Unterrichtsmaterial geringer ([AST99]).

- Mayer [MAY98] verweist in einer Studie darauf, dass unter bestimmten Voraussetzungen die Kombination verschiedener Zeichensysteme in Multimedia die Bildung mentaler Modelle verbessert.

- Durch das Auslagern institutionalisierter Bildung wird Raum geschaffen für neue Lernziele, wie z.B. Persönlichkeitsbildung in akademischer und gesellschaftlicher Hinsicht.

- Neue Berufschancen und –perspektiven für Lehrende.

5.3.2. Mögliche Nachteile von Distance Education [MUE02b]

Als mögliche Nachteile von Distance Education sind laut [MUE02b] folgende zu nennen:

- Geringere soziale Kontakte zwischen Lehrern und Studierenden, sowie den Studierenden untereinander.

- Zusätzliche Kompetenzen gefragt: Ein gewisses Grundwissen im Umgang mit der Technik und den Lernstrategien selbstgesteuerten Lernens wird vorrausgesetzt, bzw. muss erlernt werden. Handelt es sich um internationale Distance Learning Angebote, so ist zusätzlich die Beherrschung der jeweiligen Fremdsprache nötig (vgl. [AST99], [ZIM97]).

- Technische Voraussetzungen: Um an einem Multimedia gestütztem Unterricht teilnehmen zu können, ist teuere technische Ausrüstung notwendig ([ZIM97]). (Anmerkung: Dieses Argument wird durch die technische Entwicklung und den Preisverfall technischer Geräte über die Zeit hinweg an Bedeutung verlieren.)

- Probleme mit der Technik: Der Einsatz neuer Technologien bedeutet auch das Auftreten neuer Probleme, z. B. bei Softwareinstallationen oder Ausfall der Anbindung an das Internet.

- Eigenmotivation ist gefragt: Durch die verringerte direkte Kontrolle durch Lehrende ist ein hohes Maß an Selbstdisziplin notwendig, um die gesteckten Lernziele zu erreichen.

- Die offene Lernform bereitet in der Regel schwächeren Studierenden mehr Probleme als guten Studierenden ([AST99]).

- Terminprobleme bei verschiedenen Zeitzonen: Findet der Distance Learning Unterricht in internationalem Kontext statt, so können durch die Verteilung der Beteiligten auf unterschiedliche Zeitzonen Probleme bei Terminvereinbarungen für zeitliche synchronisierte Unterrichtseinheiten (Video conferencing) auftreten.

- Die Möglichkeit des globalen Unterrichtens bringt eine Tendenz zur Zentralisierung mit sich, die zur Monopolisierung solcher Institutionen führen kann. Lokale Entscheidungsträger verlieren dadurch an Einfluss, da Entscheidungen überregional getroffen werden und den Bedürfnissen lokaler Gegebenheiten vielleicht nicht entsprechen.

- „Konkurrenz“ zum traditionellen Unterricht: Durch die Konkurrenzsituation der beiden Unterrichtsformen könnten Arbeitsplätze in traditionellen Einrichtungen gefährdet sein. (Anmerkung: Neue Studienpläne im Lehramtsbereich nehmen darauf allerdings bereits Rücksicht.)

- Erstellung von Lehrmaterial aufwendig: Wiemer [WIE97] beschreibt im Zusammenhang mit der Produktion von multimedialem Material, dass die Erstellung relativ aufwendig ist und zusätzliche Kompetenzen des Unterrichtenden verlangt. Der Materialtausch zwischen den Institutionen erwies sich als schwierig, da einerseits ein Mangel an qualitativ hochwertigem Material besteht (siehe auch [ZIM97]), andererseits Copyright-Bestimmungen den Austausch behindern. Giegerich und Füllen ([GIE97]) weisen darauf hin, dass auch die „Pflege“ des Materials, insbesondere der darin meist vorkommenden Hyperlinks in das Internet, einen zusätzlichen Aufwand bedeutet.

5.4. Wichtige Elemente in der Distance Education

5.4.1. Lernumgebung – Plattform

Gerade im Bereich der Distance Education gibt es zahlreiche Plattformen, die teilweise sehr stark im Funktionalitätsumfang, in der Benutzerfreundlichkeit, Performance und vor allem im Preis differieren. Zusätzlich ergeben sich je nach Umfeld, Kurstyp, Zielgruppe, usw.

unterschiedliche Anforderungen an eine Plattform. Wichtig ist, dass sich eine Plattform an die Gegebenheiten anpasst und nicht umgekehrt.

Eine Plattform sollte dabei Anforderungen in den Bereichen Inhalt (content), Dienste (services) und Technology (technology) erfüllen. In [UNI] sind einige wichtige Aspekte dargestellt (siehe Abbildung 39).

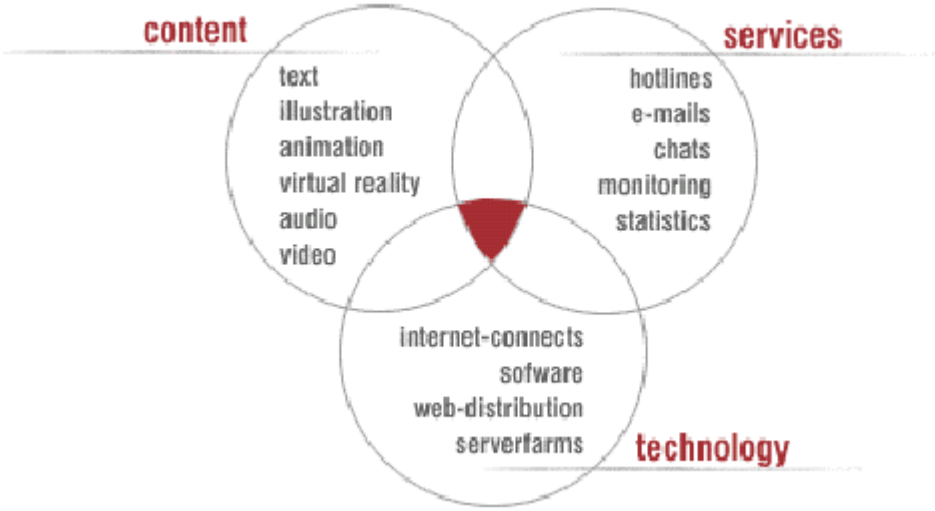


Abbildung 39: Wichtige Aspekte einer Lernplattform [UNI]

Eine Plattform soll möglichst universell sein und variable Inhaltsformate und –darstellungen zulassen. Dienste, wie multiple Kommunikationskanäle, Help-Services, aber auch Monitoring und Evaluierungssupport machen eine Plattform aus. Um einen Kurs effizient anbieten zu können, soll die Organisation von unterschiedlichen Arten von Foren (Gruppen-, Informations-, Diskussionsforen, etc.) möglich sein. Generell müssen Kursanbieter auf der Kursorganisations- und Administrationsseite Unterstützung durch ein offenes dynamisches System finden, das eine simple und flexible Einrichtung und Durchführung von unterschiedlichen Kursmodellen und die Präsentation, Erweiterung und Wartung von Kursmaterialien nicht nur zulässt, sondern aktiv unterstützt. Auch „personal webspaces“ für Lehrende und Lernende gehören zu den Grundfunktionen einer Lernplattform. Nicht zuletzt spielt auch die Technologie, welche die Plattform voraussetzt und unterstützt, eine wichtige Rolle für den Einsatz einer Lernplattform. Gerade eine Distance Education Plattform muss bestimmten technischen Anforderungen genügen. Unterschiedliche Internetanbindungen der Lernenden müssen unterstützt werden. Darüber hinaus muss sie für Kursanbieter so einfach wie möglich zu installieren sein und von allen intuitiv eingesetzt werden können. Last, but not

least muss das Erscheinungsbild (Interface) einer Lernplattform übersichtlich und eindeutig gestaltet sein. Auch die Navigation muss einheitlich und simpel sein und die Kurse müssen sich in die Lernplattform integrieren lassen.

5.4.2. Kursstruktur und Lernmaterial

Beim konventionellen Unterricht ist das Lernmaterial oftmals rein sequentiell strukturiert. Bei Distance Education kann das Lernmaterial aber unterschiedliche Strukturen aufweisen. Zum einen kann es sequentielle Elemente geben. Elemente mit unterschiedlich aufbereiteten Pfaden und gleichen oder vertiefenderen Inhalten können enthalten sein. Auch gibt es die Möglichkeit, dass Elemente übersprungen werden können, etwa bei entsprechenden Vorkenntnissen der Lernenden können nach einer Einführung in ein Themengebiet, die eigentlichen Erklärungen und Ausführungen übersprungen werden und sofort zum Assessment-Teil übergegangen werden. Und last but not least kann es auch die Möglichkeit geben, dass die Lernenden vollkommen autonom ihren individuellen Weg durch den Lernstoff suchen und kein roter Faden oder möglicher Pfad durch den Lernstoff aufgezeigt ist.

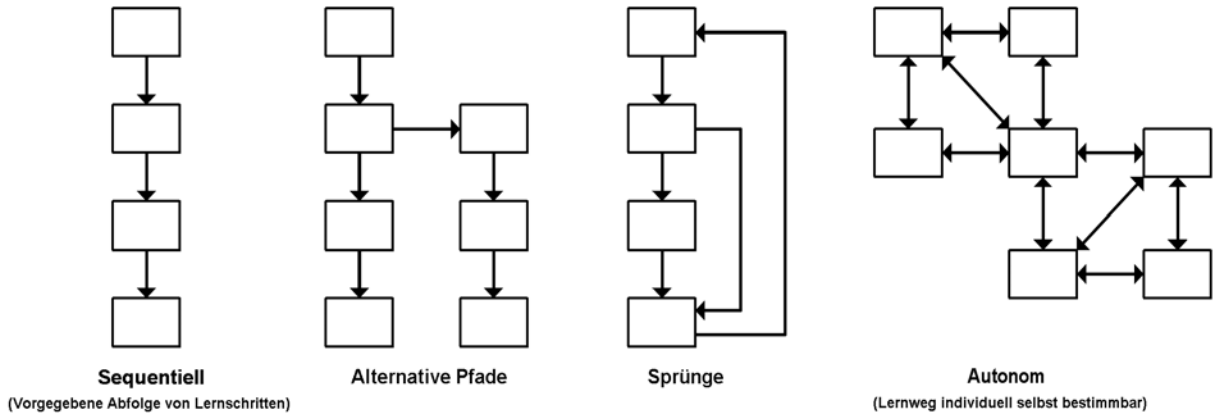


Abbildung 40: Unterschiedlich Arten von Lernmaterialstrukturierung

Der individuelle Pfad durch den Lernstoff beinhaltet den Vorteil, dass der Lernende sich punktgenau mit Detailinformationen beschäftigen und komplexe Sachverhalte nach eigenen Mustern erlernen kann. Möglich ist jedoch auch, gerade beim Einstieg in diese neue Lernform, eine Desorientierung und damit auch eine Überforderung der Lernenden durch die ungewohnte Struktur.

Oftmals treten die vorgestellten Strukturen als Mischform innerhalb eines Lernmaterials auf. Teils ist ein individueller Weg möglich, Teile davon sind aber sequentiell gestaltet, da manches im Kontext miteinander gesehen und erlernt werden muss. Dafür soll es für den Lernenden einen Leitfaden durch den Lernstoff geben. Die Kursdaten wurden um eine Metadaten-Struktur erweitert (siehe Kapitel 7), die einerseits einen individuellen zielgruppenorientierten Leitfaden ermöglicht und andererseits schon im Vorfeld den Lernenden und Kursorganisatoren wichtige Informationen gibt.

Auch das Lernmaterial kann vielfältig gestaltet werden (siehe dazu auch 6.2.2 Eingebettetes Kursmaterial).

In [KER02] wird dies folgendermaßen beschrieben: „Mediale Lernangebote, die sich als didaktisch sinnvoll und in der Nutzung erfolgreich erwiesen haben, können z.B. durch folgende Charakteristika beschrieben werden:

- Die Lernprozesse basieren in starkem Maße auf *Eigenaktivitäten* der Lernenden. Die Lernenden sollen in der Verfolgung ihrer (Lern-) Interessen durch die Umgebung unterstützt werden (z.B. durch empfohlene Lernpfade, Hinweise oder Rückmeldungen), aber gleichzeitig so wenig wie nötig bei ihren Lernaktivitäten eingeschränkt werden.

- Die Medien sind so aufbereitet, dass sie das *Eintauchen* in eine Umwelt, die Lernprozesse besonders anregt, fördern: Die Beschäftigung mit dem Medium sollte „in sich“ motivierend sein. Hierbei helfen insbesondere Bezüge („Anker“) zu Ereignissen oder Objekten der realen Welt, die Darstellung aus verschiedenen Sichtweisen und die direkte Manipulierbarkeit der dargestellten Objekte durch die Lernenden.

- Sie beinhalten unterschiedliche Arten von Medien (Einzel-, Multi- oder Telemedien), Hilfsmitteln (Geräte), Einrichtungen Selbstlernzentrum, Lerninseln usw.) und personalen Dienstleistungen (Medienberatung, tutorielle Betreuung usw.), die systematisch aufeinander *bezogen* sind.

- Das mediale Lernangebot ist ggfs. Teil einer bewusst gestalteten physikalisch-sozialen Umwelt, z.B. eines Weiterbildungs- oder Fernstudiensystems, mit unterschiedlichen Arten personaler Betreuung und Dienstleistungen.

- Digitale Multimediasysteme sind als Elemente solcher Arrangements besonders interessant, weil sie didaktische Konzepte, wie die Forderung nach einer anschaulichen Darstellung oder nach reaktiven oder gar interaktiven Systemen, besonders gut einlösen."

5.4.3. Didaktisches Modell

Bei Distance Education kann es eine Vielfalt von didaktischen Modellen geben. Diese unterscheiden sich nicht zuletzt dadurch, ob der Lernende vollkommen autonom, unterstützt durch Gruppen und/oder Tutoren und Coaches lernt. Aber auch, ob ausschließlich webbasiert gelernt wird oder ob ein hybrides Lernmodell vorliegt.

Beim hybriden Lernmodell handelt es sich um eine Mischform aus Präsenzphasen (realen Treffen) und webbasiertem Lernen und Lernunterstützung. Bei dieser Form tritt der Lehrende als Coach in Erscheinung, auch Gruppenbildung und/oder selbständiges Lernen sind mögliche Ausprägungen.

Ein didaktisches Konzept kann daher je nach Typ und Inhalt des Kurses, Zielgruppe, verfügbaren Ressourcen (Lehr-/Lernmaterial, Mitarbeiter, ...) ein Mix aus folgenden Methoden sein:

- Kick-Off Meetings, Meilensteine, Abschlusstreffen

- Impulsreferate durch Experten in der einleitenden Präsenzphase (schaffen Motivation)

- Betreutes Online-Lernen (Offline- und Online-Studium der Unterlagen, begleitende, anleitende fachliche und pädagogische Betreuung durch Tutoren und Coaches)

- Anwendung des gelernten Wissens in Fallbeispielen, Seminararbeiten, Assessment und Learning-by-Teaching

- Gruppendiskussionen über erlerntes Wissen

- Moderierte Gruppendiskussionen, Chats

In [KER02] werden noch weitere typische Bestandteile für ein hybrides Lernarrangement dargestellt. Die einzelnen möglichen Bestandteile werden dabei durch Beispiele für deren didaktisch-methodische Organisation, sowie telemedialen Varianten tabellarisch gegliedert.

BESTANDTEIL	BEISPIELE FÜR DIE ORGANISATION	TELEMEDIALE VARIANTE
1. Vortrag mit Diskussion	Frontalunterricht im Seminarraum, Hörsaal	Rundfunk, Video auf Abruf (digitales Fernsehen, Internet, etc.), Video-konferenz
2. Selbstlernaktivität	Print-, AV-Medien (Kassette), Multimedia (CD-ROM, DVD) am Arbeitsplatz, im Selbst-lernzentrum oder zu Hause	Rundfunk, WWW-Seiten
3. Kooperatives Lernen	Partner- und Gruppenarbeit (inhaltsbezogen)	Computer mediated communication/conferencing (CMC), Groupware-basierte Kooperation
4. Tutoriell betreutes Lernen	Mentoren-Modelle (auch: Peer-Tutoren)	Tele-Coaching, Tele-Tutoring
5. Kommunikatives/ soziales Lernen	Gruppenansätze (Team-Building, Gruppenfeedback, Metakommunikation, etc.)	Internet-Café, Chat-Räume, Diskussionsforen
6. Beratung	Einzelgespräche, Informationsveranstaltungen	E-Mail, WWW
7. Tests, Zertifizierung	Klausur, Prüfung, computerbasiertes (adaptives) Testen	Internet-basierte Tests, Videokonferenz

Tabelle 5: Typische Bestandteile von Lernangeboten in hybriden Lernarrangements [KER02]

Ad 1.) Vortrag mit Diskussion

Vorträge oder reale Treffen bieten beim hybriden Lernmodell nicht nur die Möglichkeit Wissen zu vermitteln, sondern dienen in diesem Kontext dazu eine Einführung in Themen oder einen roten Faden durch komplexe Bereiche zu geben. Vor allem aber steht die soziale Komponente des Lernens hier im Vordergrund. Die Lernenden sollen motiviert werden, man soll einander kennen lernen, Gruppen sollen sich bilden und eine gemeinsame Kommunikationsbasis soll entstehen.

Ad 2.) Selbstlernaktivität

Selbstlernaktivitäten können eigenständig geplant werden. Eigenständiges Lernen setzt auch Selbstorganisations- und Motivationsfähigkeit voraus. Ablenkungen durch Freunde, Familie, etc. muss widerstanden werden. Es empfiehlt sich daher, seine Lernaktivitäten zu planen und diesen Plan auch seiner Umwelt zugänglich zu machen. Durch die Möglichkeit des Selbstlernens ergeben sich jedoch auch eine Reihe von Vorteilen. Es entsteht eine erhöhte Flexibilität in Zeit und Raum, die Geschwindigkeit und Intensität kann jederzeit frei gewählt werden. Auch stehen meist unterschiedliche und interaktive Medien zur Verfügung, die das Lernen und das Verarbeiten von Informationen erleichtern. Diese Medien können ausschließlich online (über das Internet) zur Verfügung stehen, aber auch offline etwa auf CD-ROM angeboten werden.

Ad 3.) Kooperatives Lernen

"Kooperatives Lernen wird als wesentliches Element nicht nur für den Aufbau kommunikativer, sondern auch kognitiver Kompetenzen aufgefasst. Zusammenarbeit in Gruppen fördert soziale Schlüsselqualifikationen wie Gruppenorganisation und Teamfähigkeit und der Austausch unterschiedlicher Perspektiven trägt wesentlich zu einer intensiven Auseinandersetzung mit Wissensgebieten bei, die einer besonderen geistigen Durchdringung bedürfen. Schließlich zeigen sich positive motivationale Effekte, die – über eine erhöhte Identifikation mit dem Lernangebot – mit einer erhöhten Lernintensität und Persistenz, sowie einer geringeren Abbruchquote einhergehen. [KER02]"

Ad 4.) Tutoriell betreutes Lernen

Tutoren und Coaches sorgen für eine individuelle Betreuung in der Distance Education. Sie können auf die Fragen der Lernenden eingehen, Hinweise oder Problemlösungen und auch persönliche Rückmeldungen zum Lernfortschritt geben. Aber auch das Fördern der Gruppenbildung und die Gruppenbetreuung gehören zum Aufgabengebiet des Coach und der Tutoren.

Ad 5.) Kommunikatives/soziales Lernen

Soziale Lernprozesse, gemeinsames sich gegenseitig unterstützendes Lernen muss bei Distance Education speziell gefördert werden. Bei Präsenzveranstaltungen stehen die

Lernenden per se miteinander im Kontakt. Bei webbasierten Kursen sind die Lernenden disloziert und daher muss die Kommunikation durch unterschiedliche synchrone und asynchrone Kommunikationsformen angeregt und gefördert werden.

Ad 6.) Beratung

Im Sinne einer Lernberatung sollen die Lernenden, die noch keine oder wenig Erfahrung mit Distance Education haben, angeleitet werden. Sie werden darüber informiert wie der Kurs als solches organisiert ist, was von ihnen erwartet wird, wie sie an das Medium, die Lerninhalte herangehen und mit anderen in Kontakt treten können. Die Lernenden sollen fortwährend unterstützt und somit die nötigen Fähigkeiten entwickelt bzw. verstärkt werden.

Ad 7.) Test, Zertifizierung

„Für alle Beteiligten ist die Qualitätskontrolle und –sicherung der Lernprozesse und Lernergebnisse von Bedeutung. Die Prüfung und Zertifizierung des Lernerfolges ist für das lernende Individuum nicht nur aus Sicht eines möglichen beruflichen Fortkommens wichtig, sondern auch für das Gefühl, ein definiertes Pensum bewältigt zu haben. Für den Bildungsanbieter [..] werden solche Informationen über Lernfortschritte noch wichtiger als bei konventionellen Maßnahmen, da der mehr oder weniger valide, unmittelbare Eindruck des Dozenten aus dem Unterrichtsgespräch fehlt. ([KER02])“

5.4.4. Kommunikation und Interaktion

Kommunikation ist ein wichtiger Bestandteil der Distance Education. Dabei wird nicht nur zwischen 1:1 und 1:N Kommunikation unterschieden, sondern vielmehr auch ob die Kommunikation synchron (zeitgleich) oder asynchron (zeitversetzt) erfolgt. In [COS] wird eine tabellarische Übersicht über mögliche internetgestützte Kommunikationsformen und deren jeweilige Anwendungsszenarien aufgezeigt.

		KOMMUNIKATIONS-FORM	BEISPIELSZENARIO
ASYNCHRON	1	E-Mail	Austausch von Fragen und Antworten Verteilung von Informationen und Dokumenten

	2	Nachrichtenforen (Newsgroups)	Diskussion Allgemeiner Gedankenaustausch
	3	Dateiaustausch	Einsendung/Abgabe/... von Übungen und Aufgaben
SYNCHRON	4	Chat	Kennenlernen zwischen den Lernenden/Lehrenden Allgemeiner Gedankenaustausch Diskussion von Fragen und Anregungen
	5	Whiteboard	Darlegung von Gedanken Verdeutlichung von Sach-verhalten Präsentation von Zusatz-informationen
	6	Application Sharing (gemeinsames Arbeiten an einem Dokument)	Verdeutlichung spezieller Sach-verhalte Präsentation von Teil-ergebnissen Gemeinsames Erstellen von Präsentationen
	7	Group-Browsing (gemeinschaftliches Surfen)	Präsentation von Inhalten Einführung in Aufgaben Erläuterung von Beispielen
	8	Audio-Konferenzen	Klassischer Vortrag Telefonat Fragen vieler Lernender an einen Lehrenden
	9	Video-Konferenzen	Klassischer Vortrag Präsentation von Laborübungen Einspielung von Videos Fragen an Expertenrunde
	10	MUDs (Multi User Dimension), Virtuelle Arbeitsräume	Gemeinschaftliche Diskussion Gemeinschaftliche Bearbeitung von Inhalten Gedankenaustausch

Tabelle 6: Internetgestützte Kommunikationsformen [COS]

Bei der Online-Kommunikation müssen aber auch Hemmschwellen überwunden werden. Die unterschiedliche soziale Präsenz ([MEI]) ist der wesentliche Unterschied in der Offline- und Online-Kommunikation. Online-Kommunikation wird oft als unpersönlicher beschrieben. Dieser Eindruck kann dadurch hervorgerufen werden, dass bei der schriftlichen Kommunikation die Menschen distanzierter sind, Hemmschwellen haben und Emotionen bei Schriftlichen unterdrückt werden. Die Erfahrungen in den Pilotstudien haben jedoch gezeigt, dass Hemmschwellen offenbar durch den ständigen Umgang mit schriftlicher Kommunikation (Email) nicht mehr so stark ausgeprägt sind, wie noch vor wenigen Jahren. Auch in [MEI] wird beschrieben, dass es häufig zu verbalen Ausbrüchen (flamings) kommt. Diese werden dort durch vier Faktoren erklärt, die ein ungehemmtes Verhalten beeinflussen können:

- „Eine fehlende Online-Identität: Je mehr Identität eine Person hat, desto mehr hält sie sich an diese Identität. Dies hat weniger ungehemmtes Verhalten zur Folge.

- Fehlende lokale Auswirkungen: Annahme, dass das Online-Verhalten keinen Einfluss auf das tägliche Leben hat führt zu mehr ungehemmten Verhalten. Eine Gegenmaßnahme wäre, das Online-Verhalten mit möglichst vielen Offline-Konsequenzen zu verknüpfen.

- Fehlende soziale Normen: Je mehr soziale Normen in einer Online-Gruppe vorkommen, desto eher kann das ungehemmte Verhalten sanktioniert werden.

- Zeit ist ein Schlüsselfaktor: Genügend Zeit für eine Arbeit ist wichtig, damit die Gruppe ihre sozialen Normen und Identitäten aufbauen kann.“

Wie in Kapitel 6.3.3 beschrieben wurde im Sommersemester 2002 die Lehrveranstaltung „Einführung in die Betriebssysteme“ an der Johannes Kepler Universität Linz als Distance Education Kurs abgehalten. Hier waren in der Diskussion um die neue Lernform verbale Ausbrüche zu beobachten. Zwar ist jeder Beitrag per se mit dem Namen des Verfassers versehen, dennoch waren einzelne Beiträge sehr ungezwungen formuliert und gingen teilweise „unter die Gürtellinie“. Auch war feststellbar, dass in den Face-to-Face Meetings Beiträge mit gleichem Inhalt gänzlich anders dargebracht wurden.

5.4.5. Feedback

Feedback spielt in der Distance Education eine zentrale Rolle. Dabei kann es sich sowohl um Feedback der Gruppe als auch um Feedback des Coach handeln. Feedback kann die Motivation der Lernenden erhöhen und den Lernerfolg verbessern. Promptes Feedback kann jedoch im Laufe des Kurses immer schwieriger für den Coach werden. Die Anfragen häufen sich, Abgaben treffen ein, Informationen werden verstärkt angefordert usw.

Um promptes Feedback garantieren zu können, sollten daher auch bestimmte „Spielregeln" festgelegt werden. So kann vereinbart werden, dass Fragen, die nicht mit dem Inhalt, sondern mit technischen Problemen (Übertragungsprobleme, Handhabung der Plattform, etc.) zu tun haben, direkt an den technischen Support zu richten sind.

Fragen zu speziellen inhaltlichen Bereichen sollen direkt an die jeweiligen dafür zur Verfügung stehenden Coaches oder Tutoren gehen.

Einfache Rückmeldungen, wie beispielsweise Eingangsbestätigungen von Abgaben, können von intelligenten Agenten übernommen werden.

Agenten können auch eigenständig, wenn auch teilweise limitiertes Feedback geben und somit eine wichtige Funktion im Feedback einnehmen.

Auch reagieren Agenten auf wiederholt falsche Antworten immer gleich und der Lernende muss sich durch sein technisches Gegenüber nicht blamiert vorkommen.

Der Coach kann sich nach der Filterung den Fragen zuwenden, die wirklich für ihn bestimmt sind. Dadurch hat er mehr Zeit für diese Anfragen und kann sich kniffligen Problemlösungen entsprechend widmen.

Zusätzlich darf nicht vergessen werden, dass schriftliches Feedback im allgemeinen mehr Zeit benötigt als verbales bzw. nonverbales Feedback. Oftmals kann durch bloßen Augenkontakt sehr schnell Feedback gegeben und im Notfall auch sanktioniert werden. Diese lern-begleitende und lernerfolgsfördernde Funktion muss in der Lernplattform durch entsprechendes Design und Funktionen ersetzt werden. Auch der zeitweise Einsatz von Videokonferenzen bzw. Face-to-Face Meetings können dabei von Vorteil sein.

5.5. Umsetzung von Distance Education

5.5.1. Wichtige Überlegungen für die Vorbereitung

In der Vorbereitung eines DE-Kurses sind wichtige Aspekte, wie Zielgruppenanalyse, Einsatz von Werkzeugen, Gestaltung des Lernmaterials, Lernorganisation, etc. zu bedenken. Diese müssen sich bewusst gemacht werden, damit die Durchführung reibungslos und vor allem erfolgreich sein kann. Als Vorbereitung sollte man sich daher einige Fragen stellen, wie:

Zielgruppenanalyse:

- Welche Zielgruppe möchte ich ansprechen?
- Was sind die Wünsche und Bedürfnisse der Zielgruppe?
- Welche Fähigkeiten (Vorwissen, Lerngewohnheiten, Akzeptanz, etc.) bringt die Zielgruppe mit?

Einsatz von Werkzeugen

- In welchen Kontext findet der Kurs statt (Schule, Universität, Firmen, etc.)?
- Welche Ressourcen stehen zur Verfügung (Finanzen, Personen, Einrichtungen, wie Räume, Hardware, Internetanbindungen, etc.)?
- Gibt es die Möglichkeit der Kooperation und Koordination mit anderen?
- Welche Plattform sollte gewählt werden hinsichtlich der Technologie (Plattformunabhängigkeit, Bedienerfreundlichkeit, etc.), Kosten, gewünschte Funktionalität, etc.?
- Welche Kosten entstehen durch den Betrieb?

Gestaltung des Lernmaterials

- Welche Funktionen sollen durch welche Art von Medium erfüllt werden (Steuerung Organisation, Wissenserwerb, Kommunikation, Motivation, Assessment, etc.)?

- Welche Lern-, Simulations- (Hands-on Experience) und Informationsmodule sollen bereitgestellt werden?
- Wie sollen Verarbeitungshilfen (Anknüpfen an das Vorwissen, Anregungen, Zusammenfassungen, etc.) umgesetzt werden?
- Welche Kosten bringt die Erstellung des Lehrmaterials mit sich?
- Welchen Nutzen bringt eine Standardisierung der Lerninhalte mit sich (Wiederverwendbarkeit, Qualität, Kosteneinsparungen, etc.)?

Didaktische Struktur - Lernorganisation

- Ist das Lernmaterial sequentiell, mit alternativen Pfaden, mit individuellen Lernpfad, etc. strukturiert?
- Welche Arten von Kommunikation (synchron, asynchron) werden unterstützt?
- Welche Kommunikationsformen (von, an und zwischen Einzelnen und/oder Gruppen, one-to-one, one-to-many) sind vorgesehen?
- Wie werden die Lernenden unterstützt (Tutoring, Coaching, Kooperatives Lernen, Selbstorganisation, etc.)?

Zusammenfassend haben die Planung einer Lernumgebung Rothenfluh und Aeppli ([ROT01]) graphisch wie folgt beschrieben:

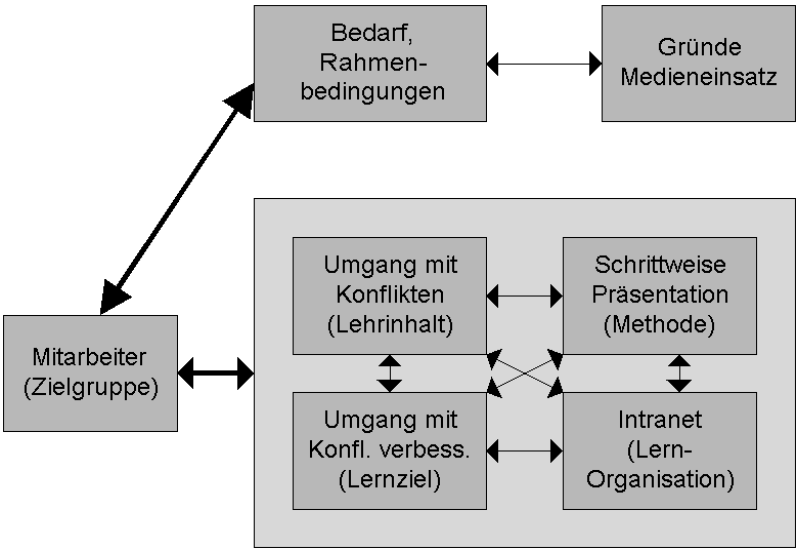


Abbildung 41: Planung einer Lernumgebung [ROT01]

5.5.2. Durchführung und Reflexion

Die Organisation und der Ablauf einer Distance Education Veranstaltung können sich auf variable Weise gestalten. Dies hängt von multiplen Faktoren, wie Zielgruppe, didaktische Vorlieben des Durchführenden, technischen Möglichkeiten, etc. ab.

Wird eine Distance Education Veranstaltung betreut durchgeführt, so hat sich gezeigt, dass ein konzertiertes Kick-Off-Meeting, Meilensteine und eine Abschlussveranstaltung als besonders wichtig und förderlich empfunden werden.

Das Kick-Off Meeting dient dazu, die Richtung aufzuzeigen in der der Kurs gehen soll, wie er organisiert und aufgebaut ist. Die Teilnehmer sollen mit ihren Ansprechpartnern, dem Coach und vor allem mit anderen Teilnehmern, bekannt werden. Falls Gruppenarbeiten vorgesehen sind, sollen sich die Gruppen im Kick-Off Meeting finden und die Gruppenmitglieder miteinander Kontakt knüpfen.

Danach sind im Selbstlernprozess Meilensteine für Einzelne und Gruppen ein wichtiger Bestandteil als Ziel und als Evaluationsmöglichkeit des eigenen Lernfortschritts.

In dieser Phase kann es auch zu Rollenwechseln oder Learning-by-Teaching kommen. Die Lernenden übernehmen für bestimmte Teilbereiche die Rolle des Coach. Somit sind sie gefordert sich intensiv mit dem Stoff auseinander zu setzen, indem sie auf Fragen ihrer Kollegen eingehen und Wege zur Lösung aufzeigen.

In der Abschlussveranstaltung sollen letzte noch offene Fragen geklärt werden. Vor allem aber soll eine Reflexion über den Ablauf, als Feedback für den Coach und die Gruppenmitglieder und zur Evaluierung und möglichen Verbesserung oder Erweiterung des Konzepts stattfinden.

Im gesamten Ablauf sind auch Präsenzphasen wichtig, in denen sich die Teilnehmer real treffen. Falls solche Präsenzphasen möglich sind, bieten sich dafür natürlich das Kick-Off Meeting und die Abschlussveranstaltung besonders an. Je nach Dauer des Kurses können auch dazwischen noch reale Treffen förderlich sein.

Besonders im schulischen Bereich, in welchem Distance Education als Erweiterung im Sinne eines virtuellen Klassenzimmers (virtual classroom) eingesetzt wird, sind solche Präsenz-

phasen problemlos durchführbar. Hier steht nicht so sehr das Kennenlernen, sondern das Miteinander beim Kick-Off Meeting im Vordergrund.

Ein möglicher beispielhafter Ablauf einer Distance Education Veranstaltung ist in [MIC01] graphisch umgesetzt und hier um wichtige Aspekte ergänzt dargestellt:

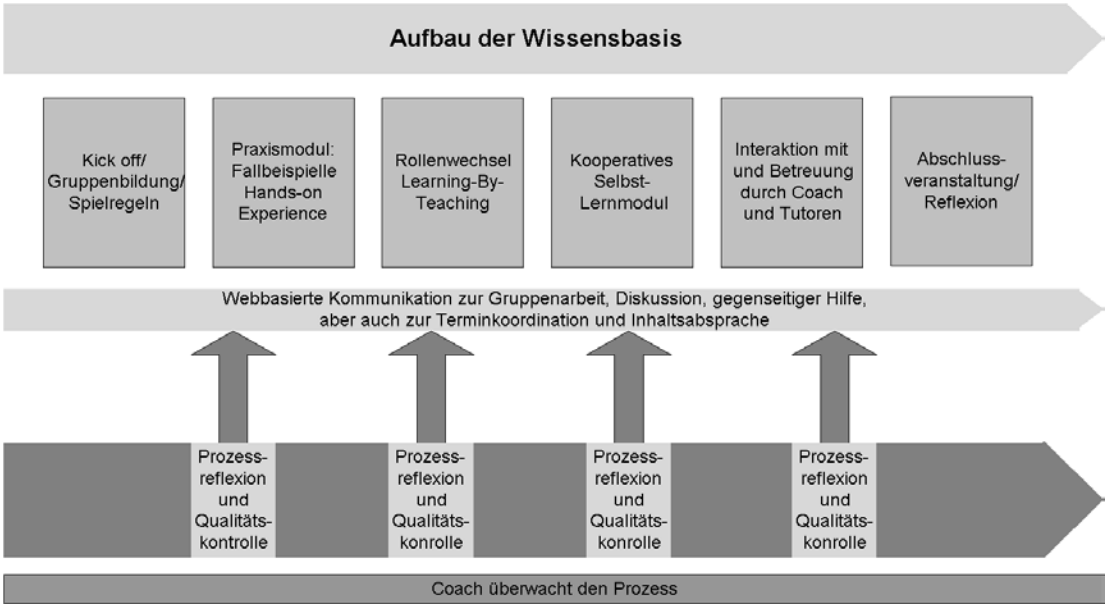


Abbildung 42: Möglicher Ablauf einer Distance Education Veranstaltung

6. PIAF- PROJEKT INFORMATIK IM FERNSTUDIUM

Im Rahmen des Projekts PIAF – Projekt Informatik als Fernstudium fanden mehrere Pilotstudien zum Thema Distance Teaching/Coaching/Learning statt. PIAF war eine Kooperation zwischen dem Institut für Informationsverarbeitung und Mikroprozessortechnik (FIM) und dem Zentrum für Fernstudien an der Johannes Kepler Universität Linz. Diese hatte das Ziel, eine Einführungsvorlesung aus Informatik (Propädeutikum aus Informatik) als webbasierte Distance Learning Lehrveranstaltung an der Universität Linz und an anderen Bildungseinrichtungen anzubieten. PIAF wurde vom Bundesministerium für Bildung, Wissenschaft und Kultur (bm:bwk) gefördert und findet momentan in PIAFII am FIM seine Fortsetzung. In PIAFII werden weitere Pilotstudien mit unterschiedlichen Lehrveranstaltungstypen, Lernmodellen und Anforderungen durchgeführt.

6.1. Begriffsbestimmung im Rahmen der Pilotstudien von PIAF

Die in dieser Arbeit verwendeten Begriffe, wie Distance Teaching, Distance Coaching und Distance Learning stehen jeweils für unterschiedliche Lernmodelle, die jedoch oftmals keine echte Abgrenzung zueinander haben, sondern schleichend ineinander übergehen. In unserem Sinne sind diese Modelle wie folgt zu verstehen:

6.1.1. Distance Teaching

Distance Teaching ist die Anleitung zum selbstgesteuerten Lernen, nicht Auge zu Auge, sondern mit den Mitteln der Informationstechnologie. Dabei steht der Lehrer im Mittelpunkt. Er ist die zentrale Figur und bietet den Lernenden einen Leitfaden durch die elektronischen Lerninhalte an.

6.1.2. Distance Learning

Beim Distance Learning liegt der Fokus auf den Lernenden. Der Lernende steht im Mittelpunkt des Geschehens. Bei Distance Learning, in dem Sinn, dass der Lernende selbständig ohne Anleitung und ohne Kontakt zu anderen lernt, besteht die Gefahr der sozialen Defizite. Der Lernende läuft Gefahr sich in seinem Lernprozess in Sackgassen zu verrennen bzw. bei bestimmten Fragestellungen alleine zu sein, nicht weiter zu kommen und somit an sich selbst

zu zweifeln beginnt. Als Lösung für dieses mangelnde soziale Umfeld bietet sich Distance Coaching als Lösungsweg an.

6.1.3. Distance Coaching

Distance Coaching kann unterschiedliches Aussehen haben:

- Zentralistische Lösung über einen Lehrer – tutoriell betreutes Lernen
- Kollaboratives Arbeiten in der Gruppe
- Hybride Form aus Coach und Gruppenarbeit

Zentralistische Lösung über einen Lehrer - tutoriell betreutes Lernen

Beim „reinen“ Distance Teaching fungieren ausschließlich die Lehrer bzw. Tutoren als Ansprechpartner für die Lernenden. Der Lehrer oder Tutoren helfen den Lernenden in schwierigen Fragen weiter, geben Hinweise oder zeigen zusätzliche Wege auf. Dies bedeutet aber einen enormen Aufwand für den Lehrer, da er in der Regel mehrere Lernende individuell zu betreuen hat. Auch können bei dieser Form die Lernenden wiederum eine passive Haltung im Lernprozess einnehmen. Wenn Fragen auftauchen wissen die Lernenden, dass der Lehrer diese für sie beantworten wird. Der Lernprozess findet nicht statt und wird in Richtung des Lehrers verlagert. Dadurch, dass sich die Lernenden nicht mehr aktiv selbst mit den Fragen auseinander setzen, selbst mögliche Lösungen ausprobieren, etc. geht der Verinnerlichungsprozess hier verloren, da die Lerninhalte nicht mehr eigenständig erarbeitet und daher auch nicht verfestigt werden. Die Nachhaltigkeit ist nicht mehr gegeben und das „Gelernte“ ist nur bedingt reproduzierbar, da keine selbständige Gewinnung des Wissens stattfindet.

Kollaboratives Arbeiten in der Gruppe

Beim kollaborativen Arbeiten in der Gruppe wird der Lernprozess zum einen vom Lernenden selbst und zum anderen von der Gruppe (mit)bestimmt. Dabei steht die Selbstorganisation mehr im Mittelpunkt als bei einer zentralistischen Lösung durch einen Coach. Verständnis-probleme, Problemlösungsprobleme, etc. können innerhalb einer Gruppe gelöst und über unterschiedliche Ansätze in der Gruppe diskutiert und argumentiert werden. Da die eigenen

Ideen den anderen Gruppenmitgliedern präsentiert werden müssen, kann man hier auch zeitweise in die Rolle des Lehrers schlüpfen und dies führt zu Learning-by-Teaching. Dadurch, dass man gezwungen ist seine eigenen Gedanken sprachlich auszuformulieren und zu präsentieren, findet eine Verinnerlichung statt, die stärker ist als die, die durch passives Konsumieren erlangt wird. Ideen, Argumente, etc. sprachlich auszudrücken hilft dabei diese zu verstehen. Wichtiger Nebeneffekt dieser Gruppenarbeit ist, dass soziale Schlüssel-qualifikationen, wie Gruppenorganisation und Teamfähigkeit gefördert werden ([KER02]). Obwohl die Gruppe viele Vorteile bietet, so kann jedoch nicht ausgeschlossen werden, dass in manchen Fällen es auch der Gruppe nicht möglich ist zu einer Lösung für ein bestimmtes Problem zu kommen. Eine größere Gruppe hat es hier leichter als eine Kleinere, allerdings je größer eine Gruppe ist, desto leichter fällt es auch „Trittbrettfahrer“ zu sein, sprich abermals ausschließlich passiv zu Konsumieren.

Hybride Form aus Coach und Gruppenarbeit

Die hybride Form ist eine Mischform, die dreistufig arbeitet. Bei den beiden eingangs angeführten Modellen handelt es sich jeweils um zweistufige Modelle. Einerseits der Lernende selbst und danach als Ansprechpartner der Lehrer, der als Coach fungiert oder die Gruppe. Bei der hybriden Form handelt es sich um ein dreistufiges Modell. Zuerst steht der Lernende, der durch lesen, experimentieren und forschen Wissen erlangt, danach ist die Gruppe die durch präsentieren, argumentieren und diskutieren für Nachhaltigkeit und Verinnerlichung sorgt und erst wenn diese beiden Stufen zu keiner bzw. keiner befriedigenden Lösung gekommen sind, ist es sinnvoll sich an den Coach zu wenden. Dieser hat dann die Möglichkeit die einzelnen Lösungsvorschläge zu beleuchten, sie zu hinterfragen und somit die Lernenden auf den richtigen Weg hinzuleiten.

Generell sollte die Rolle des Coach nicht unterschätzt werden. Zum einen wird es in den meisten Fällen keine 1:1 Kommunikation zwischen Lernenden und Coach geben, sondern die Kommunikation findet sternförmig statt. Der Coach hat die nicht triviale Aufgabe, die Lernenden in ihrem Lernprozess zu fördern. Dabei muss er vor allem zu Beginn die Gruppen-bildung und die Gruppenmitglieder in der Bearbeitung der Lernaufgaben unterstützen. Auch die Motivation der Lernenden und die Motivation der Gruppen ist eine wichtige Aufgabe des Coach in der Distance Education. Oft hört man auch die Kritik, dass es bei Distance

Education zu einer Vereinsamung der Lernenden kommt. Hier ist der Coach derjenige, der dies verhindert und als konstanter Ansprechpartner dient.

6.2. WeLearn – Web Environment for Learning

Für PIAF haben Fachleute und Studierende aus unterschiedlichen Bereichen (Informatik, Didaktik/Pädagogik und Psychologie) zusammengearbeitet und dabei „WeLearn“ als Grundlage und Grundpfeiler des Projekts erstellt.

WeLearn (Web Environment for Learning) ist ein Framework, das aus drei wesentlichen Teilen besteht:

- Einer Plattform für Distance Teaching/Coaching/Learning, die die Präsentation und Organisation von Lernmaterial, die Administration von Kursen, Kommunikation und Interaktion und auch Self-Assessment miteinschließt.
- Webbasierten Lehr- und Lerninhalten, wie „Propädeutikum aus Informatik“, „Einführung in die Betriebsysteme“, etc. und
- Beispiel-Settings oder Lernarrangements zur didaktischen Umsetzung von Lernmodellen, die Lehrende als Grundlage für ihre Kurse heranziehen oder adaptieren, aber auch individuelle Lernmodelle umsetzen können.

6.2.1. Lernplattform

Eine Lernplattform wird bei [CAV] definiert als “eine Softwareanwendung, die es den Benutzern ermöglicht, Informationen – meist zu einem abgegrenzten Themenkreis – über das Internet resp. Intranet zu erhalten und auszutauschen. Zum Austausch stehen Technologien wie Chat-Rooms und Groupware-Tools zur Verfügung.“

WeLearn ist eine offenes Framework, das universell einsetzbar ist. Es ist für jedermann einfach bedienbar und im Erscheinungsbild und in der Navigation leicht verständlich gehalten.

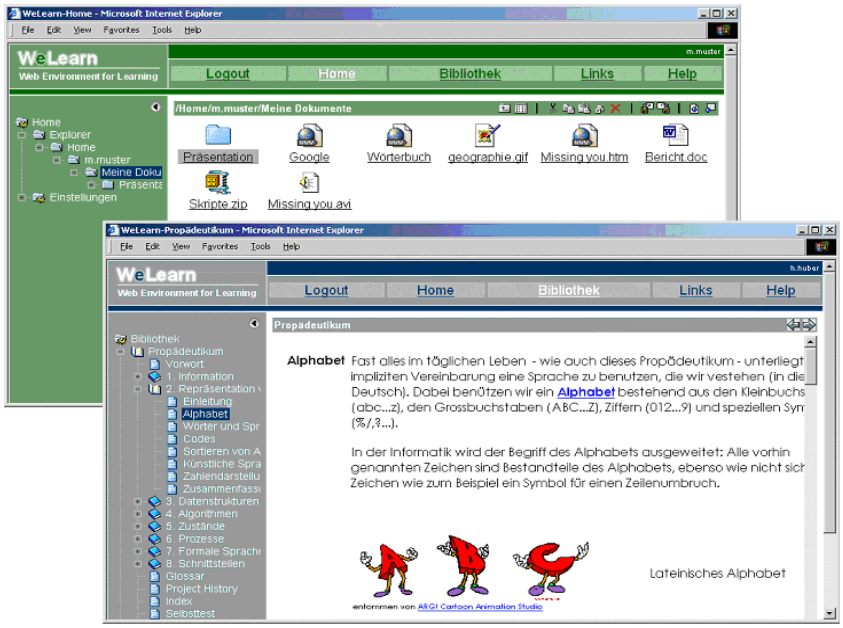


Abbildung 43: Erscheinungsbilder von WeLearn

WeLearn ist in JAVA implementiert und unterstützt Standards (etwa CPS, auf das später noch eingegangen wird) um eine Plattformunabhängigkeit, Wiederverwendbarkeit und Erweiterbarkeit zu gewährleisten. WeLearn kann im Sinne der GNU-Philosophie ([GNU]) kostenlos verwendet werden.

Besonders wurde darauf geachtet, WeLearn als Framework zu konzipieren ([DIV02b]). Dabei werden Basiskomponenten zur Verfügung gestellt, die je nach Anforderungen, Kurstyp, Zielgruppe, Ressourcen, etc. dynamisch arrangiert, erweitert oder verringert werden können. Dies ist jederzeit auch im laufenden Betrieb des Systems und während der Durchführung eines Kurses möglich.

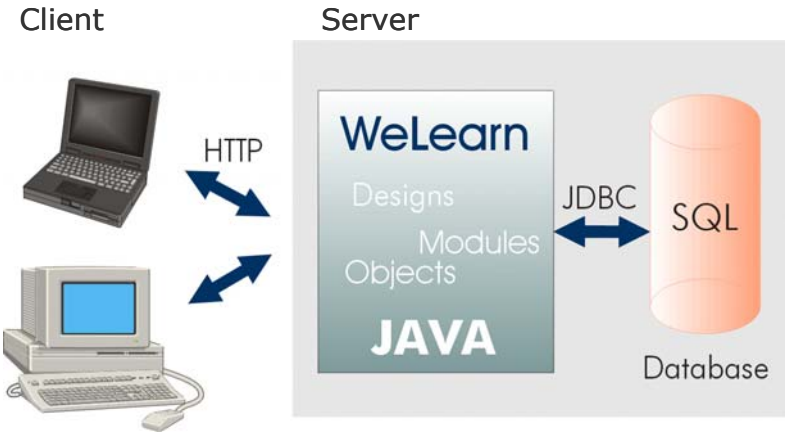


Abbildung 44: Technischer Aufbau von WeLearn [REI02]

Auf der Anwenderseite (Client) benötigt WeLearn ausschließlich einen Internet-Browser (Microsoft Internet Explorer, Netscape Navigator, etc.) und Darstellungswerkzeuge, wie Powerpoint Viewer, Acrobat Reader, usw. die zur Anzeige der Kursmaterialien vom Kursanbieter vorgegeben werden. Auf der Betreiberseite (Server) ist WeLearn wie bereits erwähnt in JAVA implementiert. Es gibt eine JDBC Verbindung zu einer SQL-Datenbank, in der alle Objekte (Verzeichnisse, Foren, Benutzer, etc.) des WeLearn-Systems gespeichert werden.

WeLearn stellt zahlreiche Funktionen zur Verfügung, darunter die Möglichkeit zur Kursorganisation, Unterstützung variabler Formate (Multimedia – HTML, Text, Präsentationen, Applets, Video, Audio, etc.), einfacher Datenaustausch, Bookmark-Verwaltung, einheitliche Navigation, Kommunikationstools und individueller Webspace. Auch wird kollaboratives Arbeiten durch „sharded folder“ und Personalisierung von Kursen unterstützt. Personalisierung kann dabei für Kurse, Gruppen und/oder einzelne Benutzer erfolgen. Dies wird vor allem durch ein striktes Rechtesystem ermöglicht, wodurch die Rechte aller Objekte (Dokumente, Verzeichnisse, Foren, etc.) spezifisch eingestellt werden können.

Die Kursinhalte können als Dateien in WeLearn präsentiert werden. Sie können aber auch im CPS-Format auf die Plattform gestellt werden. Das CPS-Format ist ein Standard des Instructional Management System ([IMSa]) des Global Learning Consortium. Dieser Standard wurde geschaffen um das reibungslose Zusammenwirken von Lerninhalten und Lernplattformen verschiedener Hersteller zu ermöglichen. Kurse, die im CPS-Format erstellt wurden, sind durch ein XML-Manifest repräsentiert, dass die Struktur und die Pfade zu den eingebundenen Dateien beschreibt. Das CPS-Format bietet dabei den Vorteil der Wiederverwendbarkeit und „Plattformunabhängigkeit“ und dient als Basis für einen internationalen Austausch von Lehrmaterialien.

Um diese Kurse auch offline anbieten und sie z.B. auf CD-ROM zur Verfügung stellen zu können, wurde von der WeLearn-Projektgruppe ein Viewer erstellt, der auf dem XML-Manifest des CPS-Formats basiert. Dadurch ist auch ein offline-Studium des Lernmaterials möglich.

Die WeLearn-Projektgruppe hat auch ein Authoringtool erstellt, das es Autoren von Kursinhalten ermöglicht, ohne spezifische Informatikkenntnisse eigenständig Kurse zu erstellen. Dabei können, wie bereits erwähnt, auch Multimediaelemente, wie Video, Audio, Animationen etc. problemlos eingebunden werden.

Mehr Informationen über WeLearn und dessen Aufbau sind auch in ([DIV02a], [DIV02b], [MUE02a], [MUE02b]) zu finden und unter [FIM] kann ein Test-Account angelegt werden.

6.2.2. Eingebettetes Kursmaterial

In der WeLearn-Plattform ist bereits zahlreiches Kursmaterial, wie etwa „Propädeutikum aus Informatik“, „Einführung in die Betriebssysteme“, „Java-Programmierung“, etc. eingebettet, das speziell für Distance Education aufbereitet wurde.

Als Beispiel sei hier kurz das Lernmaterial für „Propädeutikum aus Informatik“ beschrieben. Dieses Lernmaterial ist multimedial aufbereitet. Unterschiedliche Aspekte sind in unterschiedlichen Medien umgesetzt, wobei sie sich vielfach überlappen, einander ergänzen bzw. detaillierte Informationen bieten und somit im Lernprozess nachhaltiger wirken. So liegt ein Langtext in Deutsch und Englisch vor. Dieser Text ist aber auch als animierte Folienpräsentation aufbereitet und somit visuell umgesetzt. Zusätzlich gibt es eine HTML-Version, die Text, Graphiken und Java-Applets umfasst. Die Applets dienen dabei zur Hands-on Experience. Die Lernenden können bestimmte Beispiele ablaufen lassen, aber auch selbst ausprobieren und die Inhalte somit spielerisch erlernen. Zusätzlich gibt es zu den einzelnen Themenbereichen Self-Assessmentmöglichkeiten. Die Lernenden können Fragen beantworten, Lückentexte vervollständigen, Multiple Choice Fragen oder Kreuzworträtsel lösen und somit ihren Lernfortschritt überprüfen.

In WeLearn wird auch ein Studyguide zum Thema „Wie lernt man“ bereitgestellt, um einen Leitfaden zum selbstgesteuerten Lernen zu geben. Auch werden unterschiedliche Lerntypen und –techniken aufgezeigt, die als individuelle Unterstützung der Lernenden dienen sollen.

Durch die Kombination von Elementen unterschiedlicher methodischer und medialer Aufbereitung soll die Qualität und die Effizienz des Lernangebotes gesteigert werden.

Es ist geplant „Propädeutikum aus Informatik“ gänzlich öffentlich zugänglich zu machen. Lernende hätten dadurch die Möglichkeit sich in der Welt der Informatik zu orientieren, eigenständig Erfahrungen zu sammeln und Inhalte auszuprobieren. Maturanten könnten somit bereits im Vorfeld bei der Entscheidungsfindung zu einem Studium heraus finden, ob und inwieweit eine technische bzw. informatikorientierte Ausbildung für sie in Frage kommt und ihr Interesse weckt.

6.2.3. Lernarrangements – Settings

Nahezu so vielseitig, wie das Lernmaterial gestaltet werden kann, so kann auch das Lernarrangement variabel sein. Dabei nehmen die Art des Kurses, die Art des Lernmaterials, die Zielgruppe, didaktische Vorlieben des Durchführenden, Ressourcen etc. Einfluss auf das zum Einsatz kommende Lernarrangement oder Setting.

Hier sei kurz ein beispielhaftes Setting für ein hybrides WeLearn-Lernmodell beschrieben:

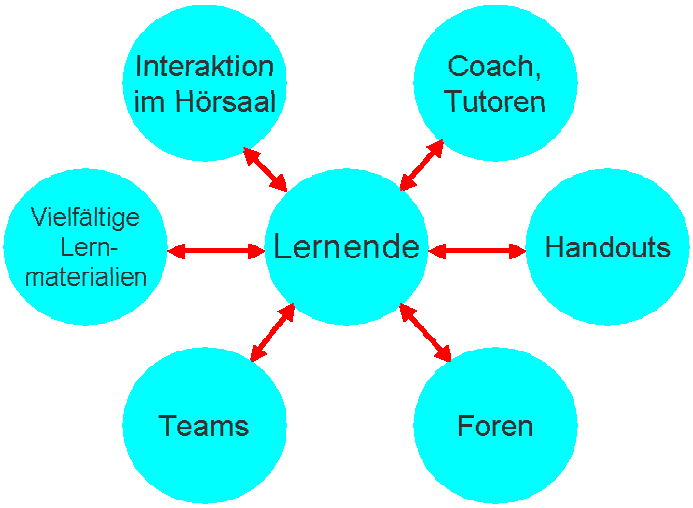


Abbildung 45: Typisches WeLearn-Setting für ein hybrides Lernmodell [REI02]

Interaktion im Hörsaal: Zu bestimmten Zeitpunkten finden reale Treffen statt. Dabei werden Fragen, die bisher aufgetreten sind und noch nicht geklärt werden konnten behandelt und im Plenum diskutiert. Zusätzlich wird durch den Lehrenden ein roter Faden durch weitere Themengebiete gegeben und auf wichtige Aspekte dieser Themen hingewiesen.

Vielfältige Lernmaterialien: Die Lernmaterialien sind teils überlappend, teils vertiefend, unterschiedlich medial aufbereitet, sodass unterschiedliche Lerntypen gleiche Unterstützung finden. Zusätzlich sind Elemente, wie Hands-on Experience und Self-Assessment zur Förderung des Lernfortschritts und Festigung des Gelernten eingebaut.

Teams: Die Lernenden bilden Teams. Die Gruppenmitglieder können sich dabei im Lernfortschritt gegenseitig unterstützen bzw. fungieren die einzelnen Gruppen bei unterschiedlichen Themen als Coaches für die anderen Gruppen und praktizieren dabei Learning-by-Teaching.

Foren: Den einzelnen Gruppen, aber auch den Lernenden als Gesamtheit stehen Diskussionsforen zur Verfügung, um zu fragen, zu diskutieren, zu antworten und auch um einen Eindruck über den Lernfortschritt der anderen zu bekommen. Auch werden Informationsforen eingesetzt, die vorwiegend zur Kursorganisation und –koordination dienen.

Handouts: Handouts in gedruckter Form werden angeboten, um Lernende zu Unterstützen, die auch arbeiten möchten, wenn sie partiell keinen Online-Zugriff haben.

Coach/Tutoren: Der Coach oder auch Tutoren wirken unterstützend. Sie motivieren, geben Hilfe und zeigen unterschiedliche Lösungswege auf.

Alle diese Elemente, bis auf die Interaktion im Hörsaal und die Handouts, werden jeweils in die WeLearn-Plattform eingebettet.

Die jeweiligen Settings für unterschiedliche Kurse, Anforderungen, Zielgruppen, etc. werden in 6.3 Pilotstudien von PIAF aufgezeigt.

6.3. Pilotstudien von PIAF

Hier soll ein kurzer Überblick über einige Pilotstudien gegeben werden, um einen Eindruck zu vermitteln in welchen Bereichen (Schule, Universität, Erwachsenenfort- und weiterbildung) unterschiedliche Kurse mit WeLearn-Unterstützung abgehalten wurden und auch um kurz deren unterschiedliche didaktische Modelle zu skizzieren (siehe auch [MUE02a], [MUE02b]).

6.3.1. Lehreraus- und fortbildung

Im Rahmen der Lehrerfortbildung des Pädagogischen Instituts wurde der Kurs „Propädeutikum aus Informatik“ für AHS-Lehrer abgehalten.

In der WeLearn-Plattform wurde dabei das Lernmaterial nach und nach zur Verfügung gestellt und diente als Ergänzung der traditionellen Handouts. Die Kursdauer war für eine Woche angesetzt. Um diese kurze Zeit effizient nutzen zu können, wurden zwei Lernmodelle miteinander verknüpft. Zum einen wurde ein hybrides Lernmodell eingesetzt, bei dem vormittags ein roter Faden durch die einzelnen Themen gegeben und Schwieriges erläutert wurde. Zum anderen wurde auch auf „Learning-by-Teaching“ als Lernmodell zurück-

gegriffen. Dies hat sich auch als sehr wirkungsvoll herausgestellt. Quasi als Widerspiegelung der Klassenzimmer bildeten die Lernenden Gruppen. Die einzelnen Gruppen hatten nachmittags die Aufgabe den anderen Gruppen Aufgaben zu stellen und auf deren Fragen zum Stoff einzugehen. Die Gruppen übernahmen dabei partiell die Rolle des Coach gegenüber anderen Gruppen. Die Gruppenmitglieder waren daher veranlasst sich eingehend mit den Lerninhalten zu beschäftigen. Dabei lernten sie auch die Sichtweisen anderer kennen. WeLearn und das dort eingebettete Lernmaterial war in dieser Zeit ständig abrufbar. Jeweils abends wurde wieder im Plenum diskutiert und der Lehrende zeigte ergänzende Problemlösungsmöglichkeiten auf.

Dieses Lernarrangement hat sich als besonders intensiv im Lernerfolg und in der Nachhaltigkeit erwiesen.

6.3.2. Propädeutikum aus Informatik an der JKU Linz

Die Lehrveranstaltung „Propädeutikum aus Informatik“ findet an der Johannes Kepler Universität Linz im ersten Semester des Studienplans für Informatik statt. Diese wurde letztes Semester mit dem Großteil der Studenten konventionell mit einer Woche Vorlesung, drei Wochen Vorbereitungszeit und abschließender Klausur abgehalten. Mit einer kleineren Gruppe von freiwilligen Studenten wurde „Propädeutikum aus Informatik“ als Distance Education Kurs abgehalten. Dieser dauerte ebenfalls vier Wochen, jedoch gab es außer einem Kick-Off Meeting und einer Abschlussveranstaltung keine realen Treffen und der Lehrstoff wurde ausschließlich über das WeLearn-System [DIV02b] vermittelt.

Dabei standen den Studenten das bereits in Kapitel 6.2.2 erwähnte multimedial aufbereitete Lernmaterial und Coaches zur Verfügung. Auch haben die Lernenden fünf Gruppen gebildet. Diese Gruppen hatten unter anderem die Aufgabe gemeinsam Fragen für die anderen Gruppen zu finden, eine gemeinsame Antwort auf die Fragen der anderen Gruppen zu finden und in die Foren zu posten. Letztendlich sollten die Gruppenmitglieder die gegebenen Antworten korrigieren und diskutieren. Learning-By-Teaching wurde von den Lernenden in einer abschließenden Evaluierung als besonders positiv empfunden. Generell wurde die neue Lernform gut angenommen und die Klausurergebnisse waren entsprechend gut.

Der genaue Ablauf und die Ergebnisse dieser Lehrveranstaltung sind detailliert in [MUE02b] beschrieben.

6.3.3. Einführung in die Betriebssysteme an der JKU Linz

Die Lehrveranstaltung „Einführung in die Betriebssysteme“ ist eine Pflichtlehrveranstaltung im Studienplan der Informatik der JKU Linz. Diese wurde im Sommersemester 2002 erstmals als hybride Distance Education Lehrveranstaltung abgehalten. Dabei wurde das in Kapitel 6.2.3 beschriebene Lernsetting verwendet. Einmal wöchentlich hat ein Coach im Hörsaal offene Fragen diskutiert und einen roten Faden durch das nächste Thema gegeben. Die Studenten waren in fünf Gruppen eingeteilt und wurden im Lernprozess durch jeweils einen Gruppentutor und die Coaches unterstützt. Eine mögliche Gruppendiskussion und auch die Unterstützung durch die Tutoren wurde allerdings von den Studenten schwächer als erwartet angenommen. In ein Gruppenforum wurde nahezu keine Beiträge gepostet. Vielfach wurde auch nicht über den Lernstoff, sondern über die neue Lernform und über WeLearn diskutiert. Ein Chattermin mit dem Coach und allen Tutoren wurde hingegen sehr gut angenommen und auch das Loginmonitoring hat ergeben, dass die Studenten zwar häufig im System eingeloggt waren, aber nicht mit den anderen kommuniziert haben. Auch war zu beobachten, dass die Studenten auf andere Kommunikationskanäle, wie reale Treffen und ICQ (einem synchronen Chatwerkzeug) ausweichen.

6.3.4. Weitere Lehrveranstaltungen, abgehalten an der JKU Linz

An der Johannes Kepler Universität wurden noch weitere Lehrveranstaltungen unter Einsatz von WeLearn abgehalten. Darunter waren ein Programmierpraktikum und Seminare, in denen WeLearn hauptsächlich unterstützend zur Kursorganisation und als Kommunikationsplattform zum Einsatz kam. Beim Programmierpraktikum fand nicht nur die Kommunikation über WeLearn statt, sondern der gesamte Bereich der Übungsangaben und vor allem der Übungsabgaben durch die Studenten wurde ausschließlich über WeLearn abgewickelt. Dabei stand jedem Studenten ein individueller Abgabeordner zur Verfügung auf dem nur der jeweilige Student und der Coach Zugriff hatten. Die Möglichkeit des Fileup- und downloads von WeLearn spielte dabei eine zentrale Rolle.

6.3.5. Virtuelle Kommunikation an den Universitäten von Zürich, München und Graz

An der Universität Zürich wurde im Wintersemester 2001/02 ein Kurs “Virtuelle Kommunikation” organisiert und abgehalten. An diesen Kurs haben sich auch die

Universitäten München und Graz beteiligt. Studenten dieser Universitäten hatten dabei die Aufgabe gemeinsam an bestimmten Themen zu arbeiten und gemeinsame Seminararbeiten und Präsentationen zu gestalten. Die WeLearn-Plattform war dabei die Schlüsseltechnologie, die diese distribuierte Zusammenarbeit und das kollaborative Arbeiten ermöglichte.

6.3.6. Propädeutikum aus Informatik für Schüler (im Alter von 14 bis 16) an zwei verschiedenen höheren Schulen

In Notebook-Klassen in der AHS kann WeLearn generell zum Einsatz kommen. Entsprechende Pilotstudien wurden dazu bereits im Schuljahr 2001 durchgeführt. Dabei wurden jeweils Teile von „Propädeutikum aus Informatik“ im Rahmen des Informatikunterrichts an der HBLA Steyr und der AHS Vöcklabruck durchgenommen. Die Lehrer waren dabei während der Unterrichtsstunden in den Klassen präsent. Begleitend dazu wurde WeLearn zur Einbettung des Lehrmaterials, als Werkzeug zur Diskussion und als Basis für die Hausaufgaben (Angaben- und Abgabenverwaltung) genützt.

6.3.7. Lehrgang akademischen Charakters an der FH Vorarlberg für Berufstätige zur Weiterbildung

An der Fachhochschule Vorarlberg in Dornbirn findet seit 2001 ein Lehrgang Akademischen Charakters zum Thema „IT-Netzwerkmanagement” statt. In der Ausschreibung für diesen Lehrgang heißt es explizit, dass nach Maßgabe der didaktischen Möglichkeiten auch Elemente des Fernunterrichts und des Distance Coaching zur Verwendung kommen sollen. Dies ist vor allem auch deshalb notwendig und sinnvoll, weil die Lehrgangsteilnehmer berufstätig sind und disloziert im Raum Vorarlberg wohnen und arbeiten. Die Teilnehmer sind Erwachsene im Alter zwischen 25 und 40 Jahren, die in verschiedenartigen Unternehmen angestellt sind und ihre Kenntnisse und Fähigkeiten im Bereich der Informatik und vor allem in den Bereichen Netzwerkadministration und JAVA-Programmierung erweitern und verbessern möchten. Aus diesem Grund werden dabei Kurse, wie Propädeutikum aus Informatik, Projektmanagement, Netzwerkmanagement, Einführung in die Betriebssysteme, JAVA-Programmierung, etc. im hybriden Lernmodell abgehalten. Regulär an Wochenenden wird vor Ort mit den Teilnehmern gearbeitet. Dazu begleitend und ergänzend werden bereits im Vorfeld Informationen zum Selbststudium zur Verfügung gestellt und für die Begleitung wurde die Möglichkeit zur Diskussion geschaffen. Diese multiplen Diskussions-

möglichkeiten, wie Diskussionsforen zu spezifischen Themen, Gruppen- und Informationsforen, aber auch quasi synchron abgehaltene Chattermine werden enthusiastisch und intensiv genützt. Generell wird diese Form des Lernens sehr begrüßt und stößt nicht nur auf reges Interesse, sondern wird als Bereicherung empfunden.

Zusätzlich plant die FH Vorarlberg eine Ausweitung ihrer Ausbildungsprogramme im Bereich IT, wofür durch den Einsatz von WeLearn bereits eine Basis geschaffen wurde.

7. EINSATZ VON AGENTEN IM BEREICH DER DISTANCE EDUCATION – AGENTEN-ORIENTIERTE WISSENSPLATTFORM (KNOWLEDGE PLATFORM FOR AGENTS - KNOWPAG)

7.1. Vision

Einerseits ist das Bildungsangebot im Distance Education Bereich zahlreich. Noch zahlreicher sind die Informationen, die im Internet zu finden sind. Viele sind auch an diesen Bildungsangeboten interessiert, die Suche danach ist aber oftmals langwierig und es erfordert viel Aufwand einen Kurs im Internet zu finden, der exakt mit den Ansprüchen des Suchenden übereinstimmt.

Andererseits können intelligente Agenten hilfreich zur Seite stehen können. Um solche intelligente Agenten auch sinnvoll einsetzen zu können, bedarf es aber vor allem einer strukturierten Aufbereitung der Informationen und einer für Agenten und Benutzer geeigneten Umgebung.

Auch können Agenten wertvolle Arbeit bei der Kursorganisation und –durchführung leisten. Durch die Integration von Agenten in DE-Kurse und DE-Plattformen ergeben sich eine Reihe von Vorteilen für unterschiedliche Benutzergruppen. Beispielsweise können Agenten in einer DE-Plattform als Notifikationssystem eingesetzt werden. Kursteilnehmer und –leitende können durch Agenten verständigt werden, wenn neue Dokumente, Nachrichten oder Beiträge in die Plattform gestellt werden. Dadurch ersparen sie sich regelmäßiges zeitraubendes Durchforsten der Foren, Verzeichnisse, etc. Agenten können auch für Wartungs- und Archivierungsarbeiten eingesetzt werden. Offene Kurs- und Kursveranstaltungsgebühren können mittels Agenten abgefragt und auch bezahlt werden. Die Personalisierung der Lernumgebung kann durch den Einsatz von Agenten übernommen und vereinfacht werden. Agenten können aber auch dazu eingesetzt werden um Kurse jeder Art aufzufinden, um Kurse zielgruppengerecht zu präsentieren bzw. gleichsprachliche Kurselemente aufzufinden. Auch können Agenten Kursorganisatoren helfen bereits bestehendes Material zur Kursorganisation aufzufinden und Kurse aus bestehenden Lernmaterialien zu assemblieren.

Im folgenden sollen zunächst mögliche Einsatzgebiete für Agenten im Bereich der Distance Education aufgezeigt werden. Es wurde ein Prozessmodell für DE-Kurse entwickelt, anhand dessen unter anderen der Einsatz von Agenten aufgezeigt wird. Das Prozessmodell stellt die Abläufe und Aufgaben bei der Kursvorbereitung, -durchführung und –nachbereitung dar. Darin werden die spezifischen Aufgabengebiete in Rollen unterschieden und beleuchtet, welche Aufgaben in welcher Abfolge durchzuführen sind. Auch unterschiedliche Kursmodelle (reiner DE-Kurs, hybrides Modell, traditioneller Kurs mit DE-Elementen) werden charakterisiert und beschrieben. Das Prozessmodell beinhaltet eine detaillierte Beschreibung der komplexen Abläufe eines DE-Kurses und stellt somit eine Anleitung oder ein Handbuch für Kursorganisatoren, Coaches, etc. dar. Zusätzlich wird im Prozessmodell aufgezeigt, welche Aufgaben von Agenten übernommen werden könnten und somit eine Arbeitserleichterung für den jeweiligen Rolleninhaber möglich wäre.

Im zweiten Teil wird konkret auf Agenten zur Kurssuche, Kursassemblierung, etc. eingegangen. Dafür wurde eine agentenorientierte Wissensplattform geschaffen, die die Basis für die inhaltliche, aber auch strukturelle Beschreibung von Kursen bietet. Auf dieser Basis lassen sich auch Datenbankschemata und Agententypen ableiten. Letztlich können aufgrund der Agententypen eine beliebige Anzahl von Agenten gleicher Funktionalität geschaffen werden, die von Benutzern (Kursorganisatoren, Coaches, Lernende, etc.) eingesetzt werden können. Einige Agenten laufen dabei unsichtbar und abgeschirmt von Benutzern. Andere Agenten kommunizieren mit ihren Benutzern. Für diesen konkreten Einsatz von Agenten wurde ein Prototyp für ein Web-Interface geschaffen, dass die Benutzer bei der Bedienung der Agenten unterstützt. Die Benutzer selbst merken durch das Web-Interface nicht, dass in Wirklichkeit mehrere Systeme im Einsatz sind, sie arbeiten weiter in ihrem gewohnten System und das Web-Interface fungiert als Mittelschicht zwischen den Benutzern und den Systemen (Lernplattform, Agentensysteme).

7.2. Einsatzmöglichkeiten von Agenten in Distance Education Kursen

Hier wird die Vorbereitung, Durchführung und Nachbereitung eines Distance Education Kurses als Prozessmodell dargestellt. Dabei wird auf die Prozesse, Aufgaben und Rollenverteilung innerhalb eines DE-Kursablaufs von der Konzeption bis zur Nachbereitung eingegangen. Das Prozessmodell wurde dabei so universell, wie möglich gehalten. Es bietet

eine prozess-orientierte Gliederung eines DE-Kurses und entstand vor dem Hintergrund der Erfahrungen, die aus den in Kapitel 6.3 erwähnten Pilotstudien gewonnen wurden.

Besonders wurde im Modell auf die Rolle und auf die zahlreichen Einsatzmöglichkeiten der Agenten eingegangen. Dafür wurden Rollen, wie Kursorganisator/Agent, Coach/Agent, etc. spezifiziert, die ausdrücken sollen, dass die für diese Rollen beschriebenen Aufgaben gleichermaßen von Personen, aber auch von Agenten durchgeführt werden könnten.

7.2.1. Prozessmodell: Distance Education Kurs

7.2.1.1. Beschreibung der Zeichnungselemente

Bei der Modellierung und der graphischen Darstellung des Prozessmodells und der Prozesse wurde auf AENEIS (Analyse, Entwicklung und Nutzung eingebetteter Informationssysteme, [AEN], LAN94]) zurückgegriffen.

Leider wird jedoch durch den Transfer der Diagramme von AENEIS zu Microsoft® Word® der Text in manchen Feldern abgeschnitten. Deshalb werden bei allen Prozessen die einzelnen Prozessschritte zuerst textuell und anschließend graphisch dargestellt. Bei der textuellen Angabe werden die Prozessschritte der Reihenfolge ihres Auftretens nach angeführt, angegeben, ob diese parallel oder optional ausgeführt werden und in Klammern wird die ausführende Organisationseinheit mitangegeben.

Alle Prozessschritte (Aktivitäten) sind durch Rechtecke dargestellt (Startaktivitäten durch gestrichelte Umrahmungen). In diesen ist immer der Name der Aufgabe, der ausführenden Stelle (Organisationseinheit oder Rolle) und der eines Mitarbeiters (als die nötige Qualifikationsanforderung zur fach- und sachgerechten Durchführung der Aufgabe) angeführt.

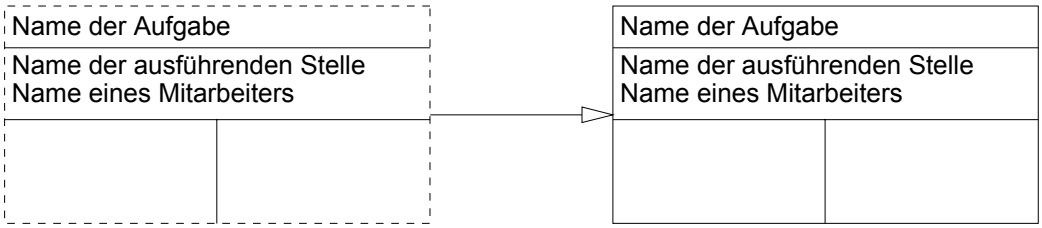


Abbildung 46: Darstellung von Aktivitäten

Bei den Aktivitäten sind rechts in den Kästchen immer Symbole eingeblendet. Diese haben folgende Bedeutung:

Die Lupe kennzeichnet, dass der Arbeitsschritt noch verfeinert wird.

Eingangsschnittstelle: Liste der erhaltenen Informationen

Ausgangsschnittstelle: Anzeige des Aufgabenträgers, an den Informationen weitergeleitet werden

Verbal beschriebener Arbeitsschritt

Anweisung einer Aufgabe des gleichen oder eines anderen Aufgabenträgers

Gruppierung von Arbeitsschritten

Gruppe von Arbeitsschritten (nach einer Gruppierung)

Wiederholte Ausführung von Arbeitsschritten

Weiters werden stellenweise verschiedene Teilungsarten verwendet. Diese können sein:

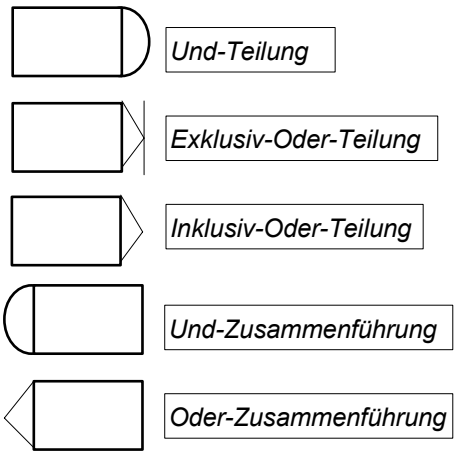


Abbildung 47: Mögliche Symbole in Prozessdiagrammen

Und-Teilung: Jeder der folgenden Zweige wird nach Ausführen der Teilungsaktivität parallel ausgeführt.

Exklusiv-Oder-Teilung: Genau einer der folgenden Zweige wird ausgeführt.

Inklusiv-Oder-Teilung: Keiner, einer oder mehrere der folgenden Zweige werden ausgeführt.

Und-Zusammenführung: Aktivität hat mehrere Vorgänge, wobei jeder ausgeführt wird und in diese Aktivität einmündet.

Oder-Zusammenführung: Aktivität hat mehrere Vorgänger, wobei optional nur einer ausgeführt wurde.

7.2.1.2. Das Prozessmodell

Das Prozessmodelle gliedert sich zuerst in 2 Teile: die Geschäftsprozesse und die Diagramme.

Die Geschäftsprozesse wiederum gliedern sich in drei Teile:

- Teil 1: Kursvorbereitung
- Teil2: Kursdurchführung und
- Teil 3: Kursnachbereitung/Improvement

Teil 1 – die Kursvorbereitung umfasst folgende Prozesse:

- Kursvorbereitung
- Entwicklung, Wahl, Aufbereiten des Lernmaterials und
- Umsetzung der Lernlandschaft

Teil 2 – die Kursdurchführung gliedert sich in folgende Prozesse:

- Kursstart
- Kick-Off
- DE-Kursdurchführung
- Kurs ist reiner Distance Education Kurs (ohne reale Treffen)
- Kurs ist ein DE-Kurs im hybriden Modell (teilweise online, teilweise reale Treffen)
- Kurs ist traditionell geführt mit DE Online-Unterstützung
- DE-Kurs Abschluss
- Abschlussveranstaltung
- Prüfung/Zertifizierung

Teil 3 – die Kursnachbereitung/Improvement besteht aus dem Prozess Kursnachbereitung.

Die Diagramme sind unterteilt in Organigramm und Informationssysteme, wobei in Organigramm alle Rollen und ihre Aufgaben beschrieben werden. In Informationssysteme wiederum wird auf die Funktionen der Plattform und auf unterschiedliche Agententypen eingegangen.

Geschäftsprozesse

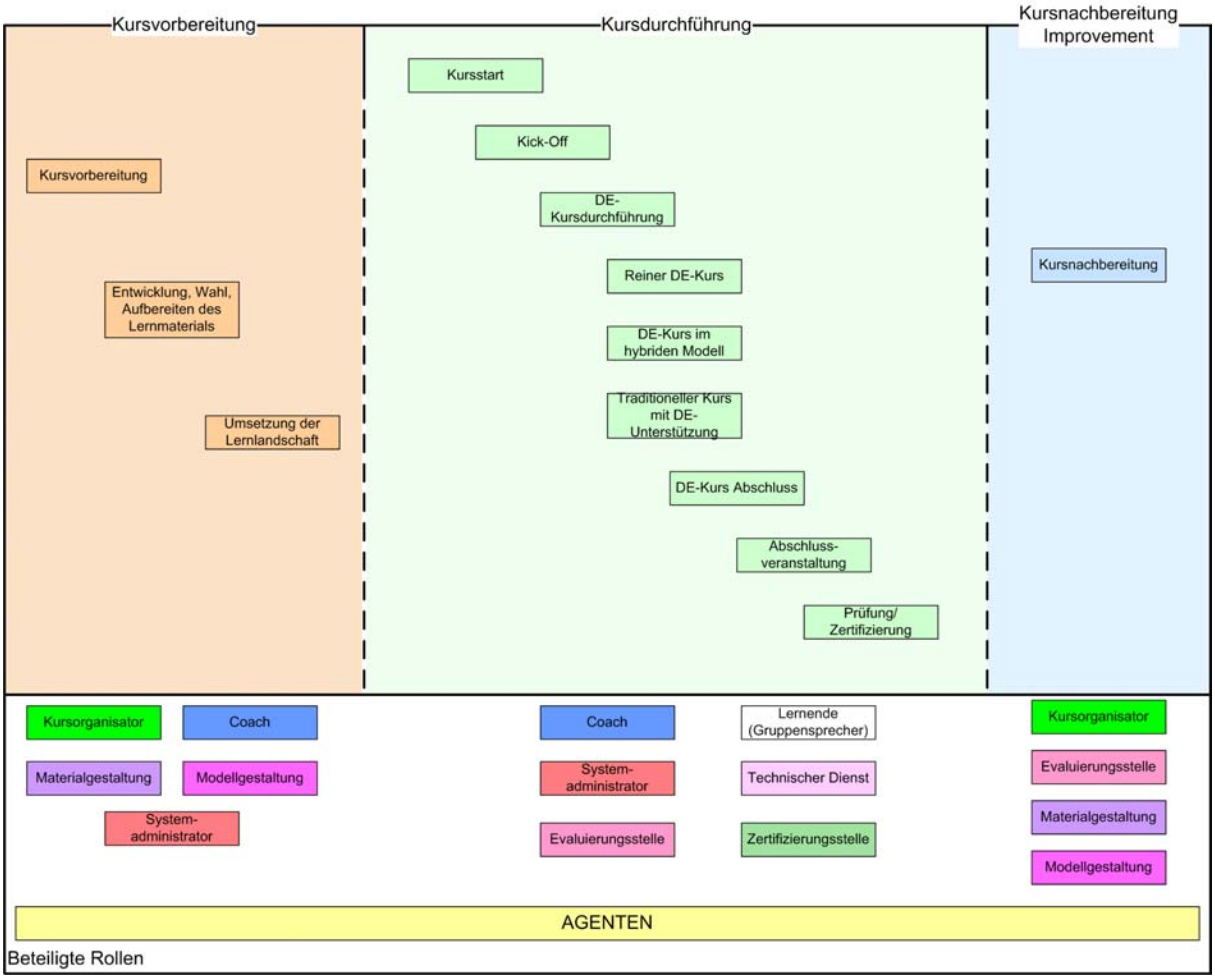


Abbildung 48: Übersicht über das Prozessmodell: Distance Education Kurs

1 Teil 1: Kursvorbereitung

Die Kursvorbereitung umfasst alle Vorarbeiten in den Bereichen Klärung der Rahmenbedingungen, Wahl der Mitarbeiter, der Werkzeuge, Wahl und Entwicklung des Lernmaterials und des didaktischen Modells, etc. Darüber hinaus wird die Lernlandschaft eingerichtet, sodass alle Unterlagen, die Lernumgebung, etc. vor Kursbeginn fertig erstellt sind.

1.1. Kursvorbereitung

Der Prozess „Kursvorbereitung“ umfasst folgende Prozessschritte:

1 Positive Entscheidung einen Distance Education Kurs anzubieten (Kursorganisator)
2 Rahmenbedingungen/Ressourcen (Kursorganisator/Agent)
3 Coach bestimmen (Kursorganisator/Agent)
4 Übergabe an Coach (Kursorganisator/Agent)
5 Wahl der Wahlzeuge (Kursorganisator/Agent)
6 Zielgruppenanalyse (Kursorganisator/Agent)
7 Didaktisches Modell (Modellgestaltung)
8 Initiierung des Prozesses „Entwicklung, Wahl, Aufbereiten des Lernmaterials“

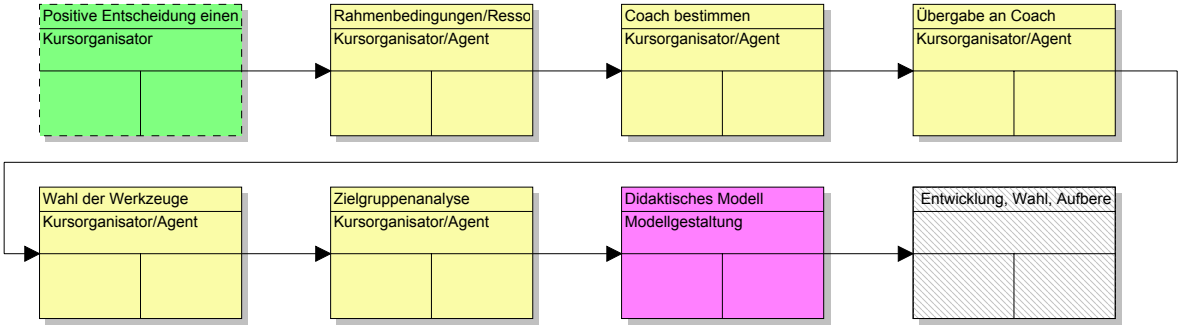


Abbildung 49: Prozessdiagramm „Kursvorbereitung“

1.1.1. Positive Entscheidung einen Distance Education Kurs anzubieten (Kursorganisator)

Hier ist eine positive Entscheidung gefallen einen Distance Education Kurs anzubieten. Dabei steht bereits das Thema des Kurses und die Zielgruppe (z.B.: Schüler, Studenten, Erwachsene, Anfänger, Erfahrene, Experten, etc.) fest.

1.1.2. Rahmenbedingungen/Ressourcen (Kursorganisator/Agent)

Der Kursorganisator muss die Rahmenbedingungen und die verfügbaren Ressourcen klären. Dafür muss er bestimmen, welche Mitarbeiter (Coaches) zur Verfügung stehen, wieviele Teilnehmer möglich sind, ob bereits Werkzeuge, Lernmaterial, etc. zur Verfügung stehen. Auch das Budget muss vorgegeben werden.

Besonders bei Distance Coaching ist die Teilnehmerzahl entscheidend. Danach richtet sich die Anzahl der Gruppen und somit auch die Anzahl der Coaches/Tutoren. Bei Pilotstudien hat

sich heraus gestellt, dass es für die Lernenden nicht wichtig ist, welche Person hinter der Rolle als Coach steckt. So gab es das Modell, dass zwei unterschiedliche Personen als Coach im System auftraten und gleichermaßen auf Fragen antworteten, motivierten, informierten, ... Wichtig dabei ist, dass alle Personen hinter dem Coach kompetent auftreten können und der Coach mit "einer Zunge spricht", d.h. es zu keinen Widersprüchen in den Auskünften kommt.

Bei der Klärung der Rahmenbedingungen/Ressourcen können intelligente Agenten ihren Einsatz finden. Diese Agenten können Checklisten abarbeiten und dabei verfügbare Ressourcen, wie Mitarbeiter, Raumkapazitäten, vorhandene Werkzeuge, usw. prüfen. Die Ergebnisse kann der Agent anschließend dem Kursorganisator bzw. dem Coach präsentieren.

1.1.3. Coach bestimmen (Kursorganisator/Agent)

Je nach gewünschter Qualifikation und Verfügbarkeit wird der Kursleiter (Coach) bestimmt, falls dieser nicht zugleich der Kursorganisator ist. Vorausgesetzt es existiert eine Datenbank mit Qualifikation und Arbeitskalender aller Mitarbeiter können passende Mitarbeiter von Agenten aus der Datenbank ermittelt werden.

1.1.4. Übergabe an Coach (Kursorganisator/Agent)

Die ermittelten Rahmenbedingungen, Ressourcen und eine Liste der zur Verfügung stehenden Werkzeuge wird an den Coach übergeben. Diese Daten und Listen sind zuvor im Schritt „Rahmenbedingung/Ressourcen“ ermittelt und zusammengestellt worden.

1.1.5. Wahl der Werkzeuge (Kursorganisator/Agent)

Bei den Werkzeugen kann unterschieden werden zwischen:

- Werkzeuge zur Lernmaterialgestaltung (Authoringtools, ...)
- Werkzeuge zur Planung des didaktischen Modells
- Werkzeuge zur Kursablaufplanung
- Werkzeuge zur Kursdurchführung (Lernplattform, Kommunikationsmittel, etc.)
- Werkzeuge zum Assessment/Zertifizierung
- Werkzeuge zur Kursevaluierung, etc.

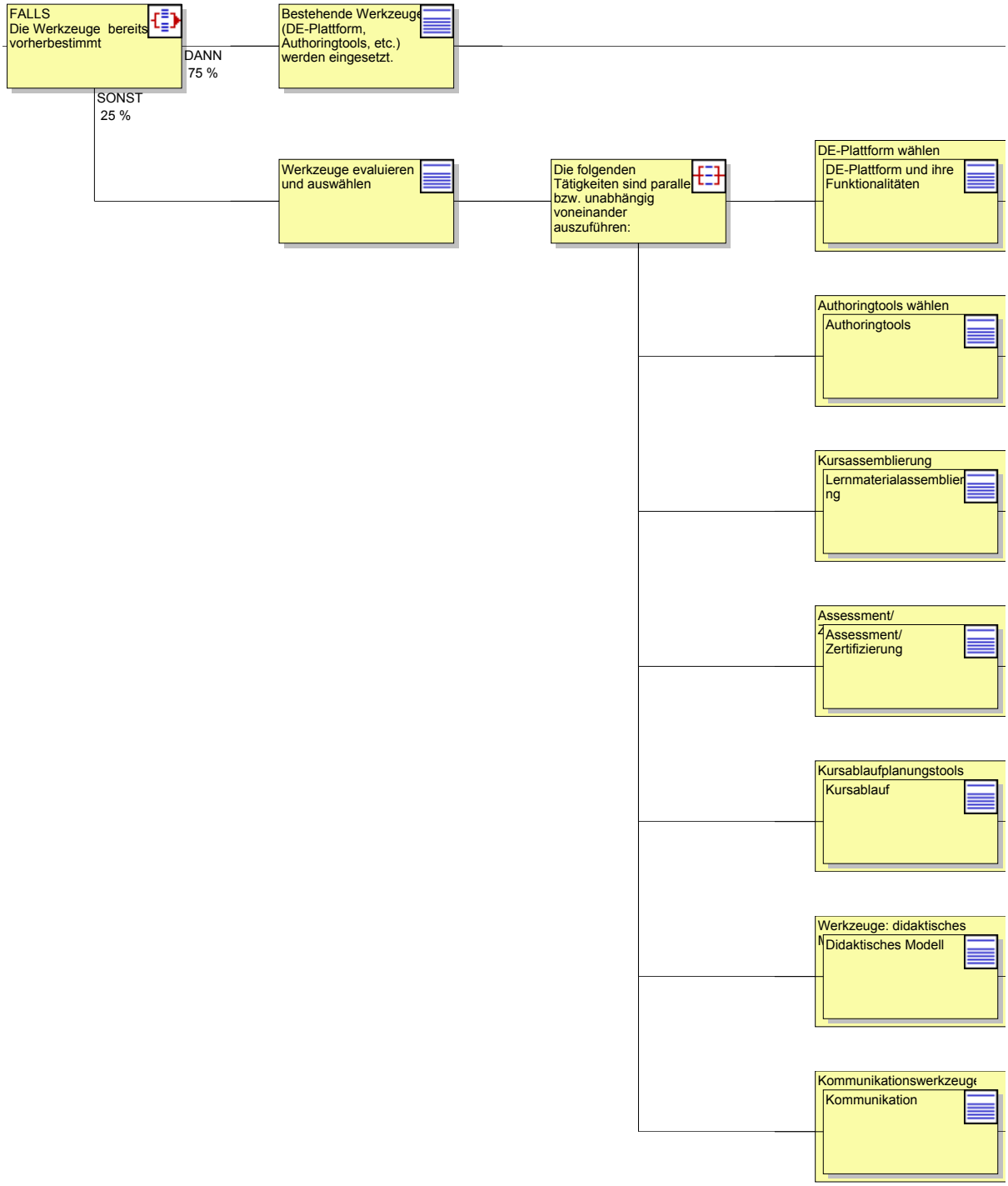


Abbildung 50: Detaildiagramm „Wahl der Werkzeuge“

IF Werkzeuge bereits vorherbestimmt,

THEN Bestehende Werkzeuge (DE-Plattform, Authoringtools, etc.) werden eingesetzt.

ELSE <u>Werkzeuge evaluieren und auswählen:</u> Für einen DE-Kurs sind unterschiedliche Werkzeuge notwendig. Dabei spielen Ressourcen, Kosten, etc. eine Rolle. Eine

zentrale Rolle spielt die DE-Plattform. Nach der Funktionalität, die eine Plattform zur Verfügung stellt, richten sich Werkzeuge, die zusätzlich zum Einsatz kommen. Welche Unterstützung für Lernmaterial ist etwa gegeben? Welche Formate, Standards (etwa CPS von IMS) werden dabei unterstützt? Auf Basis der Antworten zu diesen Fragen müssen die entsprechenden Authoringtools, gewählt werden.

DE-Plattform wählen: Die DE-Plattform und ihre Funktionalität sind zentral. Je nachdem welche Lernmaterialformate bzw. -standards unterstützt werden, richten sich auch die zur Anwendung kommenden Authoringtools für das Lernmaterial. Die in der Plattform zur Verfügung stehenden bzw. integrierbaren Kommunikationswerkzeuge bestimmen wesentlich zusätzliche Kommunikations-werkzeuge bzw. auch das didaktische Modell.

Authoringtools: Bei den Authoringtools handelt es sich um jene Werkzeuge mit denen Lernmaterial erstellt werden kann. Dabei kann es sich um XML-Werkzeuge für Standards (etwa CPS) handeln mit denen Kursinhalte aufbereitet und gegliedert werden. Es handelt sich hierbei aber auch um Werkzeuge zur Texterstellung, graphischen Umsetzung, Applet-Erstellung, etc.

Lernmaterialassemblierung: Für einen DE-Kurs wird meist nicht nur neues, sondern vor allem auch bereits vorhandenes Lernmaterial eingesetzt. Zur Auffindung von bereits vorhandenen Lernmaterial und der Lernmaterial-assemblierung können sehr gut Agenten eingesetzt werden, die ihre Informationen über Kursmetadaten gewinnen und diesen entsprechend Kurse zielgruppengerecht zusammen stellen.

Assessment/Zertifizierung: Assessment und abschließende Zertifizierung der Kursteilnehmer (durch vorhergehende Examinierung) können durch den Einsatz von Agenten weitgehend automatisiert werden. Aus Kursrepositories können den Metadaten entsprechend Assessmentfragen und Antworten beliebig zusammengestellt werden. Dies gilt auch für Prüfungen. Diese sind von Agenten vollkommen autonom durchführbar. Der Agent wählt Prüfungsfragen aus, die Teilnehmer beantworten diese im System und der Agent führt anschließend die Bewertung der Antworten durch. Ist dies nicht erwünscht kann ein Agent auch

(nur) einen Prüfungsvorschlag aufbereiten, den der Coach nachbereiten kann. Die Ausstellung der Zertifikate vermag entsprechend den Prüfungsergebnissen ebenfalls wieder ein Agenten übernehmen.

Kursablauf: Nicht nur die Kursinhalte und die Kursumgebung erfordern Werkzeuge, sondern auch für die Planung des Kursablaufs sind Werkzeuge (Projektplanung, etc.) einsetzbar.

Didaktisches Modell: Zur Erstellung des didaktischen Modells ist ein Zugriff auf bereits vorhandene Erfahrungen und Dokumente möglich. Sind die entsprechenden Daten in einem Repository inkl. Metadaten abgelegt, so kann ein Agent der Spezifikation entsprechend Material suchen und dieses bereitstellen.

Kommunikation: Es ist möglich, dass zusätzlich zu den von der DE-Plattform zur Verfügung gestellten Kommunikationswerkzeugen noch zusätzliche Werkzeuge (Email, ICQ, etc.) zum Einsatz kommen.

1.1.6. Zielgruppenanalyse (Kursorganisator/Agent)

Die Zielgruppenanalyse soll darüber Auskunft geben, welche Vorkenntnisse, Wünsche, Bedürfnisse, Lerngewohnheiten, erwartete Akzeptanz, etc. die Zielgruppe hat. So bestimmen etwa die Vorkenntnisse der Kursteilnehmer das didaktische Modell, aber vor allem das Lernmaterial mit. Auch die Vorkenntnisse im Umgang mit der Informationstechnologie und die Art der Internetverbindung der Teilnehmer sind entscheidend. Danach richtet sich der erforderliche technische Support, der Einschulungsbedarf und auch die Entscheidung, ob Handouts oder CD´s zum Offline-Lernen zur Verfügung gestellt werden sollen.

Diese Analyse kann mittels Fragebogen oder Lösung von standardisierten Aufgaben erfolgen. Die Präsentation dieser Fragebögen und Aufgaben und die anschließende Auswertung ist ein mögliches Einsatzgebiet für einen Agenten. Die Ergebnisse sollten anschließend in die Wahl des didaktischen Modells und in die Lernmaterialentwicklung einfließen.

1.1.7. Didaktisches Modell (Modellgestaltung)

Nach Zielgruppe, verfügbaren Ressourcen (Plattform, Veranstaltungssäle), Gruppengröße, Art (Workshop, Vorlesungscharakter, Praktikum, etc.) und Dauer des Kurses, etc. richtet sich die

Wahl des didaktischen Modells. Erfahrungen aus Pilotstudien haben gezeigt, dass ein hybrides Lernmodell von Lernenden sehr gut aufgenommen wird. Dabei wird Distance Coaching/Learning immer wieder von Präsenzphasen abgewechselt.

Mögliche Elemente eines didaktischen Modells (Lernarrangements):

- Interaktion in Präsenzphasen: Als Präsenzphasen bieten sich zunächst ein Kick-Off Meeting und ein Abschlusstreffen an. Dazwischen können je nach Dauer des Kurses mehrere Treffen abgehalten werden. Diese dienen vorrangig zur Verstärkung und Festigung der sozialen Kontakte und in zweiter Linie kann dabei vom Coach ein roter Faden oder eine Erläuterung besonderer Aspekte gegeben werden.

- Lernmaterial: Für die DE-Lernphase sollten unterschiedlich aufbereitetes Lernmaterial, multiple Kommunikationswege, etc. zur Verfügung gestellt werden.

- Handouts: Offline können Handouts (gedrucktes Skriptum, CD´s) zum Studium für unterwegs ausgegeben werden.

- Lerngruppen: Lerngruppen und gemeinsame Gruppenaufgaben sind besonders empfehlenswert. Zum einen sind hier die Lernenden nicht nur auf sich allein gestellt, sondern haben die Möglichkeit ihren Lernfortschritt an den Gruppenmitgliedern zu messen. Andererseits können sich die Gruppenmitglieder auch gegenseitig unterstützen. Zusätzlich wird durch 3-Level Coaching der eigentliche Coach (Kursleiter) entlastet. 1-Level: Lernende bekommen Hilfe innerhalb der Gruppe. 2-Level: Lernende bekommen gruppenübergreifende Hilfe. 3-Level: Lernende bekommen Hilfe durch den Coach. Durch dieses Modell wird der Coach entlastet und hat somit Zeit um sich kniffligen Fragen zu widmen. Dadurch wird aber auch bei mehreren Lernenden ein Problembewusstsein hervorgerufen und der Lernstoff wird intensiver bearbeitet. Auch ein 4-Level Coaching ist denkbar. Dabei wird ein Agent eingesetzt. Einfache, wiederkehrende Fragen können dabei durch den Agenten autonom beantwortet werden. Der Agent kann zusätzlich Fragen filtern und gegebenenfalls an die zuständigen Coaches oder Tutoren weiterleiten.

- Learning-By-Teaching: Bei Learning-By-Teaching übernehmen Lernende für bestimmte Lernbereiche die Rolle des Coach. Sie stellen Fragen, bewerten, diskutieren

und korrigieren die Antworten der anderen. Dabei wird der Lernstoff besonders detailliert wahrgenommen und verinnerlicht.

- Coach: Der Coach spielt im didaktischen Modell eine zentrale Rolle. Er ist der Orchestrator, derjenige der die Lernumgebung gestaltet und zur Verfügung stellt. Darüber hinaus ist er derjenige der motiviert, anregt, neue oder alternative Wege aufzeigt, moderiert und Hilfestellung leistet.

Anschließend wird der Prozess „Entwicklung, Wahl, Aufbereiten des Lernmaterials“ initiiert.

1.1.8. Entwicklung, Wahl, Aufbereiten des Lernmaterials

„Entwicklung, Wahl, Aufbereiten des Lernmaterials“ ist ein eigener Prozess der dem hier beschriebenen Prozess „Kursvorbereitung“ nachfolgt.

1.2. Entwicklung, Wahl, Aufbereiten des Lernmaterials

Bei der Wahl des Lernmaterials kann es sich um die Aufbereitung/Gestaltung/Umsetzung neuen Lernmaterials handeln bzw. um die Auswahl von bereits bestehenden Lernmaterial aus einem Repository. Beim Lernmaterial kann es sich um Lernmaterial für online und offline Teaching/Learning handeln.

Der Prozess „Entwicklung, Wahl, Aufbereiten des Lernmaterials“ umfasst folgende Prozessschritte:

1 Prüfen, ob bereits Lernmaterial vorhanden (Materialgestaltung/Agent)
2 Lernmaterial ergänzen (Materialgestaltung)
3 Parallele Prozessschritte:
  3a Online-Lernmaterial zielgruppengerecht aufbereiten (Materialgestaltung/Agent)
  3b CD-Rom´s, Handouts zusammen stellen und anfertigen lassen (Materialgestaltung/Agent)
4 Initiierung des Prozesses „Umsetzung der Lernlandschaft“

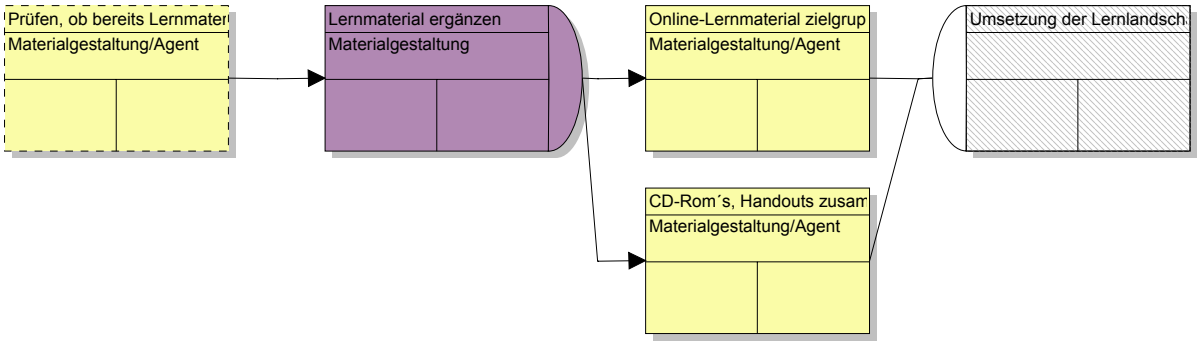


Abbildung 51: Prozessdiagramm „Entwicklung, Wahl, Aufbereiten des Lernmaterials“

1.2.1. Prüfen, ob bereits Lernmaterial vorhanden (Materialgestaltung/Agent)

Die Prüfung, ob bereits Lernmaterial vorhanden ist, ist von Agenten durchführbar. Diese können entsprechend der Kursspezifikation (Zielgruppe, Thema, etc.) Kursrepositories durchforsten und Kursmaterial daraus zur Verfügung stellen.

1.2.2. Lernmaterial ergänzen (Materialgestaltung)

Fehlendes Lernmaterial muss ergänzt werden. Dazu wird festgestellt in welchen Bereichen und welche Art von Lernmaterial fehlt. Fehlen erklärende Teile (Lernen durch Hinweise), Assessment (Lernen durch Übung), Hands-on Experience-Teile (entdeckendes Lernen), Quizzes (unterhaltendes Lernen), etc. werden erstellt und ergänzt.

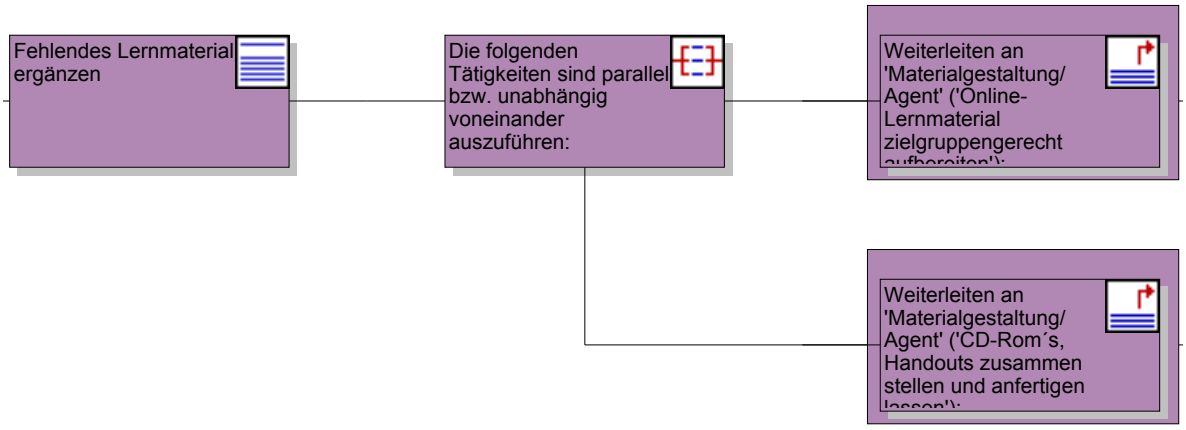


Abbildung 52: Detaildiagramm „Lernmaterial ergänzen“

Anschließend werden die Prozessschritte „Online-Lernmaterial zielgruppengerecht aufbereiten“ und „CD-Rom´s, Handouts zusammen stellen und anfertigen lassen“ angestoßen.

1.2.3a. Online-Lernmaterial zielgruppengerecht aufbereiten (Materialgestaltung/Agent)

Bei der zielgruppengerechten Aufbereitung des Lernmaterials muss bedacht werden, wieviel Text, Graphik, etc. der Zielgruppe auf einmal zugemutet werden kann. Auch Auflockungs- und Reflektionsteile sollen im Lernmaterial vorhanden sein. Der Lernende darf mit dem Inhalt und seiner Aufbereitung nicht überfordert und überfrachtet werden. Deshalb empfiehlt sich die Inhalte in "kleine Portionen" zu unterteilen und diese mit Assessment und Hands-on Experience Teilen zu bereichern. Auch eine einheitliche Navigation durch die unterschiedlichen Lernmaterialien ist essentiell.

Generell gilt für den Aufbau von Lernmaterial:

- Übersicht (Gliederung, Text, graphische "Landkarte" - vorgeschlagener Lernpfad (guided tour)),

- Information (kurze Einheiten),
- anwendungsbezogene Medialisierung (HTML, Applets, Graphik, Video, Audio, Simulationen),
- Assessment (Übungen, Tests, Feedback).

Parallel dazu:

- Nachschlagewerke,
- Links zu zusätzlicher Information, etc.

Anschließend an diesen Prozessschritt wird der Prozess „Umsetzung der Lernlandschaft“ initiiert.

1.2.3b. CD-Rom´s, Handouts zusammen stellen und anfertigen lassen (Materialgestaltung/Agent

Um den Lernenden auch die Möglichkeit zu geben, offline zu lernen, etwa wenn diese unterwegs sind oder keinen ständigen Online-Zugriff haben, können CD-Rom´s bzw. Handouts zur Verfügung gestellt werden. Bei einer CD-Rom ist es von Vorteil, wenn diese die gleiche Lernlandschaft darbietet, wie sie auch online zur Verfügung steht, sprich das Aussehen, die Navigation und die Inhalte sind äquivalent. Der einzige Unterschied besteht in den Kommunikationswerkzeugen, Arbeitsverzeichnissen und Foren, die nur online zur Verfügung stehen.

Anschließend wird der Prozess „Umsetzung der Lernlandschaft“ initiiert.

1.2.4. Umsetzung der Lernlandschaft

„Umsetzung der Lernlandschaft“ ist ein eigener Prozess der dem hier beschriebenen Prozess „Entwicklung, Wahl, Aufbereiten des Lernmaterials“ nachfolgt.

1.3. Umsetzung der Lernlandschaft

Die Settings werden in der DE-Plattform umgesetzt und das Online-Lernmaterial eingespielt. Auch die Offline-Bereiche werden aufbereitet, CD´s gebrannt, Handbücher gedruckt, Veranstaltungsräume reserviert, etc.

Der Prozess „Umsetzung der Lernlandschaft“ umfasst folgende Prozessschritte:

1 Parallele Prozessschritte:
- 1a Kursanmeldung der Lernenden (Lernende)
- 1b Festlegung erster Termine (Coach)
- 1c Bereitstellung der Handouts (Coach)
- 1d1 Plattform aufsetzen (Systemadministrator)
  - 1d2 Accounts anlegen (Systemadministrator/Agent)
  - 1d3 Arbeitsstruktur anlegen (Systemadministrator/Agent)
  - 1d4 Lernmaterial einbetten (Systemadministrator/Agent)
  - 1d5 Zusatzinformationen einbetten (Systemadministrator/Agent)
  - 1d6 Stellt erste Postings in die DE-Plattform (Coach)

2 Initiierung des Prozesses „Kursdurchführung“

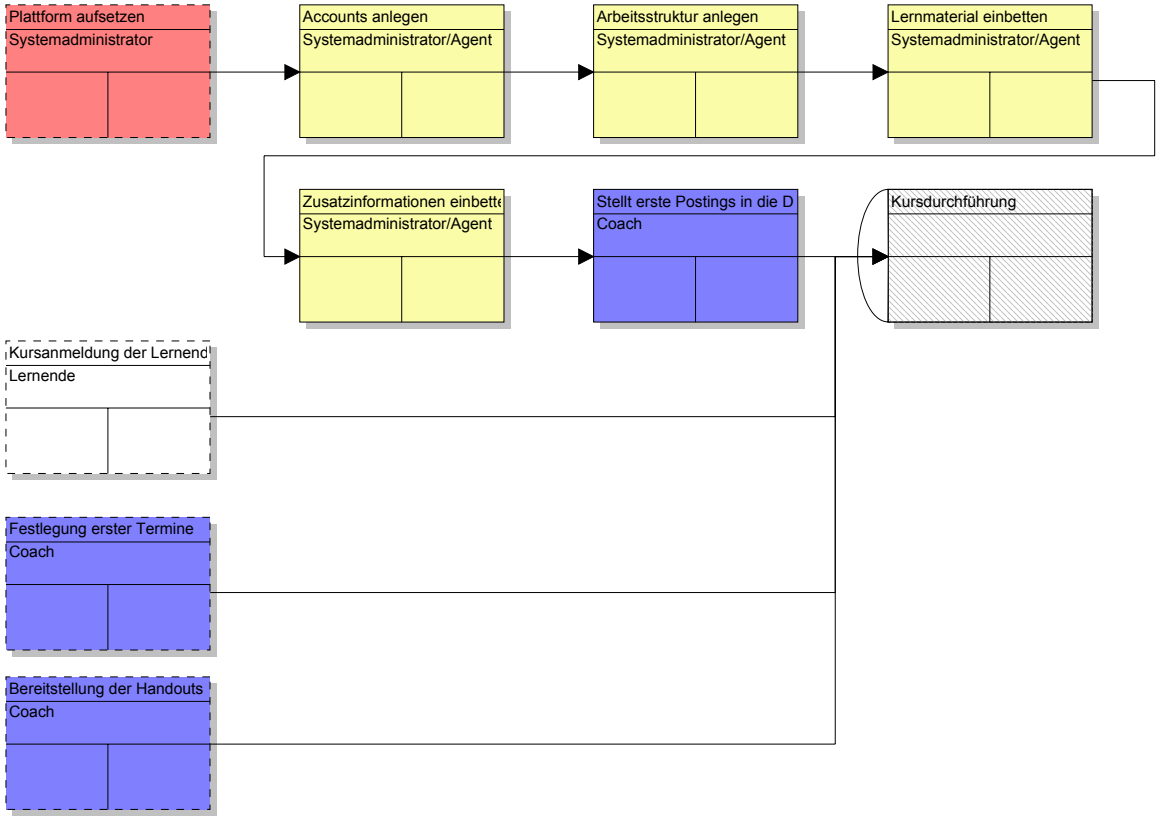


Abbildung 53: Prozessdiagramm „Umsetzung der Lernlandschaft“

1.3.1a. Kursanmeldung der Lernenden (Lernende)

Die Lernenden melden sich für den Kurs an. Dabei können bereits Daten, Vorstellungen und Wünsche erfragt werden und in die Plattform und das didaktische Modell einfließen.

1.3.1b.Festlegung erster Termine (Coach)

Der Coach legt Termine für die Präsenzphasen, für die Abgabe von Aufgaben und für eine eventuelle abschließende Klausur fest.

1.3.1c.Bereitstellung der Handouts (Coach)

Die Handouts, die das gedruckte Lernmaterial enthalten, werden den Lernenden zur Verfügung gestellt. Zusätzlich können die Handouts Anleitungen zur Lernform (wie lernt man selbstorganisiert, etc.), Informationen zur Organisation des Kurses (Ansprechpartner, Ablauf, etc.) und Hinweise zur Bedienung der Plattform enthalten.

1.3.1d1. Plattform aufsetzen (Systemadministrator)

Der Systemadministrator setzt die Plattform und die dazugehörige Software (Datenbank, etc.) auf und stellt Zugriffs-, Replikations- und Backupmöglichkeiten zur Verfügung.

1.3.1d2. Accounts anlegen (Systemadministrator/Agent)

Für die Lernenden und Coaches/Tutor(en) werden Accounts in der DE-Plattform angelegt. Dabei müssen Zugriffsrechte, Loginnamen und Passwörter vergeben werden.

Dies kann auch ein Plattformagent übernehmen. Der Plattformagent kann Rollen entsprechend Zugriffsrechte (Administratorrechte, Schreib- und Leserechte, etc.) vergeben, Loginnamen (etwa kreiert aus dem Kursnamen oder den Namen der Teilnehmer) generieren und Defaultpasswörter anlegen. Die Struktur kann dabei ohne konkrete Daten der Lernenden aufbereitet werden. Stehen konkrete Daten fest, werden diese in die Plattform übernommen.

1.3.1d3. Arbeitsstruktur anlegen (Systemadministrator/Agent)

In der Plattform werden entsprechend dem didaktischen Modell Foren, Verzeichnisse, Kommunikationswerkzeuge eingerichtet. Dies kann ebenfalls wieder ein Plattformagent übernehmen. Dieser richtet Informations-, Diskussions- und Gruppenforen mit entsprechenden Zugriffsrechten für Lernende und Coaches/Tutoren ein. Ebenso geht er mit Arbeitsverzeichnissen für Lernende, Lerngruppen, etc. vor.

Je nach Kurstyp (Seminar, Praktikum, etc.) variiert die Verzeichnisstruktur. Bei einem Praktikum sind etwa Abgabeordner für jeden einzelnen Lernenden bzw. für jede Gruppe

sinnvoll. Jeder Lernende bzw. jedes Gruppenmitglied sieht dabei nur sein (Gruppen-) Abgabeverzeichnis und nur er selbst und der Coach können darauf zugreifen. Für die Angabe hingegen wird meist nur ein oder ein Angabeverzeichnis pro Gruppe benötigt. Auch ist wichtig, dass die Lernenden und die Lerngruppen eigene Arbeitsbereiche (personal webspaces) zur Verfügung haben.

Die interne Struktur die ausschließlich zur internen Kursorganisation dient, kann ebenfalls durch den Plattformagenten abgebildet werden. Der Agent kann vorweg Kommunikationsmittel, wie Foren, Email, etc. einrichten oder diese aber auch erst im laufenden Betrieb einrichten, wenn sie tatsächlich gebraucht werden (etwa zum tatsächlichen Chattermin).

1.3.1d4. Lernmaterial einbetten (Systemadministrator/Agent)

Das zur Verfügung stehende Lernmaterial wird in die Plattform gestellt. Ein Agent prüft auf Wunsch oder periodisch, ob ein Update des Lernmaterials in die Plattform zu stellen ist.

1.3.1d5. Zusatzinformationen einbetten (Systemadministrator/Agent)

Zusätzliche Informationen, wie Links, Wörterbücher, Glossars, Literaturverzeichnisse und Dienste, wie Technischer Support können in die Plattform gestellt und eingerichtet werden. Aufgrund einer Metadatenbeschreibung vermag dies ein Agent zu übernehmen. Dieser führt diese Aufgabe entweder autonom durch oder unterbreitet dem Coach einen Vorschlag, der nach eventueller Modifikation in die Plattform übernommen wird.

1.3.1d6. Stellt erste Postings in die DE-Plattform (Coach)

Es empfiehlt sich, dass der Coach von Beginn an Präsenz zeigt. Beim ersten Posting sollten die Lernenden begrüßt und hinsichtlich der Kursziele und der Lernform motiviert werden. Anschließend wird der Prozess „Kursdurchführung“ initiiert.

1.3.2. Kursdurchführung

Der Prozess „Kursdurchführung“ folgt dem Prozess „Umsetzung der Lernlandschaft“ nach.

2 Teil 2: Kursdurchführung

Die Kursdurchführung umfasst alle Prozesse vom Kick-Off Meeting, über die eigentliche Durchführung bis hin zur Abschlussveranstaltung, der Prüfung und der Zertifizierung.

2.1. Kursstart

Im Prozess Kursstart wird der Kurs gestartet.

Der Prozess „Kursstart“ umfasst folgende Prozessschritte:

1 Coach startet Kurs (Coach)

2 Initiierung des Prozesses „Kick-Off“

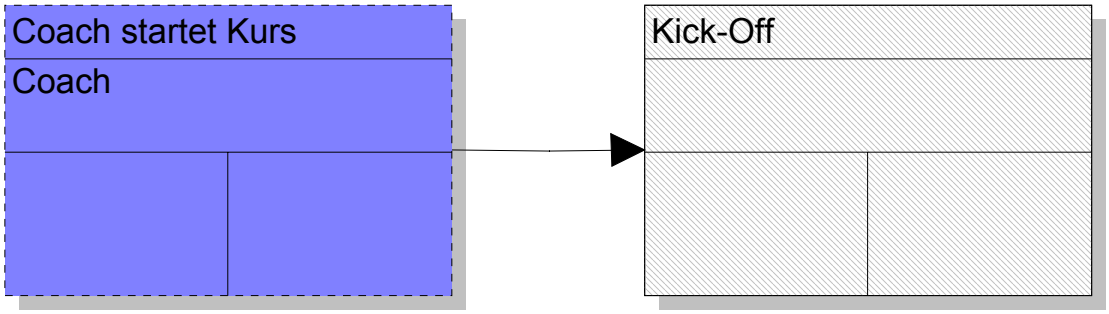


Abbildung 54: Prozessdiagramm „Kursstart“

2.1.1. Coach startet Kurs (Coach

Nach Abschluss der Vorbereitung wird der Kurs vom Coach gestartet.

Anschließend wird der Prozess „Kick-Off“ initiiert.

2.1.2. Kick-Off

„Kick-Off“ ist ein eigener Prozess der dem hier beschriebenen Prozess „Kursstsart“ nachfolgt.

2.2. Kick-Off

Der Prozess „Kick-Off“ umfasst folgende Prozessschritte:

1 Kick-Off wird gestartet (Coach)
2 Teamvorstellung (Coach)
3 Erläuterung der DE-Plattform (Coach/Agent)
4 Optionaler Prozessschritt (je nach Teilnehmerzahl):
Teilnehmer lernen sich kennen (Lernende)
5 Gruppenbildung (Lernende)
6 Parallele Prozessschritte (entweder oder):
6a1 Gruppenfindung (Lernende)
6a2 Lösen erste gemeinsame Aufgabe(n) (Lernende)
6a3 Gruppensprecher bestimmen (Lernende)
6a3 Ergebnisse präsentieren (Gruppensprecher)
6b1 Lösen erste gemeinsame Aufgabe(n) (Lernende)
6b2 Ergebnisse präsentieren (Lernende)
7 Ideen sortieren (Coach)
8 Pause (Coach)
9 Weitere Vorgehensweise (Coach)
10 Optionaler Prozessschritt: Präsentation Lernmaterial (Coach/Agent)
11 Aufgaben lösen (Lernende)
12 Abschluss (Coach)
13 Erläutern von noch offenen Fragen (Coach)
14 Kick-Off wird beendet (Coach)
15 Evaluierung (Evaluierungsstelle/Agent)
16 Parallele Prozessschritte:
16a Initiierung des Prozesses „DE-Kursdurchführung“
16b Hilft bei technischen Anfragen und Problemen weiter (Technischer Dienst)

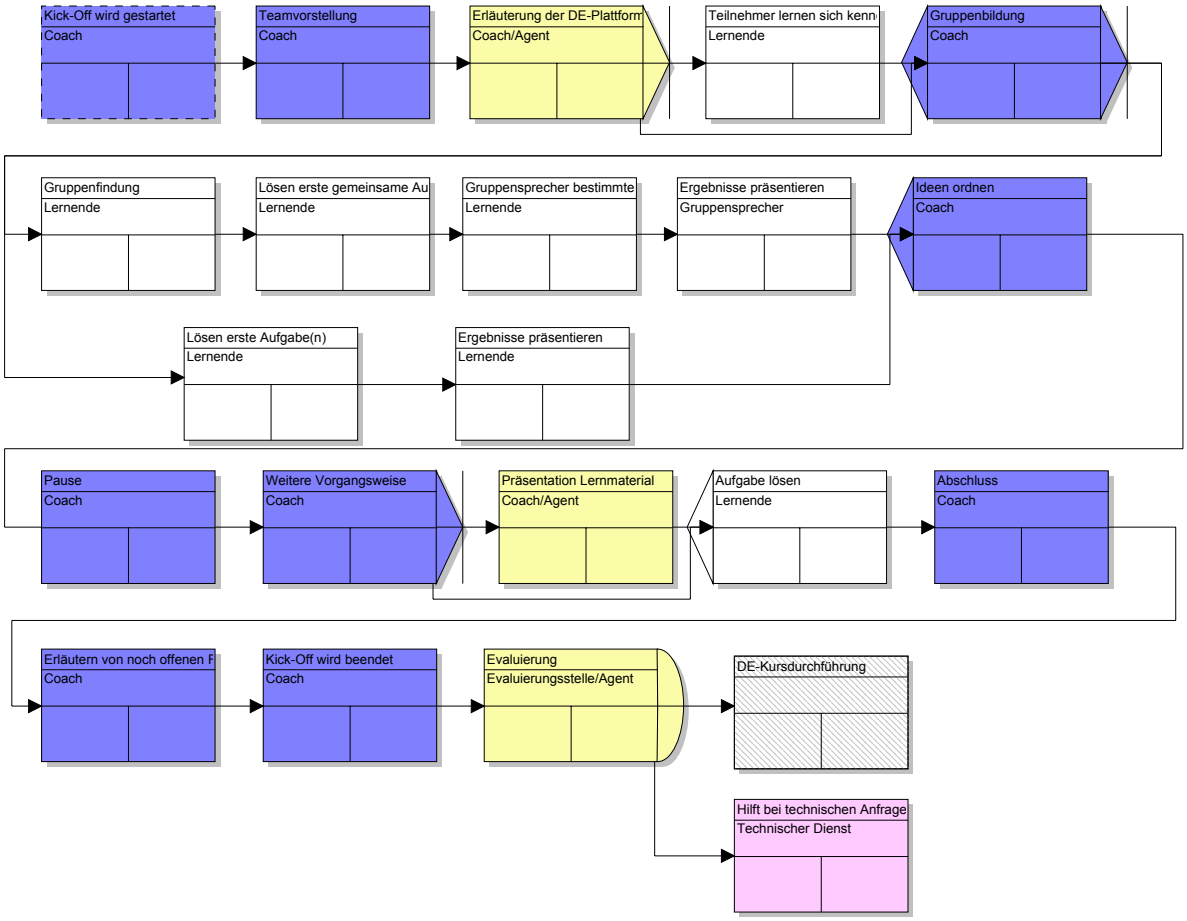


Abbildung 55: Prozessdiagramm „Kick-Off“

2.2.1. Kick-Off wird gestartet (Coach)

Der Coach startet das Kick-Off Meeting. Das Kick-Off Meeting ist von zentraler Bedeutung für den potentiellen Verlauf des Lernprozesses.

Das Kick-Off Meeting kann online stattfinden. Einfacher und auch effizienter für das Gemeinschaftsgefühl ist es jedoch, wenn das Kick-Off Meeting als Präsenzveranstaltung offline abgehalten werden kann. Für das Kick-Off Meeting sollte je nach Gruppengröße einige Zeit eingeplant werden. Bei einer Teilnehmerzahl von rund 50 Personen ist mit 4 Stunden zu rechnen.

2.2.2. Teamvorstellung (Coach)

Als ersten Schritt stellt der Coach sich selbst und sein Team vor. Dabei werden die Aufgabengebiete klar definiert dargelegt. Der Coach positioniert sich gegenüber den

Lernenden und gegenüber den Lerninhalten. In einem Kick-Off Meeting von etwa 4 Stunden umfasst dies 15-20 Minuten.

2.2.3. Erläuterung der DE-Plattform (Coach/Agent)

Je nach Kenntnisstand der Lernenden wird die DE-Plattform, in der der Kurs eingebettet ist, erläutert. Wird die Plattform und ihre eingebettete Struktur erklärt, so ist es wichtig auf die zur Verfügung stehende Funktionalität, die möglichen Kommunikationswege (Foren, Chats, etc.), die eingerichteten Verzeichnisse, etc. einzugehen. Wichtig ist vor allem auch aufzuzeigen, welches Lernmaterial (Inhalt und Form) zur Verfügung steht und wo Zusatzmaterial, technischer Support, etc. zu finden sind.

Sind entsprechende Agenten vorhanden, kann die Erläuterung und Einführung in die Plattform durch Agenten übernommen werden, die dabei als Guide fungieren. Durch den Einsatz von Agenten, kann die Einführung vom Kick-Off Meeting auch in die Kursdurchführungsphase verlagert werden.

In einem Kick-Off Meeting von etwa 4 Stunden umfasst dies 30 - 40 Minuten. Wird die Plattform erläutert, sollte danach eine Pause von 15 Minuten stattfinden.

2.2.4. Teilnehmer lernen sich kennen (Lernende)

Die Lernenden sollen sich selbst vorstellen und ihr Startprofil (Vorwissen, Einstellung zum Gegenstand, zur Lernform und ihre Motivation) darlegen.

Dabei bekommen alle Lernenden einen ersten Eindruck voneinander und der Coach kann bereits abschätzen, wo hoch die Ausgangsmotivation hinsichtlich des Kursinhalts und des Lernmodells ist und diese Motivation entsprechend verstärken, bzw. auf befürchtete Nachteile eingehen.

In einem Kick-Off Meeting ist pro Teilnehmer mit etwa 1-3 Minuten zu rechnen. Bei einer sehr großen Teilnehmerzahl kann dies als erste Aufgabe für die Online-Lernphase in den Prozess der Kursdurchführung verschoben werden. Dabei hat dann jeder Teilnehmer die Aufgabe eine Kurzvorstellung seiner Person in ein spezielles Forum zu posten.

2.2.5. Gruppenbildung (Coach)

Falls im Laufe des Kurses Aufgaben in der Gruppe zu lösen sein werden, ist es sinnvoll bereits im Kick-Off Meeting diese Gruppen zu bilden und somit das Gemeinschaftsgefühl zu verstärken.

Dabei wird den Lernenden die Aufgabe gestellt, sich zu Gruppen von x Personen (Kleingruppen bis zu 5 Personen arbeiten besonders effektiv) zusammen zu finden, einen Gruppensprecher und Gruppenregeln festzulegen und eine erste Aufgabe gemeinsam zu lösen und die Lösung zu präsentieren.

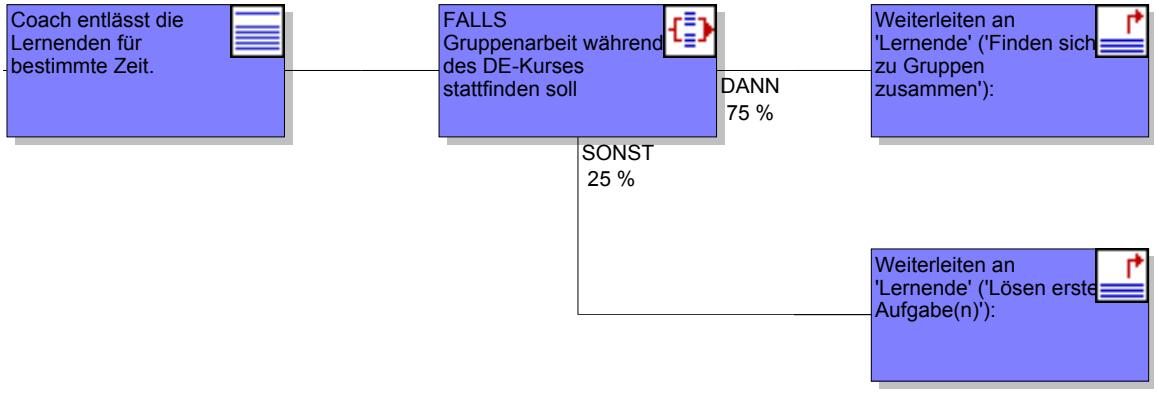


Abbildung 56: Detaildiagramm „Gruppenbildung“

Coach entlässt die Lernenden für bestimmte Zeit: Die Gruppen sollen etwa ein Drittel der Zeit, die für das Kick-Off Meeting zur Verfügung steht, zur Zusammenarbeit haben.

IF Gruppenarbeit während des Kurses stattfinden soll

THEN Prozessschritt „Finden sich in Gruppen zusammen“

ELSE Prozessschritt „Lösen erste Aufgabe(n)“

2.2.6a1. Gruppenfindung (Lernende)

Die einzelnen Lernenden sollen sich zu Gruppen zusammen finden. Danach können sich die neu gebildeten Gruppen für einige Zeit zurückziehen, um ihre Aufgabe zu lösen. In einem Kick-Off Meeting von etwa 4 Stunden umfasst dies 5-10 Minuten.

2.2.6a2. Lösen erste gemeinsame Aufgabe(n) (Lernende)

Damit sich die zukünftigen Gruppenmitglieder besser kennen lernen, sollen die Gruppenmitglieder erste Aufgaben gemeinsam lösen. Z.B.: Brainstorming zum Thema des Kurses (was verbinden die einzelnen Gruppenmitglieder mit dem Thema, was versprechen sie sich vom Kurs, etc.). In einem Kick-Off Meeting von etwa 4 Stunden umfasst dies 35-40 Minuten.

2.2.6a3. Gruppensprecher bestimmen (Lernende)

Die Gruppe soll einen Gruppensprecher bestimmen, der die Gruppe nach außen präsentiert. In einem Kick-Off Meeting von etwa 4 Stunden umfasst dies 3-5 Minuten.

2.2.6a4. Ergebnisse präsentieren (Gruppensprecher)

Die Gruppensprecher präsentieren die Ergebnisse ihrer Gruppe dem Plenum. Die Präsentation soll dabei über einfache Mittel, wie Flipcharts oder Overheadfolien erfolgen. In einem Kick-Off Meeting von etwa 4 Stunden umfasst dies pro Präsentation 3-5 Minuten.

2.2.6b1. Lösen erste Aufgabe(n) (Lernende)

Damit sich die Lernenden besser kennen lernen, sollen sie erste Aufgaben lösen, dies kann auch hier in Kleingruppen geschehen. Mögliche Aufgaben: Brainstorming zum Thema des Kurses (was verbinden die einzelnen Lernenden mit dem Thema, was versprechen sie sich vom Kurs, etc.). In einem Kick-Off Meeting von etwa 4 Stunden umfasst dies 35-40 Minuten.

2.2.6b2. Ergebnisse präsentieren (Lernende)

Die Lernenden präsentieren den anderen ihre Ergebnisse. Dabei soll die Präsentation über einfache Mittel, wie Flipcharts oder Overheadfolien erfolgen. In einem Kick-Off Meeting von etwa 4 Stunden umfasst dies pro Präsentation 3-5 Minuten.

2.2.7. Ideen ordnen (Coach)

Die Ergebnisse werden zusammen gefasst, kommentiert und gegebenenfalls ergänzt. Der Coach kann aus den Ergebnissen und Ideen der Lernenden eine Abbildung als Mapping-Spinne entwickeln. Wichtig dabei: abstrakte Metabegriffe finden, was wird im Kurs (nicht) gemacht werden und warum, eventuell persönlicher Bezug, etc. In einem Kick-Off Meeting von etwa 4 Stunden umfasst dies 30 Minuten.

2.2.8. Pause (Coach)

Anschließend sollte eine Pause von etwa 15 Minuten stattfinden.

2.2.9. Weitere Vorgangsweise (Coach)

Der Coach erläutert die weitere Vorgangsweise. D.h., die geplante Kursdurchführung, die einzelnen Aufgaben, Termine (Zeitplanung), Meilensteine und Ziele ("wo geht die Reise hin") werden exakt definiert und mit allen kommuniziert. Auch die Bewertung wird besprochen (fließen Beiträge der Lernenden mit in die Benotung ein, wie fließen die Ausarbeitungen der Aufgaben für einzelne Lernende und Gruppen ein, gibt es eine Abschlussprüfung, etc.).

Die Spielregeln (Netiquette, wann und wie schnell ist Hilfe von wem zu erwarten, welche Aufgaben gilt es zu erfüllen, wie schaut der Kursabschluss aus (Kursmitarbeit, Abgaben, Prüfung), etc.) werden erläutert.

Auch die Lernmechanismen werden besprochen: wie können Inhalte erlernt werden, welche Möglichkeiten (alleine, Gruppen, Email, Foren, Chats, etc.) stehen zur Verfügung. In einem Kick-Off Meeting von 4 Stunden umfasst dies 20-30 Minuten.

2.2.10. Präsentation Lernmaterial (Coach/Agent)

Falls nicht bereits bei der Präsentation der Plattform auf das zur Verfügung stehende (online und offline) Lernmaterial eingegangen wurde, soll dies hier präsentiert werden. Dabei sollen vor allem auch Hands-on Experience und Assessment-Teile angesprochen werden. In einem Kick-Off Meeting von 4 Stunden umfasst dies 10-15 Minuten.

2.2.11. Aufgabe lösen (Lernende)

Die Gruppen sollen sich nochmals zur Lösung einer gemeinsamen Aufgabe finden. Mögliche Aufgaben wären: Finden eines Gruppennamens, Kurzdarstellung der Gruppenmitglieder in das Gruppenforum der Plattform posten, Austausch von Kontaktadressen (Email), Festlegung der Gruppenregeln durch die Gruppenmitglieder. Die Gruppenmitglieder bestimmen ihre eigenen Regeln, wie sie während des Kurses miteinander arbeiten wollen, wie sie mit etwaigen „Trittbrettfahrern“ umgehen, etc. Die Gruppenbindung wird somit verstärkt und ein erster Umgang mit der Lernumgebung wird geübt. In einem Kick-Off Meeting von 4 Stunden umfasst dies 20 Minuten.

2.2.12. Abschluss (Coach)

Der Coach gibt ein Resümee. Zeitplan, Meilensteine, Aufgaben (für Einzelne und Gruppen) und Termine werden zusammengefasst wiedergegeben. In einem Kick-Off Meeting von 4 Stunden umfasst dies 10-15 Minuten.

2.2.13. Erläutern von noch offenen Fragen (Coach)

Falls noch Fragen offen sind, werden diese vom Coach beantwortet. Dauer etwa 5 Minuten.

2.2.14. Kick-Off wird beendet (Coach)

Danach beendet der Coach das Kick-Off Meeting mit motivierenden Worten und fordert die Lernenden auf, die Möglichkeit zum Feedback zu nutzen.

2.2.15. Evaluierung (Evaluierungsstelle/Agent)

Lernende und Coach können Feedback zum Kick-Off geben. Dies kann über Interviews oder Fragebögen erfolgen. Wird eine Evaluierung mittels Fragebögen gewählt kann die Durchführung und Auswertung durch Agenten erfolgen. Dauer: etwa 10 Minuten/Lernenden und Coach.

Anschließend werden die Prozesse „DE-Kursdurchführung“ und „Hilft bei technischen Anfragen und Problemen weiter“ initiiert.

2.2.16a. DE-Kursdurchführung

„DE-Kursdurchführung“ ist ein eigener Prozess der dem hier beschriebenen Prozess „Kick-Off “ nachfolgt.

2.2.16b. Hilft bei technischen Anfragen und Problemen weiter (Technischer Dienst)

„Hilft bei technischen Anfragen und Problemen weiter“ ist ein eigener Prozess, der dem hier beschriebenen Prozess „Kick-Off“ nachfolgt und parallel zum Prozess „DE-Kursdurchführung“ läuft.

2.3. DE-Kursdurchführung

Der Prozess „DE-Kursdurchführung“ umfasst folgende Prozessschritte:

1 Parallele Prozessschritte

1a1 Kursablauf (Coach)

1a2 Parallele Prozessschritte (entweder oder):

1a2a Initiierung des Prozesses „Kurs ist traditionell geführt mit DE Online-Unterstützung“

1a2b Initiierung des Prozesses „Kurs ist ein DE-Kurs im hybriden Modell (teilweise online, teilweise reale Treffen)“

1a2c Initiierung des Prozesses „Kurs ist reiner Distance Education Kurs (ohne reale Treffen)“

1b Technische Probleme (Technischer Dienst)

1c Wartung der DE-Plattform (Systemadministrator/Agent)

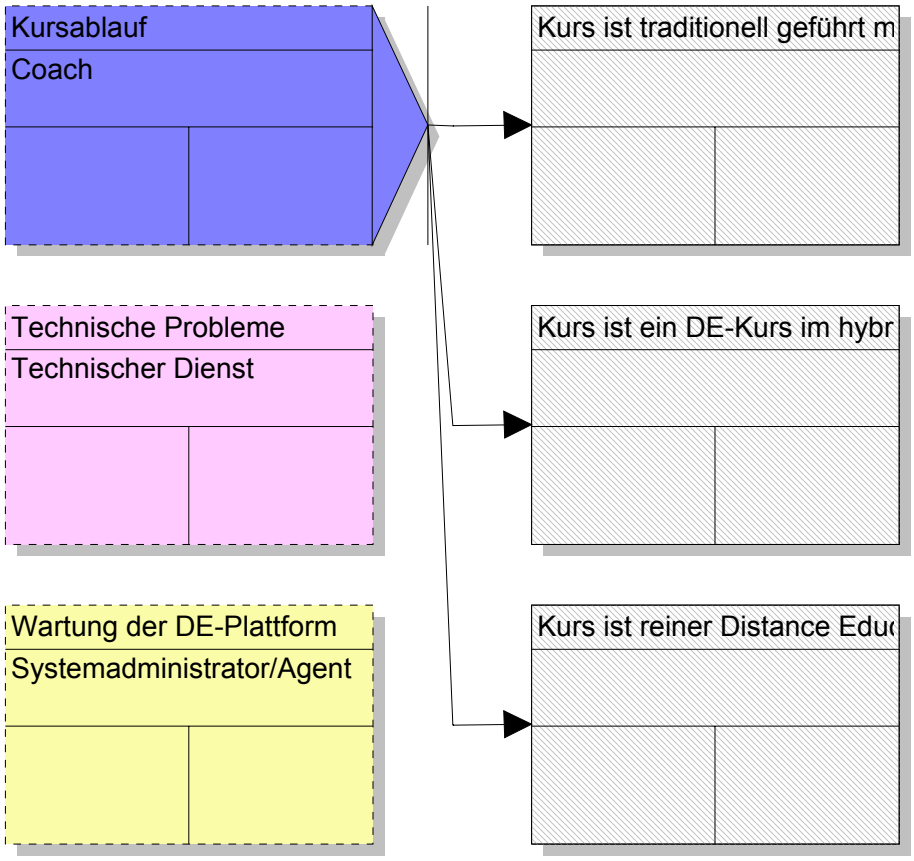


Abbildung 57: Prozessdiagramm „DE-Kursdurchführung“

2.3.1a1. Kursablauf (Coach)

DE-Kursform bestimmt den Kursablauf. Dabei wird hier unterschieden zwischen:

- Einem traditionell geführten Kurs, bei dem Distance Education unterstützend und begleitend eingesetzt wird.
- Einem hybriden Lernmodell mit Distance Coaching und Präsenzphasen und
- einem reinen Distance Education Kurs, der ausschließlich online geführt wird.

Anschließend wird entweder der Prozess „Kurs ist traditionell geführt mit DE Online-Unterstützung“ oder der Prozess „Kurs ist ein DE-Kurs im hybriden Modell (teilweise online, teilweise reale Treffen)“ oder der Prozess „Kurs ist reiner Distance Education Kurs (ohne reale Treffen)“ initiiert.

2.3.1a2a. Kurs ist traditionell geführt mit DE Online-Unterstützung

„Kurs ist traditionell geführt mit DE Online-Unterstützung“ ist einer von drei Prozessen (2.3.1a2a, 2.3.1a2b und 2.3.1a2c), die dem hier beschriebenen Prozess „DE-Kursdurchführung“ nachfolgen können.

2.3.1a2b. Kurs ist ein DE-Kurs im hybriden Modell (teilweise online, teilweise reale Treffen

„Kurs ist ein DE-Kurs im hybriden Modell (teilweise online, teilweise reale Treffen)“ ist einer von drei Prozessen (2.3.1a2a, 2.3.1a2b und 2.3.1a2c), die dem hier beschriebenen Prozess „DE-Kursdurchführung“ nachfolgen können.

2.3.1a2c. Kurs ist reiner Distance Education Kurs (ohne reale Treffen)

„Kurs ist reiner Distance Education Kurs“ ist einer von drei Prozessen (2.3.1a2a, 2.3.1a2b und 2.3.1a2c), die dem hier beschriebenen Prozess „DE-Kursdurchführung“ nachfolgen können.

2.3.1b. Technische Probleme (Technischer Dienst)

Der technische Dienst ist Ansprechpartner bei allen technischen Fragen zur Lernumgebung, Internetzugang, Hardware- und Softwareproblemen. Bei entsprechenden Anfragen und Problemen hilft hier der technische Support weiter.

2.3.1c. Wartung der DE-Plattform (Systemadministrator/Agent)

Im laufenden Betrieb muss die Plattform gewartet werden. Die Plattform und ihre Inhalte müssen gespeichert und repliziert werden, sodass bei einem Crash möglichst der Letztzustand wiederhergestellt werden kann. Auch müssen alte Beiträge archiviert, bzw. müssen Beiträge kategorisiert werden.

Diese Aufgaben können allesamt durch Agenten übernommen werden, die periodisch eine Speicherung der Plattform (Strukturen und Inhalte) vornehmen, Beiträge gemäß ihrer Zeitstempel falls gewünscht in Archive verschieben bzw. Beiträge mittels Schlagwörtern kategorisieren und bereits nach der Erstellung in die dafür vorgesehenen Foren stellen.

2.4. Kurs ist reiner Distance Education Kurs (ohne reale Treffen)

Der Prozess „Kurs ist reiner Distance Education Kurs (ohne reale Treffen)“ umfasst folgende Prozessschritte:

1 Parallele Prozessschritte
- 1a1 Lernen anhand des Lernmaterials (Lernende)
- 1a2 DE-Kursform bestimmt Kursablauf (Coach)
- 1a3 Parallele Prozessschritte:
  - 1a3a1 Gruppenaufgaben (Lernende)
  - 1a3a2 Optionale parallele Prozessschritte:
    - 1a3a2a Gruppenlösungen präsentieren (Gruppensprecher) (weiter zu Prozessschritt 1a1 oder 2)
    - 1a3a2b weiter zu Prozessschritt 1a3c1 (Gruppenfragen veröffentlichen)
  - 1a3b Gruppenkommunikation (Lernende) (weiter zu Prozessschritt 1a1)
  - 1a3c1 Gruppenfragen veröffentlichen (Gruppensprecher)
  - 1a3c2 Beantworten die Fragen der anderen Lernenden (Lernende)
  - 1a3c3 (Re)Aktion des Coach (Coach) (weiter zu Prozessschritt 1a3c1 oder 1a3d1 oder 1b oder 2)
  - 1a3d1 Arbeiten alleine an Aufgaben (Lernende)
  - 1a3d2 Optionale parallele Prozessschritte:
    - 1a3d2a Ergebnisse präsentieren (Lernende) (weiter zu Prozessschritt 1a1 oder 1a3e1 oder 2)
    - 1a3d2b weiter zu Prozessschritt 1a3c1 (Gruppenfragen veröffentlichen)
  - 1a3e1 Learning-By-Teaching (Lernende)
  - 1a3e2 Präsentieren Fragen an die anderen Gruppen (Gruppensprecher)
  - 1a3e3 Gruppenantworten erarbeiten (Lernende)
  - 1a3e4 Präsentieren ihre Antworten (Gruppensprecher)
  - 1a3e5 Bewertung/Korrektur (Lernende) (weiter 1a1 oder 1a3c1 oder 2)
- 1b Aufgaben (Coach/Tutoren/Agent) (weiter zu 1a1 oder 1a3c1 oder 2)
- 1c Chats (Coach)

2 Kurs abschließen (Coach/Agent)

3 Initiierung des Prozesses „DE-Kurs Abschluss“

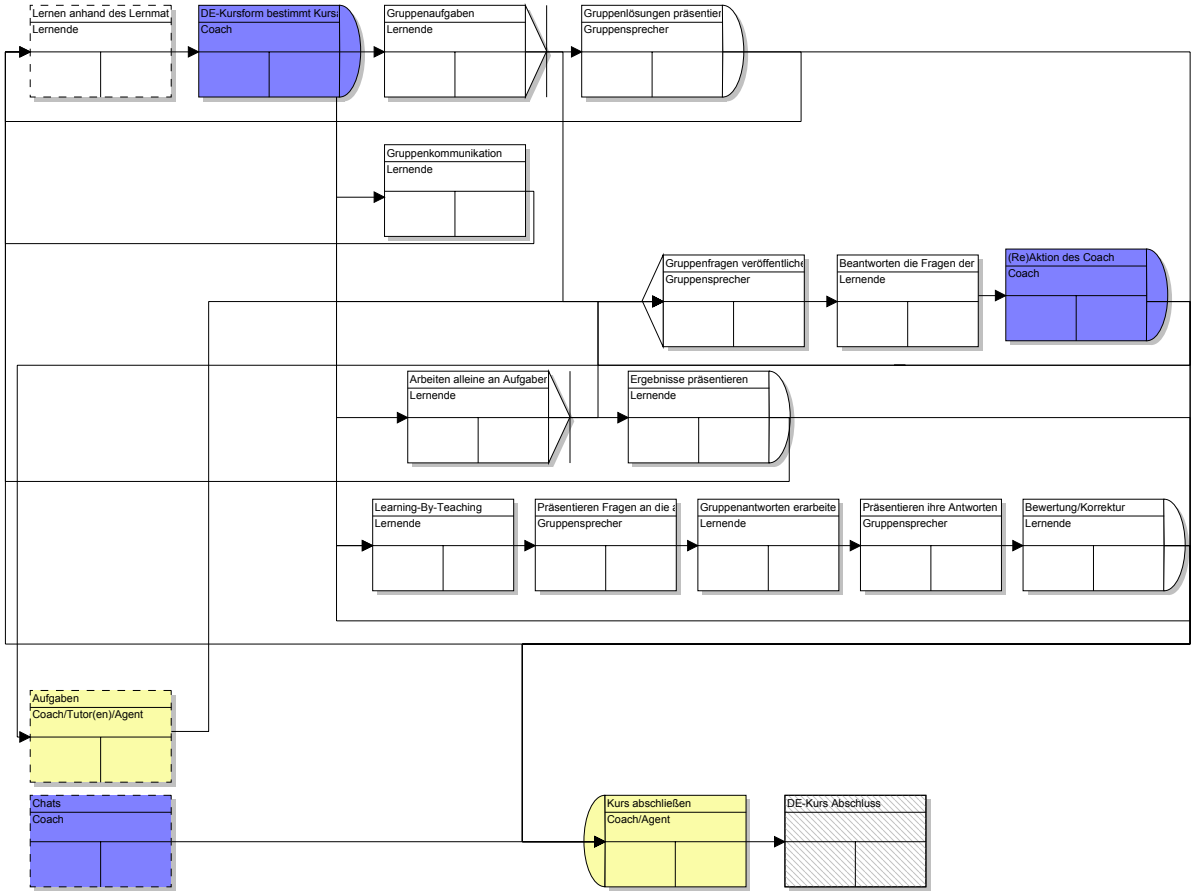


Abbildung 58: Prozessdiagramm „Kurs ist reiner Distance Education Kurs (ohne reale Treffen)“

2.4.1a1. Lernen anhand des Lernmaterials (Lernende)

Die Lernenden sollten während der gesamten Kursdauer das Recht haben, auf Lernmaterial innerhalb und falls auch Handouts verfügbar sind, auch außerhalb der Plattform zuzugreifen.

2.4.1a2. DE-Kursform bestimmt Kursablauf (Coach)

Die gewählte DE-Kursform bestimmt den Kursablauf. Hier betrifft dies vor allem die Möglichkeit bzw. die Verpflichtung zur Gruppenarbeit.

2.4.1a3a1. Gruppenaufgaben (Lernende)

Alle Gruppenaufgaben müssen auch gemeinsam in der Gruppe erarbeitet und ein gemeinsames Ergebnis präsentiert werden. Dabei arbeiten die Gruppen autonom ihren Regeln entsprechend, die sie im Kick-Off Meeting festgelegt haben.

2.4.1a3a2. Gruppenlösungen präsentieren (Gruppensprecher)

Die jeweiligen Gruppensprecher haben die Aufgabe die Gruppenlösungen auf der DE-Plattform zu präsentieren, sprich sie dem Coach oder den anderen Lernenden in aufbereiteter Form zugänglich zu machen.

2.4.1a3b. Gruppenkommunikation (Lernende)

Die Gruppen haben die Möglichkeit innerhalb ihrer Gruppenforen miteinander und mit dem Coach auf der DE-Plattform zu kommunizieren.

2.4.1a3c1 Gruppenfragen veröffentlichen (Gruppensprecher)

Falls innerhalb der Gruppe Fragen auftreten, die die Gruppenmitglieder nicht gemeinsam lösen können, werden diese von den Gruppensprechern öffentlich präsentiert, damit andere Lernende oder der Coach Hilfe leisten können.

2.4.1a3c2 Beantworten die Fragen der anderen Lernenden (Lernende)

Alle Lernende haben die Möglichkeit und sind angehalten auf die Fragen der anderen einzugehen und sich gegenseitig zu unterstützen. Dadurch beschäftigen sie sich nicht nur intensiver mit dem Lernstoff, sondern bekommen auch einen Einblick in andere Denkweisen oder Auffassungen des Lerninhalts.

2.4.1a3c3. (Re)Aktion des Coach (Coach)

Der Coach ist derjenige der sich erst zuletzt, wenn keine Lösung durch die Lernenden gefunden werden kann, oder die Diskussion in eine falsche Richtung geht, an der Diskussion aktiv beteiligt. Dann kann er falls nötig Antworten berichtigen, ergänzen oder neue Lösungswege aufzeigen. Der Coach ist auch derjenige der von Zeit zu Zeit neue Themen oder Aspekte anspricht und die Lernenden und die Gruppen motiviert.

Feedback durch den Coach ist sehr wichtig im Lernprozess. Um den Coach aber auch die Möglichkeit zu geben auf wirklich knifflige Fragen einzugehen, ist es notwendig Multilevel-Coaching einzusetzen. Dabei versuchen auf erster Ebene die Lernenden selbst (event. auch mit Hilfe eines Agenten) Antworten auf ihre Fragen zu finden, danach wird die Gruppe damit beschäftigt, danach alle Lernenden und erst abschließend wird auf den Coach zurückgegriffen. Der Coach ist nicht derjenige der in erster Linie Auskunftsstelle ist und quasi

als Almanach missbraucht wird. Die Lernenden sollen den Lernprozess vorantreiben und somit auf vielfachen Weg aus den Diskussionen profitieren. Oftmals hilft es schon Ideen oder Probleme schriftlich zu formulieren, um auf Lösungshinweise zu stoßen. Der Coach begleitet dabei die Lernenden und fungiert unter anderen als Impulsgeber.

2.4.1a3d1. Arbeiten alleine an Aufgaben (Lernende)

Lernende können auch außerhalb bzw. unabhängig von Gruppenforen in ihren eigenen Arbeitsverzeichnissen an Aufgaben arbeiten. Die Ergebnisse können dann durch File-Sharing oder durch Kopieren in allgemein zugängliche Bereiche anderen zugänglich gemacht werden.

2.4.1a3d2. Ergebnisse präsentieren (Lernende)

Die Lernenden haben jederzeit die Möglichkeit ihre Erkenntnisse, Ergebnisse oder Erfahrungen in die Lernlandschaft zu posten und somit anderen Information oder Anregungen zukommen zu lassen.

2.4.1a3e1. Learning-By-Teaching (Lernende)

Learning-By-Teaching bedeutet, dass die Lernenden für bestimmte Themen, die Rolle des Coach übernehmen. Gruppen entwickeln gemeinsam Fragen, die die anderen Gruppen beantworten müssen. Nachdem die anderen Gruppen die Antworten präsentiert haben, werden diese von der fragestellenden Gruppe gemeinsam bewertet und gegebenenfalls korrigiert. Dadurch müssen sich die Gruppenmitglieder detaillierter mit den Lerninhalten auseinander setzen, lernen andere Sichtweisen kennen und das Gelernte wird somit stärker verinnerlicht.

2.4.1a3e2. Präsentieren Fragen an die anderen Gruppen (Gruppensprecher)

Sind im Sinne von Learning-By-Teaching Fragen an andere Gruppen durch eine Gruppe ausgearbeitet worden, so stellt der Gruppensprecher diese in die entsprechenden DE-Plattform Foren, damit die anderen Gruppen darauf reagieren und diese bearbeiten können.

2.4.1a3e3. Gruppenantworten erarbeiten (Lernende)

Die Gruppenmitglieder arbeiten gemeinsam an den Antworten auf die Fragen der anderen Gruppen. Wichtig ist dabei gemeinsam zu einer Lösung zu kommen, die die jeweilige Frage hinreichend beantwortet und in der sich alle Gruppenmitglieder wiederfinden können.

2.4.1a3e4. Präsentieren ihre Antworten (Gruppensprecher)

Die Gruppensprecher posten die Antworten zu den korrespondieren Fragen.

2.4.1a3e5. Bewertung/Korrektur (Lernende)

Die Gruppenmitglieder bewerten und korrigieren gegebenenfalls die Antworten.

2.4.1b. Aufgaben (Coach/Tutor(en)/Agent)

Der Coach stellt während des Kursablaufs Aufgaben in die Plattform. Diese Aufgabe kann auch von einem Agenten durchgeführt werden, der wahlweise entsprechende Aufgaben aus einem Repository entnimmt und präsentiert.

2.4.1c. Chats (Coach)

Pilotstudien haben gezeigt, dass von Zeit zu Zeit Chattermine von den Lernenden sehr gut aufgenommen und als äußerst positiv und produktiv empfunden werden. Dabei nehmen jedenfalls alle Coaches und Tutoren teil und auch alle Lernenden sind eingeladen zum Chattermin online zu sein und am Chat teilzunehmen. Beim Chat können die Lernenden direkt mit dem Coach kommunizieren und Fragen stellen, die dieser sofort beantwortet.

2.4.2. Kurs abschließen (Coach/Agent)

Nach Ende der Kursdauer ist der Kurs als solches abgeschlossen. Der Termin für die Abschlussveranstaltung wird den Lernenden bekannt gegeben. Anschließend wird der Prozess „DE-Kurs Abschluss“ initiiert.

2.4.3. DE-Kurs Abschluss

Der Prozess „DE-Kurs Abschluss“ folgt dem Prozess „Kurs ist reiner Distance Education Kurs (ohne reale Treffen)“ nach.

2.5. Kurs ist ein DE-Kurs im hybriden Modell (teilweise online, teilweise reale Treffen)

Der Prozess „Kurs ist ein DE-Kurs im hybriden Modell (teilweise online, teilweise reale Treffen)“ umfasst folgende Prozessschritte:

1 Wiederhole, solange bis Kurs abgeschlossen

1a Veranstaltet Treffen (Coach)

1b Initiiere Prozess „Kurs ist reiner Distance Education Kurs (ohne reale Treffen)“

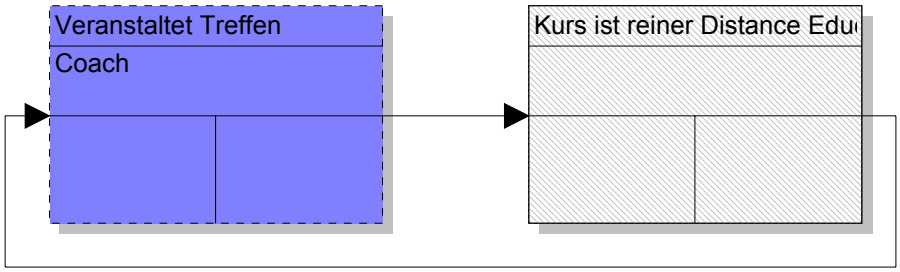


Abbildung 59: Prozessdiagramm „Kurs ist ein DE-Kurs im hybriden Modell (teilweise online, teilweise reale Treffen)“

2.5.1a. Veranstaltet Treffen (Coach)

Zusätzlich zum Vorgehensmodell des reinen Distance Education Kurses gibt es beim hybriden Modell auch reale Treffen, sprich Präsenzveranstaltungen. Je nach Kursdauer kann die Anzahl dieser Präsenzveranstaltungen variieren. Jedenfalls empfiehlt es sich aber das Kick-Off Meeting (gegenseitiges Kennenlernen, Gruppenbildung, Motivation) und das Abschlusstreffen (Resümee, Feedback) als Präsenzveranstaltungen abzuhalten. Bei den übrigen Treffen kann muss aber nicht auf Lerninhalte eingegangen werden. Diese können dazu dienen, um zusätzliche Motivation zu schaffen, um Wünsche und Vorstellungen der Lernenden besser erfassen und umsetzen zu können, etc.

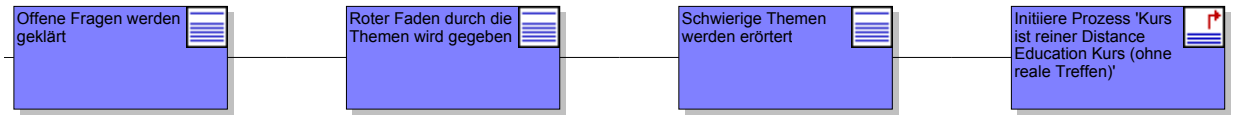


Abbildung 60: Detaildiagramm „Veranstaltet Treffen“

Anschließend wird der Prozess „Kurs ist reiner Distance Education Kurs (ohne reale Treffen)“ initiiert.

2.5.1b. Kurs ist reiner Distance Education Kurs (ohne reale Treffen)

Der Prozess „Kurs ist reiner Distance Education Kurs (ohne reale Treffen)“ ist im Prozess „Kurs ist DE-Kurs im hybriden Modell (teilweise online, teilweise reale Treffen)“ eingebettet.

2.6. Kurs ist traditionell geführt mit DE Online-Unterstützung

Der Prozess „Kurs ist traditionell geführt mit DE Online-Unterstützung“ umfasst folgende Prozessschritte:

1 Parallele Prozessschritte
- 1a1 Veranstaltet periodisch reale Treffen (Coach)
- 1a2 Parallele Prozessschritte (entweder oder)
  - 1a2a Kommunizieren auf der DE-Plattform (Coach/Lernende) (weiter zu Prozessschritt 1a1 oder 2)
  - 1a2b Arbeiten auf der DE-Plattform (Lernende) (weiter zu Prozessschritt 1a1 oder 2)
  - 1a2c Aufgaben (Coach/Tutor(en)/Agent) (weiter zu Prozessschritt 1a1 oder 2)
  - 1a2d1 Stellen Aufgaben auf die DE-Plattform (Lernende) (weiter zu Prozessschritt 1a1 oder 1a2d2)
  - 1a2d2 Bewertet Aufgaben (Coach/Tutor(en)/Agent) (weiter zu Prozessschritt 1a1 oder 2)
  - 1a2e Nutzung zusätzlicher Informationen (Lernende) (weiter zu Prozessschritt 1a1 oder 2)
- 1b Chats (Coach) (weiter zu Prozessschritt 1a1 oder 2)

2 Kurs abschließen (Coach/ Agent)

3 Initiierung des Prozesses „DE-Kurs Abschluss“

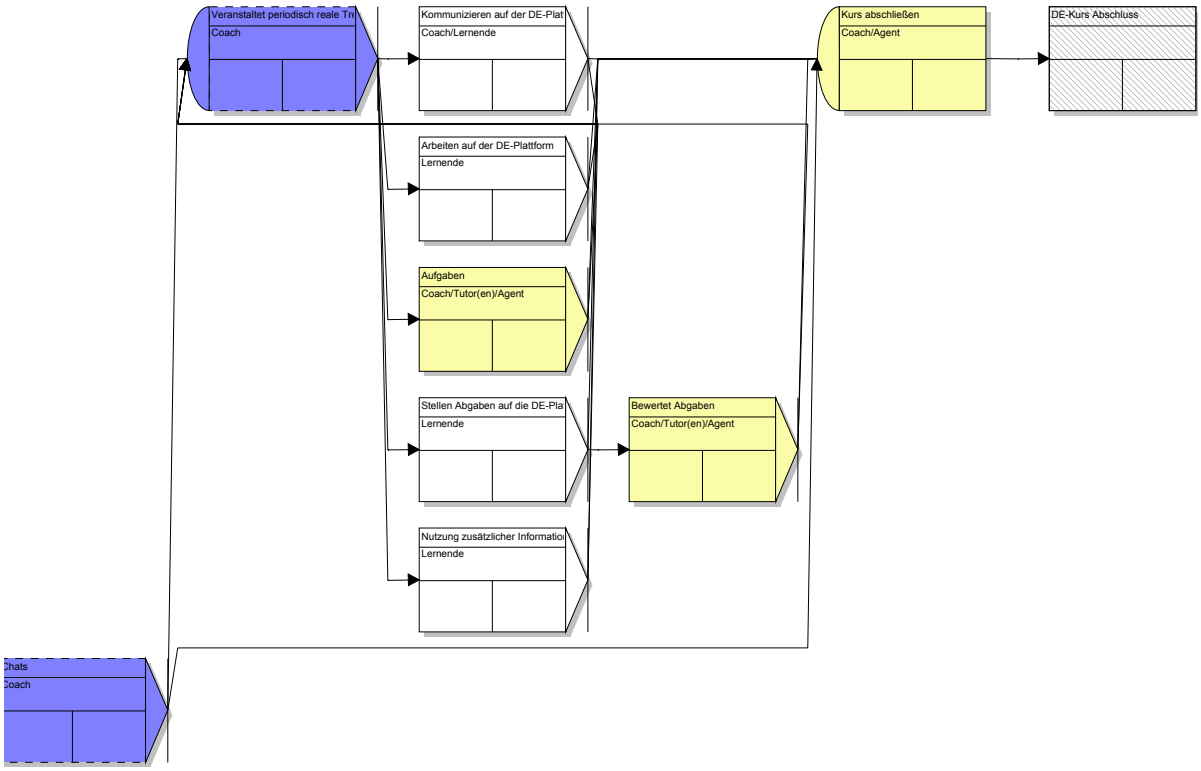


Abbildung 61: Prozessdiagramm „Kurs ist traditionell geführt mit DE Online-Unterstützung“

2.6.1a1. Veranstaltet periodisch reale Treffen (Coach)

Periodisch werden hier Präsenzveranstaltungen abgehalten. Jedenfalls empfiehlt es sich aber das Kick-Off Meeting (gegenseitiges Kennenlernen, Gruppenbildung, Motivation) und das Abschlusstreffen (Resümee, Feedback) als Präsenzveranstaltungen abzuhalten. Bei den übrigen Treffen kann muss aber nicht auf Lerninhalte eingegangen werden. Diese können dazu dienen, um zusätzliche Motivation zu schaffen, um Wünsche und Vorstellungen der Lernenden besser erfassen und umsetzen zu können, etc.

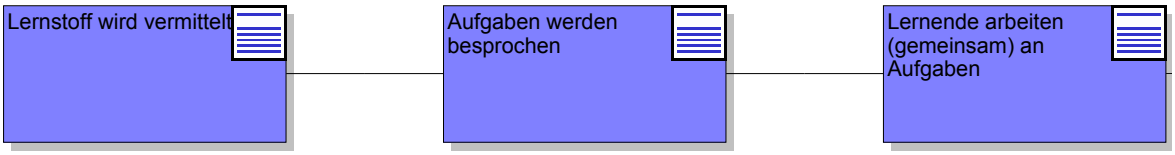


Abbildung 62: Detaildiagramm „Veranstaltet periodisch reale Treffen“

2.6.1a2a. Kommunizieren auf der DE-Plattform (Coach/Lernende)

Die Lernenden können die Kommunikationswerkzeuge nützen, um auch außerhalb der Treffen zu kommunizieren und um gemeinsam Aufgaben zu lösen.

2.6.1a2b. Arbeiten auf der DE-Plattform (Lernende)

Die Lernenden können gemeinsam und event. auch in Gruppen auf der DE-Plattform arbeiten. Dabei können sie in Verzeichnissen arbeiten auf die sie die gleichen Zugriffsrechte haben oder sie geben Dateien oder Verzeichnisse innerhalb ihrer Arbeitsbereiche zum Zugriff (shared folder) frei.

2.6.1a2c. Aufgaben (Coach/Tutor(en)/Agent

Der Coach stellt während des Kursablaufs Aufgaben in die Plattform. Diese Aufgabe ist auch von einem Agenten durchführbar, der wahlweise entsprechende Aufgaben aus einem Repository entnimmt und präsentiert.

2.6.1a2d1. Stellen Abgaben auf die DE-Plattform (Lernende)

Auch Lernende können unterschiedliche Aufgaben auf die Plattform stellen. Dabei kann auch Learning-By-Teaching zum Einsatz kommen.

2.6.1a2d2. Bewertet Abgaben (Coach/Tutor(en)/Agent)

Der Coach (Tutoren) oder der Agent bewertet die Abgaben, je nach Form in der die Abgaben vorliegen und geben Feedback.

2.6.1a2e. Nutzung zusätzlicher Information (Lernende)

Die Plattform bietet den Lernenden die Möglichkeit über das "normale" Lernmaterial hinaus, zusätzliches Lernmaterial, wie Self-Assessment, Hands-on Experience oder Wörterbücher, Literaturverzeichnisse, etc. zu nutzen.

2.6.1b. Chats (Coach)

Parallel zu den Kursaktivitäten, wie Aufgaben, Diskussionen, Treffen können als Bereicherung und Auflockerung auch hier Chattermine nach Bedarf durchgeführt werden.

2.6.2. Kurs abschließen (Coach/Agent)

Nach Ende der Kursdauer ist der Kurs als solches abgeschlossen. Der Termin für die Abschlussveranstaltung wird den Lernenden bekannt gegeben.

Anschließend wird der Prozess „DE-Kurs Abschluss“ initiiert.

2.6.3. DE-Kurs Abschluss

„DE-Kurs Abschluss“ ist ein eigener Prozess, der dem hier beschriebenen Prozess „Kurs ist traditionell geführt mit DE Online-Unterstützung“ nachfolgt.

2.7. DE-Kurs Abschluss

Der Prozess „DE-Kurs Abschluss“ umfasst folgende Prozessschritte:

1 Legt Abschluss des DE-Kurses fest (Coach)

2 Parallele Prozessschritte

- 2a Präsenzprüfung (Coach/Agent)
- 2b Online-Prüfung (Coach/Agent)
- 2c Mitarbeit/Abgaben (weiter bei Prozessschritt 4) (Coach/Agent)

3 Modalitäten (Coach/Agent)

4 Initiierung des Prozesses „Abschlussveranstaltung“

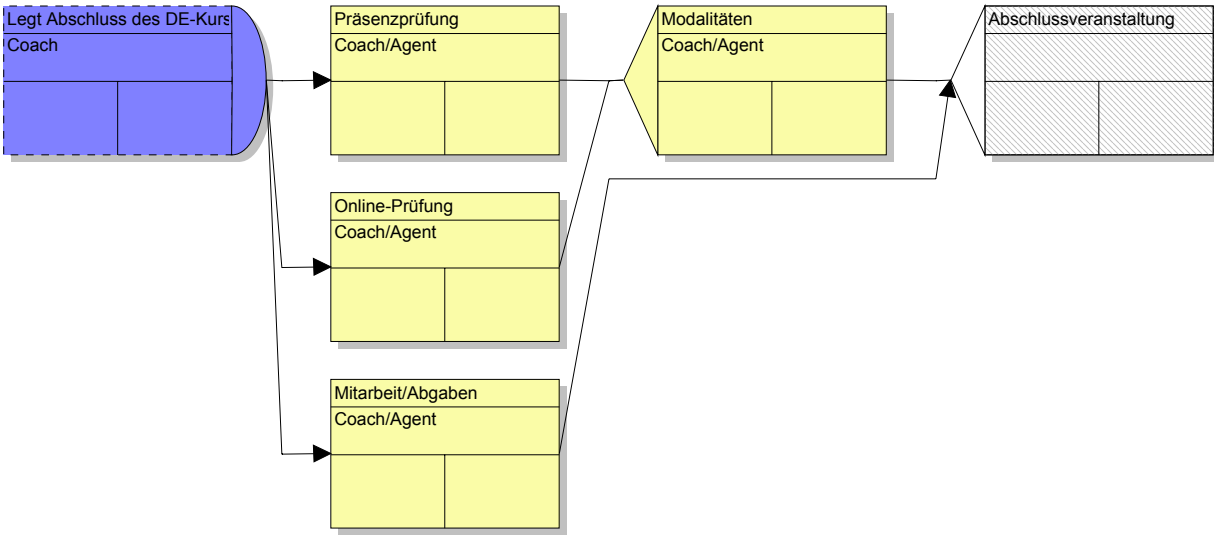


Abbildung 63: Prozessdiagramm „DE-Kurs Abschluss“

2.7.1. Legt Abschluss des DE-Kurses fest (Coach)

Der Coach hat bereits vor Beginn des Kurses festgelegt, wie der formale Abschluss des Kurses aussieht. Sprich, wird eine Prüfung stattfinden, wird die Note aus der Mitarbeit und den Abgaben ermittelt, etc.

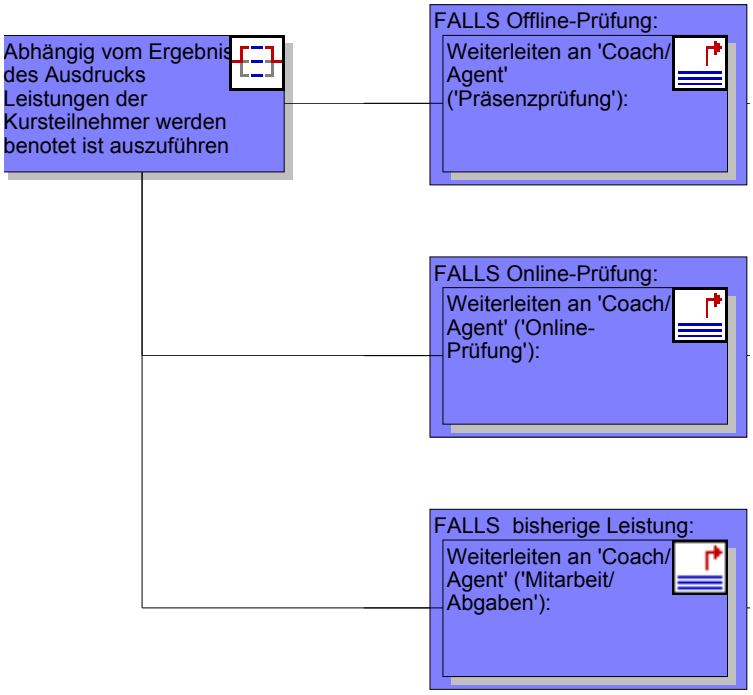


Abbildung 64: Detaildiagramm „Legt Abschluss des DE-Kurses fest“

Leistung der Lernenden benoten:

- **IF** eine Benotung durch eine Offline-Prüfung erfolgt, **THEN** initiiere Prozessschritt „Präsenzprüfung“.
- **IF** eine Benotung durch eine Online-Prüfung erfolgt, **THEN** initiiere Prozessschritt „Online-Prüfung“.
- **IF** die Benotung durch die bisherige Leistung erfolgt, **THEN** initiiere Prozessschritt „Mitarbeit/Abgaben“.

2.7.2a. Präsenzprüfung (Coach/Agent)

Wird die Prüfung als Präsenzveranstaltung abgehalten, wird der Termin festgelegt und Ort der Prüfung reserviert.

2.7.2b. Online-Prüfung (Coach/Agent)

Wird die Prüfung online abgehalten, muss dafür ein Termin bestimmt werden.

2.7.2c. Mitarbeit/Abgaben (Coach/Agent)

Die Note oder Teilnoten können auch aus der Mitarbeit und der Benotung der Ausarbeitungen, der Beiträge, etc. ermittelt werden.

2.7.3. Modalitäten (Coach/Agent)

Der Coach legt alle Modalitäten (darunter auch Mitarbeiter zur Prüfungsaufsicht, etc.) fest, sodass diese in der Abschlussveranstaltung bekannt gegeben werden können.

Anschließend wird der Prozess „Abschlussveranstaltung“ initiiert.

2.7.4. Abschlussveranstaltung

„Abschlussveranstaltung“ ist ein eigener Prozess, der dem hier beschriebenen Prozess „DE-Kurs Abschluss“ nachfolgt.

2.8. Abschlussveranstaltung

Der Prozess „Abschlussveranstaltung“ umfasst folgende Prozessschritte:

1 Startet Abschlussveranstaltung (Coach)
2 Offene Fragen klären (Coach)
3 Resümiert den DE-Kurs (Coach)
4 Resümieren ihrerseits den DE-Kurs (Coach)
5 Modalitäten des Kursabschlusses (Coach)
6 Optionaler Prozessschritt: Weitere Vorgehensweise (Coach/Agent)
7 Abschluss der Abschlussveranstaltung (Coach)
8 Evaluierung Abschlussveranstaltung (Evaluierungsstelle/Agent)
9 Initiierung des Prozesses „Prüfung/Zertifizierung“

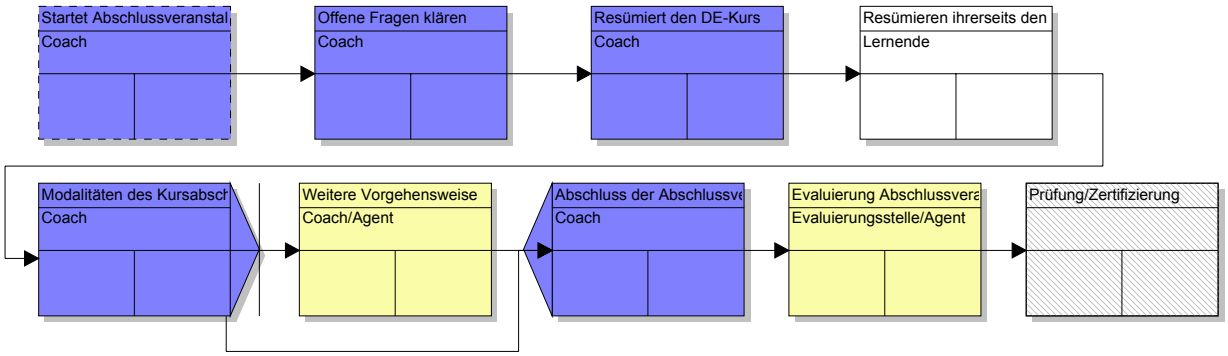


Abbildung 65: Prozessdiagramm „Abschlussveranstaltung“

2.8.1. Startet Abschlussveranstaltung (Coach)

Die Abschlussveranstaltung sollte, wenn möglich, wie auch das Kick-Off Meeting, als Präsenzveranstaltung durchgeführt werden. Dies ermöglicht einen persönlichen Abschluss und erleichtert die Motivation hinsichtlich eines Folgekurses.

2.8.2. Offene Fragen klären (Coach)

Falls aus der Onlinediskussion oder zum Kursabschluss noch Fragen offen sind, sollen diese hier gleich zu Beginn geklärt werden.

2.8.3. Resümiert den DE-Kurs (Coach)

Der Coach zieht ein Resümee aus seiner Sicht über den Distance Education Kurs (den Ablauf, die Beteiligung der Teilnehmer, die Beiträge, etc.). Spricht dabei Negatives (event. unter Angabe von Verbesserungsvorschlägen) und vor allem Positives nochmals an und beleuchtet die Resultate in Zusammenhang mit den ursprünglich gesteckten Zielen.

2.8.4. Resümieren ihrerseits den DE-Kurs (Lernende)

Auch die Lernenden sollen die Möglichkeit zum Feedback bekommen und ihre Meinung über den Kurs darstellen können.

2.8.5. Modalitäten des Kursabschlusses (Coach)

Der Coach teilt mit, ob eine Prüfung den Abschluss des Kurses bietet. Falls ja, werden die Modalitäten (Termin, Ort, Art der Prüfung, etc.) erläutert. Die Abschlussveranstaltung bildet die Brücke zur Prüfungsvorbereitung und zur Prüfung.

2.8.6. Weitere Vorgehensweise (Coach/Agent)

Die Modalitäten und die weitere Vorgangsweise (offizielles Ende des Kurses, Kurszertifikat, etc.) werden bekannt gegeben, sodass alle Lernenden darüber Kenntnis haben.

2.8.7. Abschluss der Abschlussveranstaltung (Coach)

Der Coach verabschiedet die Lernenden, motiviert für die bevorstehende Prüfung, informiert über eventuelle Folgekurse und bedankt sich bei den Lernenden und Mitarbeitern.

2.8.8. Evaluierung Abschlussveranstaltung (Evaluierungsstelle/Agent)

Lernende und Coach können Feedback zur Abschlussveranstaltung geben. Dies kann über Interviews oder Fragebögen erfolgen. Wird eine Evaluierung mittels Fragebögen gewählt kann die Durchführung und Auswertung durch Agenten erfolgen. Anschließend wird der Prozess „Prüfung/Zertifizierung“ initiiert.

2.8.9. Prüfung/Zertifizierung

Der Prozess „Prüfung/Zertifizierung“ folgt dem Prozess „Abschlussveranstaltung“ nach.

2.9. Prüfung/Zertifizierung

Der Prozess „Prüfung/Zertifizierung“ umfasst folgende Prozessschritte:

1 Zusammenstellung der Prüfung (Coach/Agent)
2 Legen Prüfung ab (Lernende)
3 Prüfungsbewertung/Notenvergabe (Coach/Agent)
4 Bekanntgabe der Noten (Coach/Agent)
5 Verabschiedung (Coach/Agent)
6 Erteilt die Abschlussbescheide (Zertifizierungsstelle)
7 Evaluierung Kurs (Coach/Agent)
8 Initiierung des Prozesses „Kursnachbereitung“

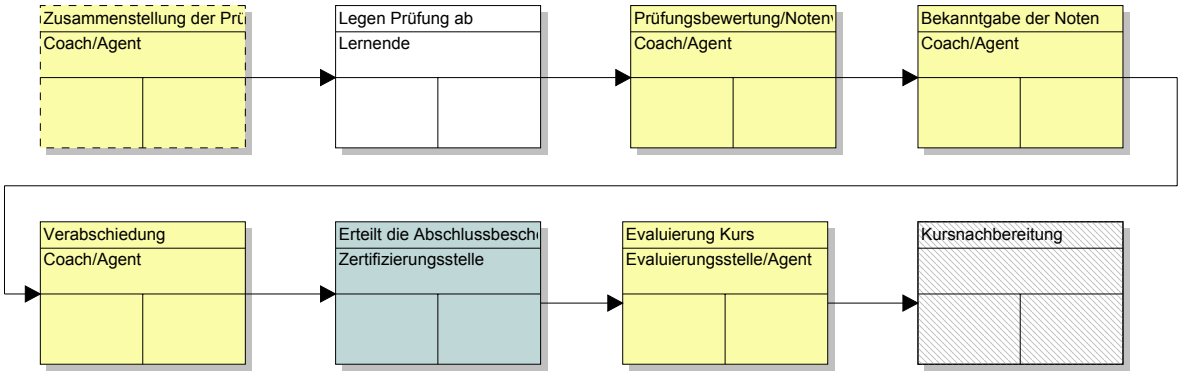


Abbildung 66: Prozessdiagramm „Prüfung/Zertifizierung“

2.9.1. Zusammenstellung der Prüfung (Coach/Agent)

Die Prüfungsfragen werden vom Coach oder auch von einem Agenten zusammengestellt. Ein Agent stellt hier gemäß den Lerninhalten des Kurses aus einem Prüfungsrepository aufgrund von Metadaten passende Prüfungsfragen zusammen. Der Coach greift dann auf einen solchen Prüfungsvorschlag zurück und falls gewünscht modifiziert er diesen.

2.9.2. Legen Prüfung ab (Lernende)

Die Lernenden legen in der festgelegten Zeit (am festgelegten Ort) ihre Prüfung ab. Diese kann Offline (mündlich oder schriftlich) oder online (schriftlich) erfolgen.

2.9.3. Prüfungsbewertung/Notenvergabe (Coach/Agent)

Die Prüfungen werden bewertet und Noten entsprechend den Ergebnissen ermittelt.

2.9.4. Bekanntgabe der Noten (Coach/Agent)

Die Bekanntgabe der Noten erfolgt in der Regel schriftlich innerhalb eines Forums der Plattform. Liegen die Ergebnisse elektronisch vor, ist diese Aufgabe auch von einem Agenten durchführbar.

2.9.5. Verabschiedung (Coach/Agent)

Ist dies nicht bereits in der Abschlussveranstaltung erfolgt, werden hier die Lernenden durch ein entsprechendes Posting verabschiedet. Dieses Posting, in dem auch über event. Folgekurse informiert wird, zu erstellen und zu veröffentlichen ist auch durch Agenten denkbar.

2.9.6. Erteilt die Abschlussbescheide (Zertifizierungsstelle)

Die zuständige Zertifizierungsstelle kann nach Erhalt der Prüfungsergebnisse die entsprechenden Zertifikate, etc. ausstellen.

2.9.7. Evaluierung Kurs (Evaluierungsstelle/Agent)

Lernende und Coach geben Feedback zum Kurs. Dies erfolgt über Interviews, Fragebögen, etc. Wird eine Evaluierung mittels Fragebögen gewählt, kann die Durchführung und Auswertung durch Agenten erfolgen.

Anschließend wird Prozess „Kursnachbereitung“ initiiert.

2.9.8. Kursnachbereitung

„Kursnachbereitung“ ist ein eigener Prozess, der dem hier beschriebenen Prozess „Prüfung/Zertifizierung“ nachfolgt.

3 Teil 3: Kursnachbereitung/Improvement

Hier wird der Kurs bzw. die Kursergebnisse „eingefroren“ und die Ergebnisse der Evaluierung fließen in die Umsetzung des Lernmaterials, didaktischen Modells, etc. ein.

3.1. Kursnachbereitung

Der Prozess „Kursnachbereitung“ umfasst folgende Prozessschritte:

1 Schließt DE-Kurs ab (Coach)
2 Parallele Prozessschritte:
  2a1 DE-Kurs Evaluierung (Evaluierungsstelle/Agent)
  2a2 Stepwise Improvement (Kursorganisator)
  2a3 Optionale parallele Prozessschritte:
    2a3a Ergebnisse der Evaluierung im didaktischen Modell umsetzen (Modellgestaltung)
    2a3b Ergebnisse der Evaluierung im Lernmaterial umsetzen (Materialgestaltung)
  2b Kursinhalte einfrieren (Kursorganisator/Agent)
3 Abschluss des Kurses (Kursorganisator)

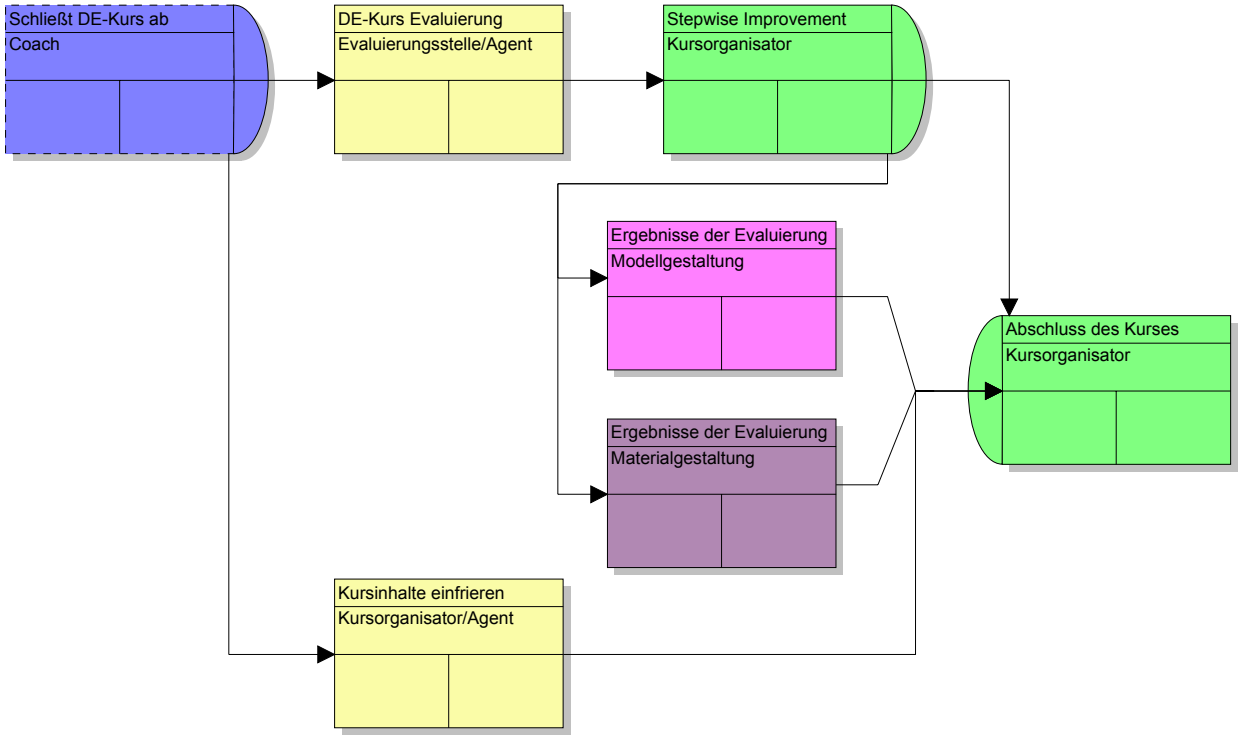


Abbildung 67: Prozessdiagramm „Kursnachbereitung“

3.1.1. Schließt DE-Kurs ab (Coach)

Der Coach schließt den Kurs ab. Dabei kann die Plattform und ihre Inhalte für bestimmte Zeit den Lernenden weiterhin zugänglich sein, etwa wenn es die Möglichkeit einer Nachklausur gibt oder dergleichen. Einzelne Bereiche, wie Diskussionsforen und Arbeitsverzeichnisse können dabei eingefroren werden, d.h. es sind nur mehr Lesezugriffe möglich. Dies kann für alle Bereiche der Plattform und für alle Teilnehmer von Plattformagenten übernommen werden.

Eine weitere Variante ist den Kurs, dessen Inhalte und die Beiträge den Lernenden auf CD-Rom zur Verfügung zu stellen. Dabei muss die Plattform, die eingebettete Kursstruktur (Verzeichnisse, Foren, etc.) und die Inhalte offline abgebildet und kopiert werden. Dies ist sehr aufwendig und langwierig und bietet sich als Aufgabe für einen Agenten an. Ein Agent kann dabei eine exakte Kopie der Bereiche erstellen, die auf CD-Rom zur Verfügung gestellt werden sollen und dies beliebig vervielfältigen.

3.1.2a1. DE-Kurs Evaluierung (Evaluierungsstelle/Agent)

Der Distance Education Kurs wird evaluiert. Meinungen der Coaches/Tutoren und Lernenden, Nutzung der Kommunikationswege, des Lernmaterials, etc. werden ausgewertet. Falls eine Notenbewertung existiert sollte diese in die Evaluierung miteinbezogen werden. Audio- und Videoaufnahmen müssen dabei durch die Evaluierungsstelle selbst bewertet werden. Aber vor allem Fragebögen und Prüfungsergebnisse können statistisch durch den Einsatz von Agenten ausgewertet werden.

3.1.2a2. Stepwise Improvement (Kursorganisator)

Die Ergebnisse der Evaluierung sollten in eine schrittweise Verbesserung des Kurses einfließen. Dabei ergeben sich eventuell Verbesserungen des didaktischen Modells, des Lernmaterials, der Lernlandschaft, etc. Etwa ist es möglich, dass sich zeigt, dass das Lernmaterial in Teilaspekten ergänzt und detailliert werden sollte, dass kleinere oder größere Gruppen nötig oder möglich sind, dass der Einsatz von Tutoren unterbleiben kann oder in jedem Fall zwingend ist, etc.

3.1.2a3a. Ergebnisse der Evaluierung im didaktischen Modell umsetzen (Modellgestaltung)

Die Evaluierung des didaktischen Modells kann etwa ergeben, dass manche Präsenztermine entfallen und statt dessen mehr Chattermine mit Lernenden und Coach gemeinsam stattfinden sollten. Möglicherweise stellt sich heraus, dass die Gruppengröße anders gewählt werden sollte oder die Anzahl der Tutoren verändert werden muss.

3.1.2a3b. Ergebnisse der Evaluierung im Lernmaterial umsetzen (Materialgestaltung)

Möglich wäre, dass sich herausstellt, dass Teile des Lernmaterials nie genutzt werden, dies kann der Fall sein, wenn die Zielgruppe besseres Vorwissen als angenommen hatte. Bei Teilen des Lernstoffs können verstärkt Fragen aufgetreten sein. Für diese Bereiche würde sich eine detailliere Aufbereitung oder eine Hands-on Experience-Unterstützung anbieten.

3.1.2b. Kursinhalte einfrieren (Kursorganisator/Agent)

Der Kurs und seine Inhalte werden nach Abschluss des Kurses "eingefroren". Dabei können die Zugriffrechte auf alle Kursobjekte (Foren, Inhalte, Verzeichnisse, etc.) auf Leserechte vermindert werden. Auch ist es denkbar, den gesamten Kurs zu kopieren und ihn auf CD-Rom den Lernenden zur Verfügung zu stellen. Diese Arbeiten können zeitsparend von Agenten übernommen werden.

3.1.3. Abschluss des Kurses (Kursorganisator

Nachdem die Plattform geschlossen, der Kurs evaluiert und bearbeitet wurde, gilt der Kurs entgültig als abgeschlossen.

Diagramme

1 Organigramm

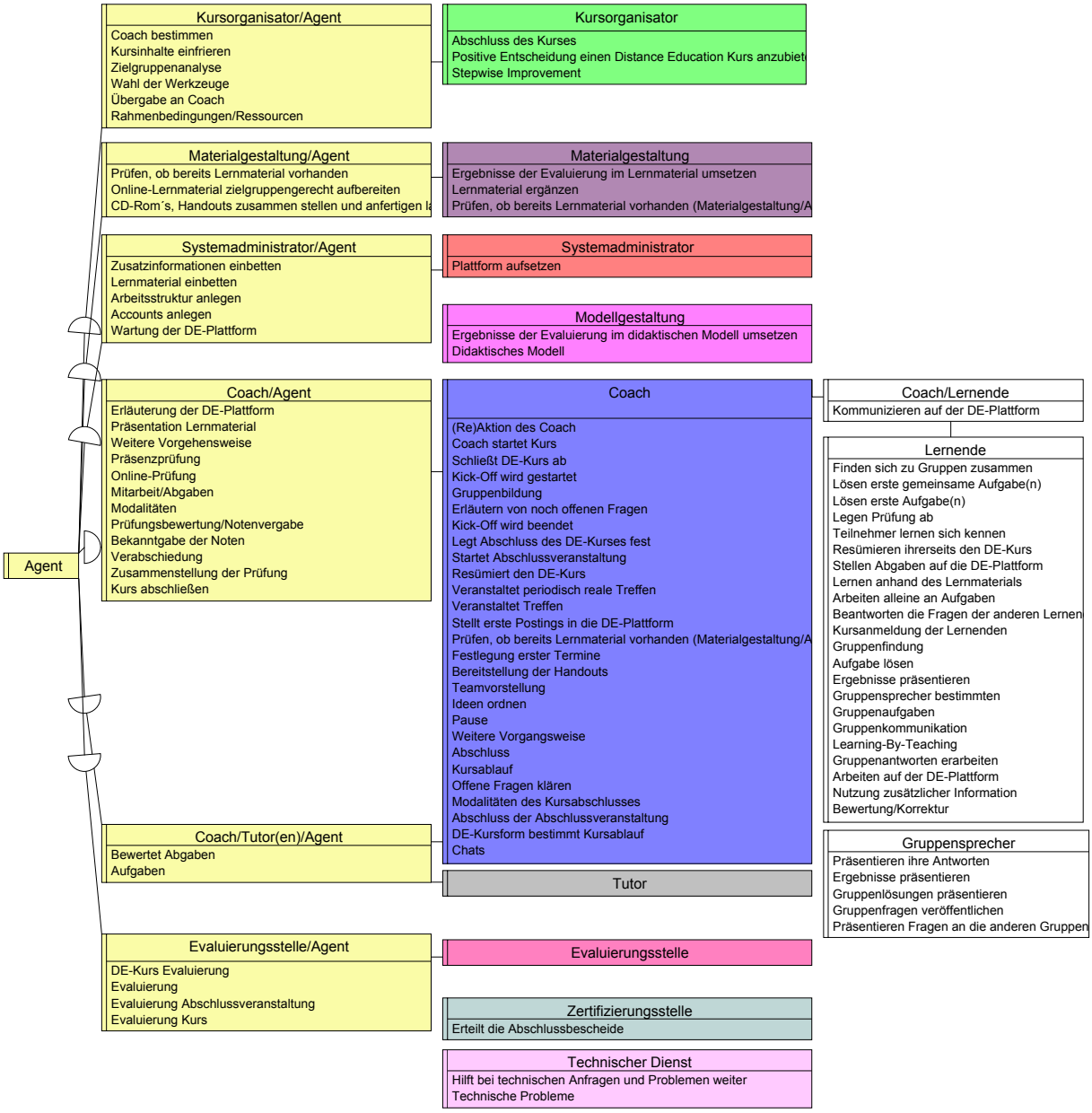


Abbildung 68: Rollen im Prozessmodell Distance Education Kurs

1.1. Agent

Agenten sind Softwareprogramme, die unterschiedliche Aufgaben übernehmen können. Besonders bieten sich dafür natürlich zeitraubende oder immer wiederkehrende Aufgaben an.

Zu folgenden Teams/Gremien besteht eine Zugehörigkeit:

- Kursorganisator/Agent
- Coach/Tutor(en)/Agent
- Systemadministrator/Agent
- Coach/Agent
- Materialgestaltung/Agent
- Evaluierungsstelle/Agent

1.2. Lernende

Die Lernenden sind die Kursteilnehmer, die im Sinne der Selbstorganisation lernen. Die Lernenden arbeiten aktiv und selbstgesteuert im Lernprozess. Sie sind nicht mehr passive Konsumenten von Wissen, sondern aktive Partner. Die Lernqualität und vor allem die Nachhaltigkeit soll durch Distance Education verbessert werden.

Von Inhabern dieser Rolle können folgende Aufgaben ausgeführt werden:

- Finden sich zu Gruppen zusammen
- Lösen erste Aufgabe(n)
- Teilnehmer lernen sich kennen
- Stellen Abgaben auf die DE-Plattform
- Arbeiten alleine an Aufgaben
- Nutzung zusätzlicher Information
- Gruppenfindung
- Ergebnisse präsentieren
- Gruppenaufgaben
- Learning-By-Teaching
- Beantworten Fragen der Anderen
- Lösen erste gemeinsame Aufgabe(n)
- Legen Prüfung ab
- Resümieren ihrerseits den DE-Kurs
- Lernen anhand des Lernmaterials
- Bewertung/Korrektur
- Kursanmeldung der Lernenden
- Aufgabe lösen
- Gruppensprecher bestimmen
- Gruppenkommunikation
- Gruppenantworten erarbeiten
- Arbeiten auf der DE-Plattform

Zu folgenden Teams/Gremien besteht eine Zugehörigkeit:

- Coach/Lernende

1.3. Gruppensprecher

Gruppensprecher werden von einer Gruppe gewählt, bzw. von den Gruppenmitgliedern bestimmt. Sie vertreten die Gruppe nach außen.

Von Inhabern dieser Rolle können folgende Aufgaben ausgeführt werden:

- Präsentieren ihre Antworten
- Ergebnisse präsentieren

- Gruppenlösungen präsentieren
- Gruppenfragen veröffentlichen
- Präsentieren Fragen an die anderen Gruppen

1.4. Coach

Der Coach führt die Lernenden durch den Lernprozess. Er ist derjenige der sich einerseits als primäre Ansprechperson (Paradigmenwechsel: vom Lehrer vom Coach) zurücknimmt, beobachtet und die Lernenden in den Vordergrund stellt, der aber auch gezielt eingreift, interveniert, motiviert und animiert.

Von Inhabern dieser Rolle können folgende Aufgaben ausgeführt werden:

- Coach startet Kurs
- (Re)Aktion des Coach
- Stellt erste Postings in die DE-Plattform
- Prüfen, ob Lernmaterial vorhanden
- Festlegung erster Termine
- Bereitstellung der Handouts
- Kick-Off wird gestartet
- Gruppenbildung
- Teamvorstellung
- Ideen ordnen
- Pause
- Weitere Vorgangsweise
- Erläutern von noch offenen Fragen
- Kick-Off wird beendet
- Veranstaltet periodisch reale Treffen
- Veranstaltet Treffen
- Startet Abschlussveranstaltung
- Resümiert den DE-Kurs
- Abschluss
- Kursablauf
- Offene Fragen klären
- Modalitäten des Kursabschlusses
- Abschluss der Abschlussveranstaltung
- DE-Kursform bestimmt Kursablauf
- Chats
- Schließt DE-Kurs ab
- Legt Abschluss des DE-Kurses fest
- Termin wird festgelegt
- Ermittelt Benotung aus den Ausarbeitungen, Beiträgen, etc.
- Termin wird festgelegt und Ort der Prüfung reserviert

Zu folgenden Teams/Gremien besteht eine Zugehörigkeit:

- Coach/Lernende
- Coach/Tutor(en)/Agent
- Coach/Agent

1.5. Coach/Agent

Die Rolle Coach/Agent drückt aus, dass die Aufgaben die hier beschrieben sind entweder von einem Coach oder einem Agenten gleichermaßen durchgeführt werden können.

Von Inhabern dieser Rolle können folgende Aufgaben ausgeführt werden:

- Erläuterung der DE-Plattform
- Weitere Vorgehensweise
- Online-Prüfung
- Modalitäten
- Bekanntgabe der Noten
- Zusammenstellung der Prüfung
- Präsentation Lernmaterial
- Präsenzprüfung
- Mitarbeit/Abgaben
- Prüfungsbewertung/Notenvergabe
- Verabschiedung
- Kurs abschließen

1.6. Tutor

Ein Tutor dient zur Unterstützung des Coach (vor allem bei der Bewertung und Korrektur von Aufgaben, bei der Impulsgebung in Diskussionen, etc.).

1.7. Coach/Tutor(en)/Agent

Die Rolle Coach/Tutor(en)/Agent drückt aus, dass die Aufgaben die hier beschrieben sind entweder von einem Coach, den Tutoren oder einem Agenten gleichermaßen durchgeführt werden können.

Von Inhabern dieser Rolle können folgende Aufgaben ausgeführt werden:

- Bewertet Abgaben
- Aufgaben

1.8. Coach/Lernende

Die Rolle Coach/Lernende drückt aus, dass der Coach und die Lernenden die gleichen Aufgaben durchführen.

Von Inhabern dieser Rolle können folgende Aufgaben ausgeführt werden:

- Kommunizieren auf der DE-Plattform

1.9. Kursorganisator

Der Kursorganisator wirkt im Vorfeld. Seine Aufgabe ist die Kursorganisation, Schaffung von Rahmenbedingungen (Plattform, Werkzeuge, Lernmaterial, etc.) Der spätere Coach kann im Vorfeld als Kursorganisator auftreten.

Von Inhabern dieser Rolle können folgende Aufgaben ausgeführt werden:

- Abschluss des Kurses
- Stepwise Improvement
- Positive Entscheidung einen Distance Education Kurs anzubieten

Zu folgenden Teams/Gremien besteht eine Zugehörigkeit:

- Kursorganisator/Agent

1.10. Kursorganisator/Agent

Die Rolle Kursorganisator/Agent drückt aus, dass die Aufgaben die hier beschrieben sind entweder von einem Kurorganisator oder einem Agenten gleichermaßen durchgeführt werden können.

Von Inhabern dieser Rolle können folgende Aufgaben ausgeführt werden:

- Coach bestimmen
- Übergabe an Coach
- Zielgruppenanalyse
- Wahl der Werkzeuge
- Rahmenbedingungen/Ressourcen
- Kursinhalte einfrieren

1.11. Materialgestaltung

Die Rolle der Materialgestaltung umfasst die Auswahl, Entwicklung und Assemblierung von Kurslernmaterial. Diese Rolle kann z.B. auch vom späteren Coach ausgeführt werden.

Von Inhabern dieser Rolle können folgende Aufgaben ausgeführt werden:

- Lernmaterial ergänzen
- Prüfen, ob Lernmaterial vorhanden
- Ergebnisse der Evaluierung im Lernmaterial umsetzen

Zu folgenden Teams/Gremien besteht eine Zugehörigkeit:

- Materialgestaltung/Agent

1.12. Materialgestaltung/Agent

Die Rolle Materialgestaltung/Agent drückt aus, dass die Aufgaben die hier beschrieben sind entweder von einem Materialgestalter oder einem Agenten gleichermaßen durchgeführt werden können.

Von Inhabern dieser Rolle können folgende Aufgaben ausgeführt werden:

- Prüfen, ob bereits Lernmaterial vorhanden
- Online-Lernmaterial zielgruppengerecht aufbereiten
- CD-Rom´s, Handouts zusammen stellen und anfertigen lassen

1.13. Modellgestaltung

Die Modellgestaltung ist für die Ausarbeitung des didaktischen Kursmodells verantwortlich. Diese Rolle kann z.B. auch vom späteren Coach ausgeführt werden.

Von Inhabern dieser Rolle können folgende Aufgaben ausgeführt werden:

- Ergebnisse der Evaluierung im didaktischen Modell umsetzen
- Didaktisches Modell

1.14. Systemadministrator

Der Systemadministrator ist derjenige der die Kursvorarbeiten auf Systemebene leistet. Er ist auch derjenige der die Lernumgebung (Plattform, Datenbanken, etc.) installiert und nach außen hin zugänglich macht.

Von Inhabern dieser Rolle können folgende Aufgaben ausgeführt werden:

- Plattform aufsetzen

Zu folgenden Teams/Gremien besteht eine Zugehörigkeit:

- Systemadministrator/Agent

1.15. Systemadministrator/Agent

Die Rolle Systemadministrator/Agent drückt aus, dass die Aufgaben die hier beschrieben sind entweder von einem Systemadministrator oder einem Agenten gleichermaßen durchgeführt werden können.

Von Inhabern dieser Rolle können folgende Aufgaben ausgeführt werden:

- Zusatzinformationen einbetten
- Arbeitsstruktur anlegen
- Wartung der DE-Plattform
- Lernmaterial einbetten
- Accounts anlegen

1.16. Technischer Dienst

Der Technische Dienst ist Ansprechpartner falls technische Probleme (Hardware-, Software-, Verbindungsschwierigkeiten, etc.) im Kursablauf auftreten.

Von Inhabern dieser Rolle können folgende Aufgaben ausgeführt werden:

- Hilft bei technischen Anfragen und Problemen weiter
- Technische Probleme

1.17. Evaluierungsstelle

Die Evaluierungsstelle ist für die Durchführung der Evaluierung und die Auswertung der Ergebnisse zuständig.

1.18. Evaluierungsstelle/Agent

Die Rolle Evaluierungsstelle/Agent drückt aus, dass die Aufgaben die hier beschrieben sind entweder von einer Evaluierungsstelle oder einem Agenten gleichermaßen durchgeführt werden können.

Von Inhabern dieser Rolle können folgende Aufgaben ausgeführt werden:

- DE-Kurs Evaluierung
- Evaluierung Abschlussveranstaltung
- Evaluierung
- Evaluierung Kurs

2 Informationssysteme

Folgende Klassen sind in diesem Diagramm beschrieben:

- DE-Plattform
- Agenten

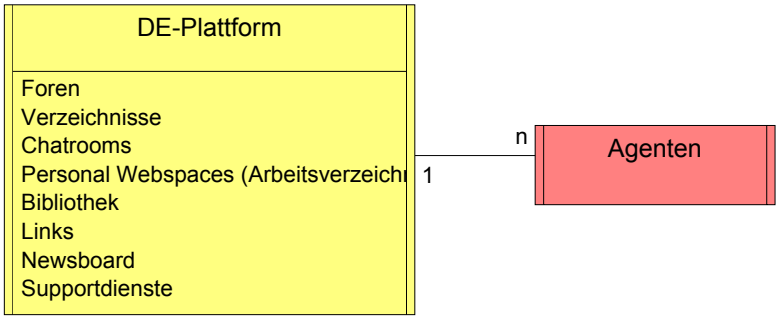


Abbildung 69: Informationssysteme DE-Kurs

2.1. DE-Plattform Elemente

Foren: Foren, dienen als Kommunikationsmittel innerhalb einer Plattform. Dabei kann es Informations-, Diskussions-, Gruppenforen, etc. geben.

Verzeichnisse: Verzeichnisse, dienen zur Strukturierung eines Kurses und lassen sich in Arbeits-, Informations-, Abgabeverzeichnisse, etc. unterscheiden.

Chatrooms: Chatrooms sind Kommunikationsmittel innerhalb eines Forums, dabei kann es Chats geben, die in Echtzeit laufen und bei denen die Beiträge nicht erhalten bleiben (wie ICQ) und es kann quasi-synchrone Chats geben, die über spezielle Foren laufen. Bei diesen bleiben die Beiträge erhalten und können zur Nachlese dienen.

Personal Webspaces: Personal Webspaces sind Bereiche auf die nur jeweils ein Benutzer Zugriff hat und darin seine eigene Struktur des Kurses, Dateien, etc. ablegen und bearbeiten kann. Über „shared folder“ können dabei Teile auch für andere Benutzer zum Zugriff freigegeben werden.

Bibliothek: In der Bibliothek wird üblicherweise das Lernmaterial und Zusatzinformationen zur Verfügung gestellt.

Links: Wichtige Links können in einer Linkrubrik zusammengefasst und zur Verfügung gestellt werden. Über Links wird z.B. auf zusätzliche Informationsquellen, etc. verwiesen.

Newsboard: Wichtige Beiträge und neueste Informationen können auf einem Newsboard übersichtlich zusammen gestellt werden.

Supportdienste: Supportdienste helfen den Benutzern falls Schwierigkeiten im Umgang mit der Plattform, mit dem Zugriff, etc. auftreten.

2.2. Agententypen

Kurssucheagenten: Kurssucheagenten können bereits im Vorfeld Lernenden helfen einen geeigneten Kurs für sie zu finden. Dabei bedienen sich die Agenten Kursmetadaten, die die Kursinhalte beschreiben.

Ressourceermittlungsagent: Dieser Agent kann aufgrund von Checklisten, die Rahmenbedingungen und Ressourcen für einen Kurs ermitteln und prüfen. Auch kann er nach Zugriff auf die Mitarbeiterdatenbank feststellen, welche Mitarbeiter entsprechend ihrer Qualifikation und Verfügbarkeit für welchen Zeitraum und welchen Kurs zur Verfügung stehen.

Zielgruppenanalyseagenten: Zielgruppenanalyseagenten können Benutzerdaten aufnehmen und diese Daten an die entsprechenden Stellen (z.B.: Kursstrukturagenten) weiterleiten. Zusätzlich können sie ein Zielgruppenprofil statistisch aus dem angegebenen Vorwissen, Vorstellungen, Wünschen, etc. der Zielgruppe ableiten.

Werkzeugevaluierungsagenten: Werkzeugevaluierungsagenten können anhand der Datenbank feststellen, welche Werkzeuge mit welchen Funktionen bereits zur Verfügung stehen. Anhand eines Testkatalogs können sie auch neue Werkzeuge prüfen und evaluieren.

Kursassemblierungsagenten: Kursassemblierungsagenten können über Metadaten in Kursrepositories geeignetes Kursmaterial suchen und dieses für die Kursanbieter kopieren. Die Agenten können dabei nach Themengebiet, Zielgruppe (Anfänger, Erfahrene, Experten, Schüler, Studenten, Erwachsene, etc.), Sprache, etc. suchen.

Offlinematerialagenten: Diese veranlassen den Druck ausgewählten Lernmaterials für alle Lernenden und erstellen Images für CD-Rom´s und vervielfältigen diese.

Kursstrukturagenten: Kursstrukturagenten richten für einen Kurs Accounts, Zugriffsrechte, Loginnamen, Passwörter, Verzeichnisse, Foren, etc. ein. Dabei kann der Benutzer aus typischen Settings wählen, die beliebig modifiziert werden können. Der Agent erkennt durch Fragen an den Benutzer die gewünschte Struktur und setzt diese dann automatisch um. Dies ist vor allem bei größeren Teilnehmerzahlen eine wesentliche Arbeitsersparnis.

Lernmaterialeinbettungsagenten: Agenten zur Lernmaterialeinbettung stellen das Lernmaterial in der Bibliothek und setzen Standards, wie CPS-Pakete in der Plattform um.

Notifikationsagenten: Notifikationsagenten überwachen Foren und Verzeichnisse. Falls in Foren ein neuer Beitrag gepostet wird oder in einem Verzeichnis etwas geändert (hinzugefügt, gelöscht, bearbeitet) wird, reagieren die Agenten darauf und verständigen ihre Benutzer. Dies erspart den Benutzern (Lernenden und vor allem den Coaches), dass sie selbst regelmäßig alle Foren und Verzeichnisse überprüfen müssen.

Plattformwartungsagenten: Wartungsagenten können in vorbestimmten zeitlichen Abständen die Plattform durchforsten. Dabei können sie alte Beiträge ins Archiv verschieben, die Struktur und die Inhalte der Plattform sichern, sodass bei einem eventuellen Crash die Plattform nahezu verlustlos wiederhergestellt werden kann.

Assessmentagenten: Assessmentagenten können zur Aufbereitung und zur Überprüfung von Aufgaben, Self-Assessmentbeispielen und Prüfungsfragen eingesetzt werden. Diese Agenten wählen anhand der Zielgruppe, dem Themengebiet, Sprache, etc. Fragen aus dem Kursrepository aus und präsentieren diese dem Coach oder falls gewünscht direkt den Lernenden. Nach Bearbeitung durch die Lernenden kann der Agent die Ergebnisse analysieren, Feedback geben, die Ausarbeitung bewerten und falls ein Notenschema hinterlegt ist eine Benotung vornehmen.

Evaluierungsagenten: Evaluierungsagenten können Fragebögen aus Repositories wählen und die Befragung der Kursteilnehmern und Mitarbeiter (Coach, Tutoren, etc.) durchführen. Diese Fragebögen können anschließend auch statistisch ausgewertet und die Ergebnisse graphisch oder textuell aufbereitet werden.

Kursabbildungsagenten: Soll ein Kurs mit allen Inhalten und strukturgleich auch offline verfügbar sein, so kann ein Agent die Aufbereitung der Struktur außerhalb der Plattform und das Kopieren aller Files übernehmen.

7.3. Agenten-orientierte Wissensplattform (knowledge platform for agents - KNOWPAG)

Hier wird der konkrete Einsatz von Agenten in der Distance Education aufgezeigt. Dafür wurde zunächst eine Wissensplattform (KNOWPAG) geschaffen.

7.3.1. Ziele

Am Institut für Informationsverarbeitung und Mikroprozessortechnik (FIM) der Johannes Kepler Universität Linz wurden zwei Systeme entwickelt. Zum einen das bereits erwähnte WeLearn (Web Environment for Learning), ein Framework für Distance Education und zum anderen POND ([PON]) – ein Agentensystem, welches mobile Agenten unterstützt. Beide Systeme werden derzeit unabhängig voneinander eingesetzt.

Um Kurse aufzufinden, zielgruppengerecht zu präsentieren, etc. und um die Kursorganisation zu erleichtern bedarf es aber der Erweiterung der Materialien um Metadaten.

Diese Metadaten dienen als Mittelschicht zwischen Benutzer und Systemen und sollten, um den gezielten Zugriff durch Agenten zu ermöglichen, einer spezifischen Struktur entsprechen. Die Einbettung der Metadaten (konkret in WeLearn bzw. in ein Kursrepository) soll eine selbständige und stetige Erweiterung der Beispiele, Inhalte, etc. durch die Lehrenden einerseits und eine individuelle Anpassung an die jeweilige Zielgruppe andererseits erlauben. So kann durch die Wahl entsprechender Sichten, Schwierigkeitsgrade, Anwendungsgebiete etc. eine optimale Anpassung an eine Personengruppe, von „Anfängern“ bis zu Lehrenden erfolgen.

Die Plattform von Metadaten (KNOWPAG) wurde unabhängig vom dahinterliegenden Agentensystem erstellt. Sie bietet beliebigen Agenten die Möglichkeit gemäß der Spezifikation Informationen über Kurse abzufragen.

Dabei kann jeder Kursanbieter selbst entscheiden, welches Agentensystem zum Einsatz kommen soll. In dieser Fallstudie wurde POND als Agentensystem gewählt. Dabei wurde zum Schutz der Informationen eine Sicherheitsschicht mittels eines eigenen Portalagenten eingesetzt. Nur ein Agent vom Typ Portalagent darf auf die Daten in der Kursdatenbank (Kursrepository) zugreifen und dient somit als alleiniger Ansprechpartner für alle Agenten, die diese Informationen gewinnen möchten. Dabei können Portalagenten innerhalb des Systems natürlich beliebig repliziert werden.

Ziel war es eine Struktur aufzubauen, die die Gewinnung von Distance Education Kursinformationen für Benutzer und Kursorganisatoren durch Agenten zulässt.

Die wichtigsten Elemente der agenten-orientierten Wissensplattform (KNOWPAG) sind die Schemata der einzelnen Kurse und der Kursinformationen. Diese geben die Struktur und den Typus der Daten vor, die zur Information über Kurse (Metadaten) dienen. Daraus ergibt sich die Struktur der Datenbanken, in der die Kursinformationen tatsächlich gespeichert werden und vor allem auch die Protokolle, die den Agenten den Zugriff auf diese Informationen ermöglichen. Wichtiger Aspekt dabei ist, dass durch die klare Definition es zu keinerlei Ungereimtheiten oder Missverständnissen hinsichtlich der Struktur der Daten, der Datenformate in der Datenbank und dem Zugriff und der Aufbereitung der Daten mittels der Agenten kommen kann.

Für die Strukturierung der Informationen wurde das Konzept der XML-Schemata herangezogen.

7.3.2. XML-Schemata

Das Prinzip der XML-Schemata wurde bereits in Kapitel 4.3.3 XML-Familie beschrieben. Kurz werden hier die wichtigsten Aspekte nochmals wiedergegeben: Durch ein XML-Schema wird die Struktur von Daten definiert, darüber hinaus werden aber auch die Datentypen und Wertebereiche der Daten festgelegt, auch Vererbung ist möglich und vor allem werden Namensräume unterstützt. So sind Elemente gleichen Namens durch unterschiedliche Namensräume eindeutig unterscheidbar.

7.3.2.1. Exkurs: Basiskonzepte

7.3.2.1.1. Exkurs: File-Schemata

Das Konzept der Schemata ist jedoch nicht neu. Es erlebt in XML als XML-Schema eine Realisierung eines bereits in den 1970iger Jahren vorgestellten Konzepts. Der Begriff des Schemas, damals als File-Schema wurde bereits bei McGee [MCG69] und Wedekind [WED73] in den 70iger Jahren als Konzept eines Fileschemata beispielsweise zur Speicherung von Daten in Datenbanken definiert.

Definition des Begriffs Schema bei Wedekind [WED73]: „Ein Schema ist eine Liste, in der jede Liste und jedes Atom

a) einen Namen

b) ein Häufigkeitsattribut (Wiederholungsattribut)

haben. Atome haben darüber hinaus noch ein Wertattribut (value attribute).

Die folgenden Häufigkeitsattribute (repetition attribute) kommen in Betracht:

1.) *Non-repeating* (N): Liste oder Atom kommen genau einmal vor.

2.) *Fixed-repetition* (F): Liste oder Atom wiederholen sich mehr als einmal.

3.) *Optional* (O): Liste oder Atom kommen einmal oder keinmal vor.

4.) *Variable-repetition* (V): Liste oder Atome kommen beliebig häufig vor.“

Eine Liste ist bei [WED73] definiert als: „Eine Liste ist rekursiv definiert als eine endliche Folge von leeren Listen, Atome (Datenelemente) oder Listen. Mit Hilfe der gebräuchlichen Notation (Kommata, Klammern) kann man leicht zwischen Atomen, Listen und der Ordnung in einer Liste unterscheiden.“ Ein Beispiel einer Liste ist in Abbildung 70 dargestellt.

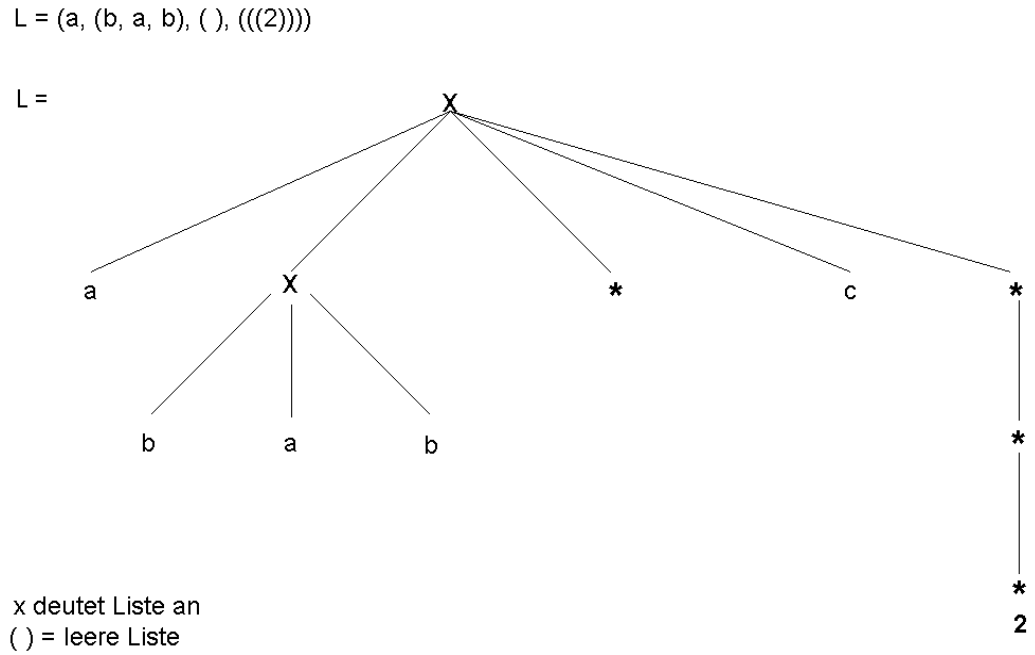


Abbildung 70: Beispiel einer Liste nach [WED73]

Als Beispiel eines Schemas gibt Wedekind die Struktur von Personaldaten an (siehe Abbildung 71).

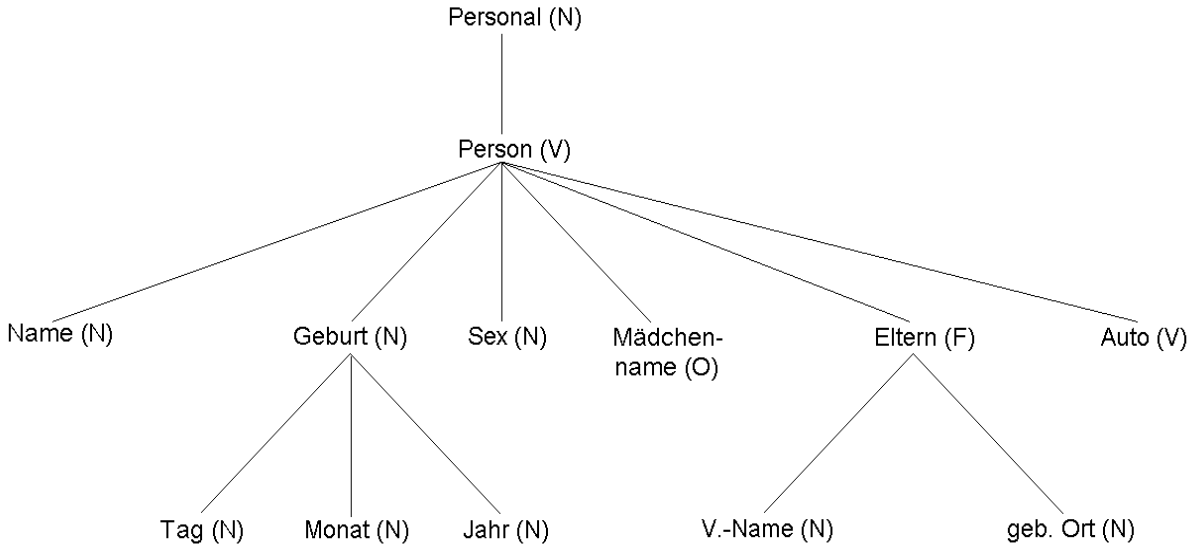


Abbildung 71: Schema für Personaldaten nach [WED73]

Der wesentliche Unterschied zu einer Baumstruktur [MUE73] besteht dabei darin, dass beim Schemabegriff entgegen dem graphentheoretischen Begriff des „Baums“, die Mengen bei der Unterlistenbildung nicht disjunkt zu sein brauchen [WED73], wie beim Baum gefordert. „Ein weiterer Unterschied zwischen Baum und Liste in der Informatik ist, dass Wurzel- und Zwischenknoten keine Wertattribute enthalten. Wertattribute werden bei Listen nur den

Endknoten (Atome) zugeordnet, während bei Bäumen alle Kanten Wertattribute besitzen. In einer erweiterten Liste können jedoch auch in den Zwischenknoten »Informationen« abgespeichert werden. Der Listenbegriff (Schematabegriff) ist eine »echte« Verallgemeinerung des Begriffes »Baum«.“ [WED73]

7.3.2.1.2. Exkurs: VDL – Vienna Definition Language

Die Vienna Definition Language (VDL, [LUC68], [WEG72]) wurde 1968/69 in den Wiener Laboren von IBM entwickelt. Sie stellt eine Metasprache dar, mit der sowohl Syntax als auch Semantik definiert werden kann. Dabei kann die Gültigkeit von Definitionen, Typregeln, etc. verifiziert werden. Die allgemeine Notation von VDL wird dabei als Baumstruktur dargestellt. Eine ihrer ersten Anwendungen fand sie beispielsweise in der formalen Spezifikation von PL/I ([LUC69]).

Die VDL setzt sich zusammen aus unterschiedlichen Objekten (Wiener Objekt [MUE73]), Operationen, etc. Im folgenden werden einige dieser VDL-Elemente exemplarisch beschrieben:

- Strukturierte oder zusammengesetzte Objekte: Diese Objekte bestehen aus mehreren Komponenten. Diese Komponenten haben die Form <s:a>, wobei s ein Selektor ist und a ein Objekt darstellt.

- Elementares Objekt: Ein Elementarobjekt hingegen ist dadurch gekennzeichnet, dass es nicht weiter zerlegt werden kann. Innerhalb einer Baumstruktur wird dies als Knoten dargestellt.

- Nullobjekt: Das Nullobjekt Ω ist das leere Objekt.

- Selektor: Ein Selektor stellt eine Funktion dar, die den Zugriff auf ein Objekt einer Komponente (eines zusammengesetzten Objekts) ermöglicht.

- μ-Operator: Durch den μ-Operator schließlich können unter anderem neue Objekte bebildet werden.

Um VDL zu veranschaulichen sei ein einfaches Beispiel für die Daten einer Person (ähnlich dem Beispiel eines XML-Schemas für Mitgliedsdaten aus Kapitel 4.3.3) gegeben:

Eine Komponente Person besteht aus einer Komponente Namen und einer Komponente Anschrift. Die Komponente Name setzt sich zusammen aus dem Objekt Familienname und dem Objekt Vorname. Die Komponente Anschrift besteht aus dem Objekt Postleitzahl, dem Objekt Stadt, dem Objekt Straße und dem Objekt Nummer. Dies wird in [MUE75] als VDL-Formalismus und als Baum, wie folgt dargestellt:

VDL-Formalismus:

Person = μ_0(<s_1:μ_0(<s_3:Familienname>,<s_4:Vorname>)>,<s_2:μ_0(<s_5:Postleitzahl>,<s_6:Stadt>, s_7:μ_0(<s_8:Straße>,<s_9:Nummer>)>)>).

Baumdarstellung:

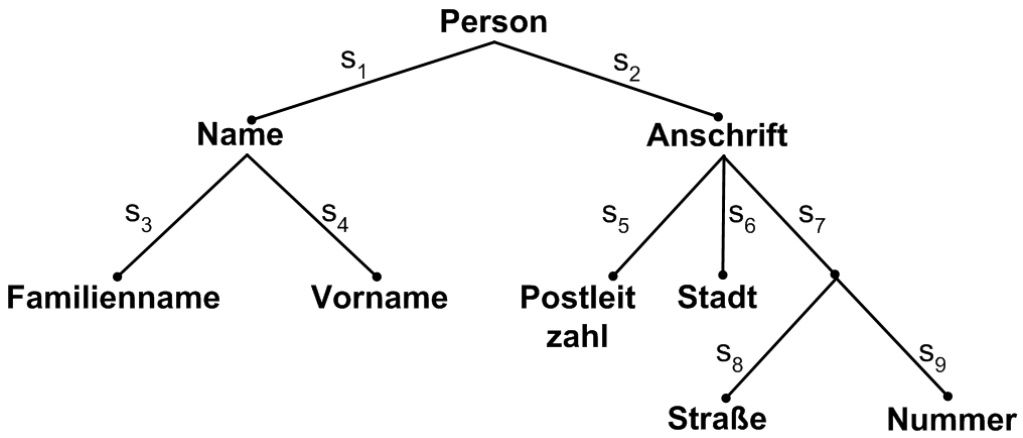


Abbildung 72: Baumdarstellung des VDL-Formalismus für ein Objekt Person (nach [MUE75])

In dieser Art und Weise kann daher durch VDL auch auf abstrakter Ebene die Struktur von Dokumenten beschrieben werden. Man sieht, dass in VDL bereits der Begriff des Objekts und der Objektklassen (durch Prädikate) eingeführt wurde. Auch können die Selektoren von VDL als Methoden auf diese Objekte verstanden werden, wie sie beim heutigen Klassenbegriff Verwendung finden.

7.3.3. Informationen über Distance Education Kurse

Zur Beschreibung von Kursinhalten bzw. Kursmodulen wurde ein XML-Schema erstellt, dass die Struktur und den Typ der Metadaten festlegt. Zur Beschreibung von Kursstrukturen (hierarchischer Aufbau von Kursen) wurde in WeLearn auf einen bestehenden Standard, dem

CPS-Format des Instructional Management System ([IMSa]) des Global Learning Consortium zurückgegriffen.

Auch zur Beschreibung von Metadaten wurden bereits einige Standards von unterschiedlichen Organisationen ausgearbeitet:

IEEE Learning Technology Standards Committee (LTSC): Bei IEEE gibt es mehrere Arbeitsgruppen, die an den unterschiedlichsten Standards im Bereich der Distance Education arbeiten. So wird etwa an Computer Managed Instructions, Platform and Media Profiles, etc. und an Learning Object Metadata gearbeitet. Learning Object Metadata (LOM, [IEE]) wurde von mehreren Organisationen, darunter IMS und ARIADNE (Alliance of Remote Instructional Authoring and Distribution Networks for Europe), entwickelt und unter die Flagge von IEEE gestellt.

IMS Global Learning Consortium: IMS bietet Standards für viele Bereiche an. Diese Bereiche sind unter anderen das bereits erwähnte Content Packaging zum Austausch von Kursen zwischen unterschiedlichen Lernplattformen, aber auch Questions & Test zur Erstellung von Prüfungen und Tests, die beliebigen IMS-kompatiblen Lernplattformen ausgeführt werden können, Learner Profiles Specification zur Verwaltung von individuellen Benutzerdaten und auch Learning-Resources Meta-data zur Speicherung von Informationen über einzelne Objekte einer Lernplattform. Der Standard für Metadaten ([IMSb]) des IMS basiert dabei auf LOM und ist eine Erweiterung der Dublin Core Meta-Data Initiative ([DUB]).

In dieser Arbeit wurde aber nicht auf diese Standards zurückgegriffen. Dies ergibt sich aus der Tatsache, dass diese Strukturbeschreibungen sehr detailliert und komplex sind und daher auch aufwendig zu realisieren sind. Einem Kursanbieter ist es kaum zumutbar, die Beschreibung seiner Kurse entsprechend diesen Standards durchzuführen. Besonders, wenn man die Tatsache bedenkt, dass für einen Kursanbieter zunächst die eigentliche Kurserstellung im Vordergrund steht und die Beschreibung der Inhalte und der zusätzlichen Kursinformationen zunächst zweitrangig erscheinen. Dass dies eigentlich eine zu kurzsichtige Betrachtung darstellt, wird meist erst im nachhinein ersichtlich. Dies verhält sich ungefähr in der Art und Weise, wie Programmierung und Dokumentation miteinander in Verbindung stehen. Viele Programmierer schenken der Dokumentation ihrer Programme wenig Beachtung. Eine gute Dokumentation „verschlingt“ viel Zeit und behindert bei der eigentlichen Programmierung.

Jedoch ist genau diese Dokumentation für die Wartung und Erweiterung „goldes Wert“ und notwendiger Bestandteil eines Programms.

Ebenso verhält es sich mit der Beschreibung von Kursinformationen. Kursinformationen sind nicht nur ein lästiges „Anhängsel“, sondern stellen sogar ein wichtiges Marketinginstrument dar. Erst durch eine entsprechende Beschreibung kann ein Kurs auch angepriesen und von Agenten aufgefunden werden.

7.3.3.1. Beschreibung von Bildungsangeboten

Hier wurde zunächst eine Struktur zur Beschreibung von Kursmetadaten erstellt. Diese ist durch ein XML-Schema aufgebaut. Dadurch, dass es als XML-Schema realisiert wurde, können auch problemlos bestehende Standards eingebunden werden, falls dieses gewünscht ist.

7.3.3.1.1. XML-Schema: Agentenprotokoll – WeLearnCourseMetaData

Das XML-Schema „WeLearnCourseMetaData“ spezifiziert die Struktur von Metadaten für Distance Education Kurse.

Das XML-Schema WeLearnCourseMetaData wurde so gestaltet, dass Anwender so wenig Information als nötig bekannt geben müssen, wenn es ihre Zeit oder andere Bedingungen eine detailliertere Kursbeschreibung nicht zulassen. Darüber hinaus können Kursinhalte, der Kursaufbau und -ablauf maßgeschneidert für bestimmte Zielgruppen strukturiert und beschrieben werden. Auch Kurspreise, bzw. Preise für einzelne Module können, falls gewünscht in unterschiedlichen Währungen hinterlegt werden. Im folgenden soll das XML-Schema WeLearnCourseMetaData exakt beschrieben werden.

XML-Schema: WeLearnCourseMetaData.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.1 U (http://www.xmlspy.com) by Reisinger Susanne (FIM) -->
<!-- XML Schema for WeLearn Course Metadata created by Susanne Reisinger -->
<xs:schema targetNamespace="http://www.fim.uni-linz.ac.at" xmlns="http://www.fim.uni-linz.ac.at"
xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified" attributeFormDefault="unqualified"
version="WeLearn CMD 1.0">
  <!-- ** Root Element WeLearnCourseMetaData ** -->
  <xs:element name="WeLearnCourseMetaData">
      <xs:annotation>
          <xs:documentation>Root element for the meta data structure of a course</xs:documentation>
      </xs:annotation>
```

```
<!-- ** ComplexType Course ** -->
    <xs:complexType>
        <xs:sequence>
            <xs:element name="Course" type="CourseType" minOccurs="0" maxOccurs="unbounded">
                <xs:annotation>
                    <xs:documentation>Root element for a course structure </xs:documentation>
                </xs:annotation>
            </xs:element>
            <xs:element name="Module" type="ModuleType" minOccurs="0" maxOccurs="unbounded">
                <xs:annotation>
                    <xs:documentation>Module(s) of a course</xs:documentation>
                </xs:annotation>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
</xs:element>

<!-- ** ComplexType CourseType ** -->
<xs:complexType name="CourseType">
    <xs:sequence>
        <xs:element name="GeneralInfo" type="GenInfoType">
            <xs:annotation>
                <xs:documentation>Structure for general course information</xs:documentation>
            </xs:annotation>
        </xs:element>
        <xs:element name="SpecialInfo" type="SpecInfoType" minOccurs="0">
            <xs:annotation>
                <xs:documentation>Information about timestamps and physical location</xs:documentation>
            </xs:annotation>
        </xs:element>
        <xs:element name="IncludedModule" type="ModuleID" minOccurs="0" maxOccurs="unbounded">
            <xs:annotation>
                <xs:documentation>List of the ModuleID´s the course consists of</xs:documentation>
            </xs:annotation>
        </xs:element>
    </xs:sequence>
    <xs:attribute name="cid" type="CourseID" use="required"/>
    <xs:attribute name="Status" type="StatusType" use="required"/>
    <xs:attribute name="lang" type="xs:string" default="DE"/>
</xs:complexType>
<!-- ** SimpleType CourseID ** -->
<xs:simpleType name="CourseID">
    <xs:restriction base="xs:ID">
        <xs:pattern value="C\d{6}"/>
    </xs:restriction>
</xs:simpleType>

<!-- ** ComplexType ModuleType ** -->
<xs:complexType name="ModuleType">
    <xs:sequence>
        <xs:element name="GeneralInfo" type="GenInfoType"/>
        <xs:element name="SpecialModuleInfo" type="SpecMInfoType" minOccurs="0">
            <xs:annotation>
                <xs:documentation>Special module information (author, copyright)</xs:documentation>
            </xs:annotation>
        </xs:element>
        <xs:element name="SpecialInfo" type="SpecInfoType" minOccurs="0">
            <xs:annotation>
                <xs:documentation>Information about location and timestamps of the module</xs:documentation>
            </xs:annotation>
        </xs:element>
        <xs:element name="StructureInfo" type="StructInfoType">
            <xs:annotation>
                <xs:documentation>Information concerning the content and the format </xs:documentation>
            </xs:annotation>
        </xs:element>
        <xs:element name="OrganisationInfo" type="OrgInfoType" minOccurs="0" maxOccurs="unbounded">
            <xs:annotation>
```

```
                <xs:documentation>Information concerning a possible organisation of a course. what is the previous
                                and next module of this module depending on the audience.</xs:documentation>
            </xs:annotation>
        </xs:element>
    </xs:sequence>
    <xs:attribute name="mid" type="ModuleID" use="required"/>
    <xs:attribute name="Status" type="StatusType" use="required"/>
    <xs:attribute name="lang" type="xs:string" default="DE"/>
</xs:complexType>

<!-- ** SimpleType ModueID ** -->
<xs:simpleType name="ModuleID">
    <xs:restriction base="xs:string">
        <xs:pattern value="M\d{6}"/>
    </xs:restriction>
</xs:simpleType>
<!-- ** ComplexType GenInfoType ** -->
<xs:complexType name="GenInfoType">
    <xs:sequence>
        <xs:element name="Title" type="xs:string">
            <xs:annotation>
                <xs:documentation>Title of a course/module</xs:documentation>
            </xs:annotation>
        </xs:element>
        <xs:element name="Description" type="xs:string" minOccurs="0">
            <xs:annotation>
                <xs:documentation>Description of a course/module</xs:documentation>
            </xs:annotation>
        </xs:element>
        <xs:element name="Discipline" type="xs:string" maxOccurs="unbounded">
            <xs:annotation>
                <xs:documentation>The disciplines the course/module is for (e.g. informatics, law,
                                psychology, music, etc.)</xs:documentation>
            </xs:annotation>
        </xs:element>
        <xs:element name="Audience" type="xs:string" minOccurs="0" maxOccurs="unbounded">
            <xs:annotation>
                <xs:documentation>Audience or target groups (e.g. beginners, aged …,</xs:documentation>
            </xs:annotation>
        </xs:element>
        <xs:element name="Keyword" type="xs:string" maxOccurs="unbounded">
            <xs:annotation>
                <xs:documentation>Keywords concerning the content of the course/module</xs:documentation>
            </xs:annotation>
        </xs:element>
        <xs:element name="Price" type="PriceType" minOccurs="0">
            <xs:annotation>
                <xs:documentation>The price of the course/module for participants</xs:documentation>
            </xs:annotation>
        </xs:element>
    </xs:sequence>
</xs:complexType>

<!-- ** ComplexType SpecInfoType ** -->
<xs:complexType name="SpecInfoType">
    <xs:sequence>
        <xs:element name="TimeStamp" type="TimeStampType" minOccurs="0">
            <xs:annotation>
                <xs:documentation>Collection of important timestamps</xs:documentation>
            </xs:annotation>
        </xs:element>
        <xs:element name="Location" type="xs:string" minOccurs="0">
            <xs:annotation>
                <xs:documentation>Physical location of the course/module (URI, folder, etc.)</xs:documentation>
            </xs:annotation>
        </xs:element>
    </xs:sequence>
</xs:complexType>
```

```
<!-- ** ComplexType OrgInfoType ** -->
<xs:complexType name="OrgInfoType">
    <xs:sequence>
        <xs:element name="PrevModuleID" type="ModuleID" minOccurs="0">
            <xs:annotation>
                <xs:documentation>ModuleID of the previous module</xs:documentation>
            </xs:annotation>
        </xs:element>
        <xs:element name="NextModuleID" type="ModuleID" minOccurs="0">
            <xs:annotation>
                <xs:documentation>ModuleID of the next module</xs:documentation>
            </xs:annotation>
        </xs:element>
    </xs:sequence>
    <xs:attribute name="Audience" type="xs:string" use="required"/>
</xs:complexType>

<!-- ** ComplexType SpecMInfoType ** -->
<xs:complexType name="SpecMInfoType">
    <xs:sequence>
        <xs:element name="Author" type="xs:string" minOccurs="0" maxOccurs="unbounded">
            <xs:annotation>
                <xs:documentation>Name of the author of the module</xs:documentation>
            </xs:annotation>
        </xs:element>
        <xs:element name="Copyright" type="xs:string" minOccurs="0" maxOccurs="unbounded">
            <xs:annotation>
                <xs:documentation>Copyright of the module</xs:documentation>
            </xs:annotation>
        </xs:element>
    </xs:sequence>
</xs:complexType>

<!-- ** ComplexType StructInfoType ** -->
<xs:complexType name="StructInfoType">
    <xs:sequence>
        <xs:element name="FormatInfo" type="FormatInfoType">
            <xs:annotation>
                <xs:documentation>Format type of the module. Possible values: Text, HTML, Presentation, Video,
                                  Audio, Picture, Database, Tool, SourceCode</xs:documentation>
            </xs:annotation>
        </xs:element>
        <xs:element name="ContentTypeInfo" type="ContentTypeInfoType">
        </xs:element>
    </xs:sequence>
</xs:complexType>

<!-- ** SimpleType FormatInfoType ** -->
<xs:simpleType name="FormatInfoType">
    <xs:restriction base="xs:string">
        <xs:enumeration value="Text"/>
        <xs:enumeration value="HTML"/>
        <xs:enumeration value="Presentation"/>
        <xs:enumeration value="Video"/>
        <xs:enumeration value="Audio"/>
        <xs:enumeration value="Picture"/>
        <xs:enumeration value="Tool"/>
        <xs:enumeration value="Database"/>
        <xs:enumeration value="SourceCode"/>
    </xs:restriction>
</xs:simpleType>

<!-- ** SimpleType ContentTypeInfoType ** -->
<xs:simpleType name="ContentTypeInfoType">
    <xs:restriction base="xs:string">
        <xs:enumeration value="Content"/>
        <xs:enumeration value="Exercise"/>
        <xs:enumeration value="Example"/>
        <xs:enumeration value="Assessment"/>
```

```
            <xs:enumeration value="Exam"/>
            <xs:enumeration value="Solution"/>
            <xs:enumeration value="Exercise/Solution"/>
            <xs:enumeration value="Exam/Solution"/>
            <xs:enumeration value="Bibliography"/>
            <xs:enumeration value="Glossary"/>
            <xs:enumeration value="GeneralInfo"/>
            <xs:enumeration value="AdditionalInfo"/>
            <xs:enumeration value="Library"/>
            <xs:enumeration value="Dictionary"/>
        </xs:restriction>
    </xs:simpleType>

    <!-- ** ComplexType TimeStampType ** -->
    <xs:complexType name="TimeStampType">
        <xs:sequence>
            <xs:element name="CreationDate" type="xs:date" minOccurs="0">
                <xs:annotation>
                    <xs:documentation>Date of the course/module creation</xs:documentation>
                </xs:annotation>
            </xs:element>
            <xs:element name="LastModifiedDate" type="xs:date" minOccurs="0">
                <xs:annotation>
                    <xs:documentation>Date of the last modification of the course/module</xs:documentation>
                </xs:annotation>
            </xs:element>
            <xs:element name="ReleaseDate" type="xs:date" minOccurs="0">
                <xs:annotation>
                    <xs:documentation>Date of the course/module release</xs:documentation>
                </xs:annotation>
            </xs:element>
            <xs:element name="ExpirationDate" type="xs:date" minOccurs="0">
                <xs:annotation>
                    <xs:documentation>Date when the course/module expires</xs:documentation>
                </xs:annotation>
            </xs:element>
        </xs:sequence>
    </xs:complexType>

    <!-- ** ComplexType PriceType ** -->
    <xs:complexType name="PriceType">
        <xs:sequence>
            <xs:element name="Amount" type="xs:nonNegativeInteger">
                <xs:annotation>
                    <xs:documentation>The amount of the price</xs:documentation>
                </xs:annotation>
            </xs:element>
            <xs:element name="Currency" type="xs:string" default="EUR">
                <xs:annotation>
                    <xs:documentation>The currency of the price (default: EUR)</xs:documentation>
                </xs:annotation>
            </xs:element>
        </xs:sequence>
    </xs:complexType>

    <!-- ** SimpleType StatusType ** -->
    <xs:simpleType name="StatusType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="Released"/>
            <xs:enumeration value="In Progress"/>
            <xs:enumeration value="Archived"/>
        </xs:restriction>
    </xs:simpleType>
</xs:schema>
```

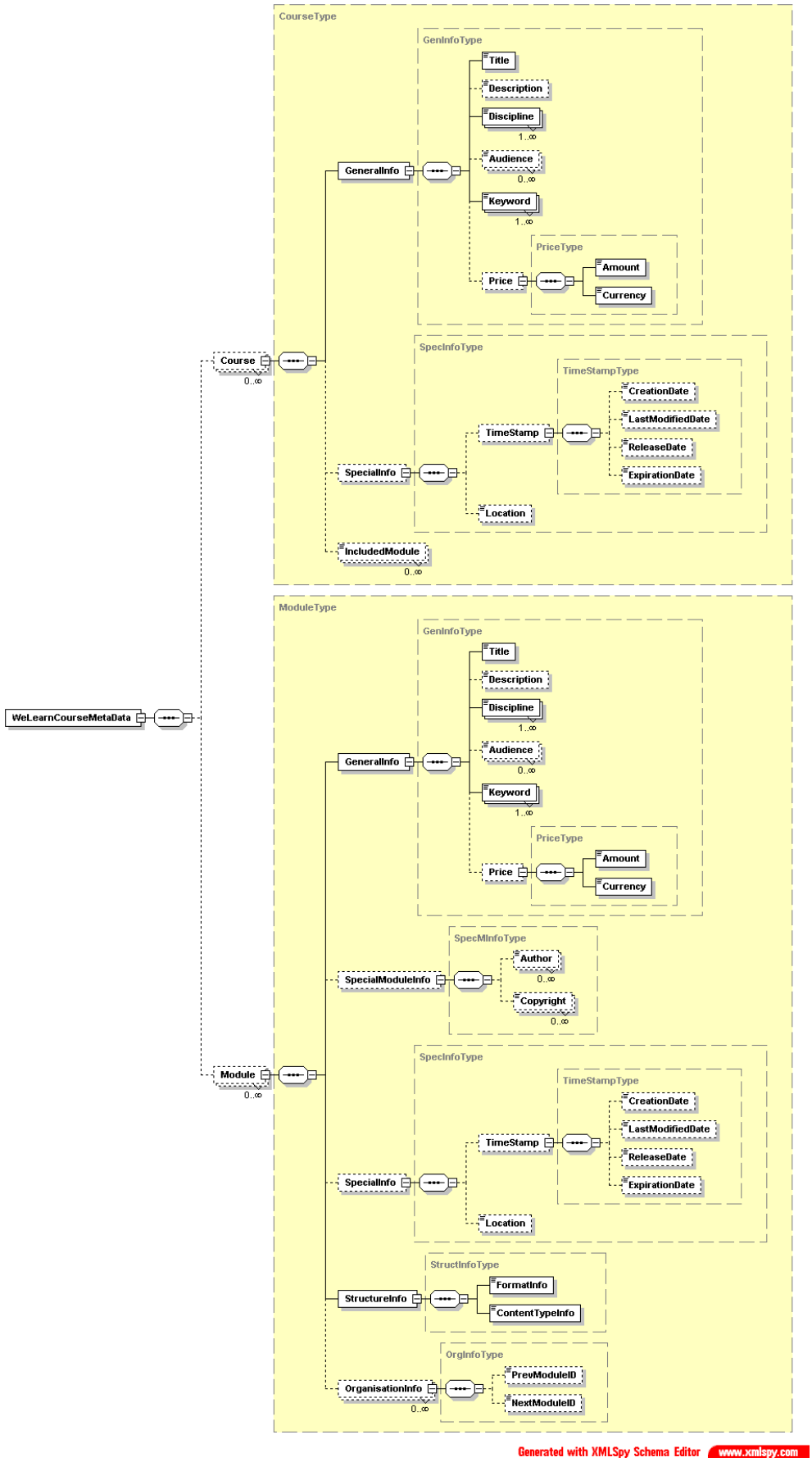
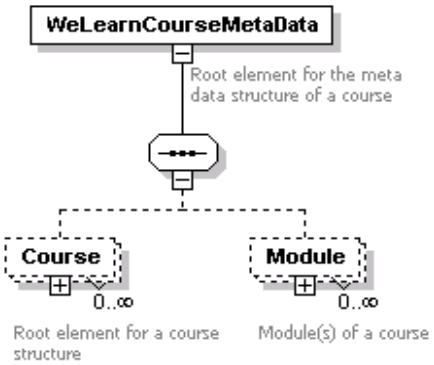
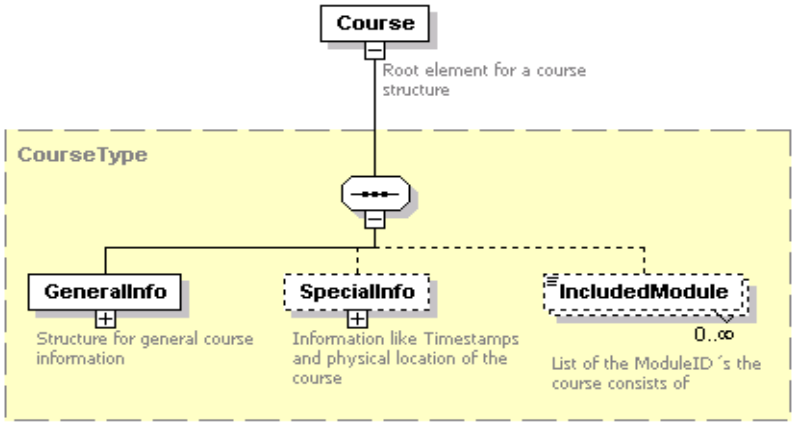
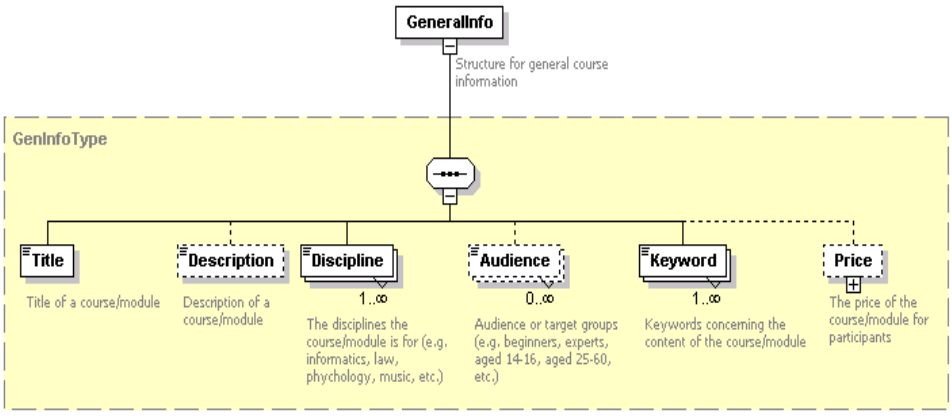


Abbildung 73: Graphische Darstellung der WeLearnCourseMetaData-Struktur

Element	**WeLearnCouseMetaData**
Beschreibung	WeLearnCouseMetaData ist so aufgebaut, dass entweder ein oder mehrere Kurse (Course) oder ein oder mehrere Module (Module) oder aber beides Kurse und Module in einer XML-Datei beschrieben werden können. Dadurch können Kurse und Kursmodule (falls gewünscht) getrennt voneinander beschrieben werden. Dies ist vor allem dann von Vorteil, wenn verschiedene Module Bestandteil von unterschiedlichen Kursen sind. So kann etwa das Modul „Studyguide“ Bestandteil von vielen DE-Kursen sein.
Diagramm	
Source	<xs:element name="WeLearnCourseMetaData"> <xs:annotation> <xs:documentation>Root element for the meta data structure of a course</xs:documentation> </xs:annotation> <xs:complexType> <xs:sequence> <xs:element name="Course" type="CourseType" minOccurs="0" maxOccurs="unbounded"> <xs:annotation> <xs:documentation>Root element for a course structure </xs:documentation> </xs:annotation> </xs:element> <xs:element name="Module" type="ModuleType" minOccurs="0" maxOccurs="unbounded"> <xs:annotation> <xs:documentation>Module(s) of a course</xs:documentation> </xs:annotation> </xs:element> </xs:sequence> </xs:complexType> </xs:element>
Element	**Course**
Beschreibung	Ein Kurs (Course) kann danach durch folgende Details beschrieben werden: Ein Kurs ist immer durch eine eindeutige ID (cid) gekennzeichnet. Diese ID ist immer durch einen führenden Buchstaben „C“ und 6 Dezimalzahlen aufgebaut. Er hat einen Status (Status), der die Werte „In Progress“, „Released“, „Archived“ annehmen kann, wobei „Released“ als Defaultwert

	hinterlegt ist. Ebenso muss die Kurssprache (lang) angegeben werden. Defaultwert hierfür ist „DE“. Zusätzlich dazu werden jedenfalls generelle Informationen (GeneralInfo) angegeben und optional dazu auch speziellere Informationen (SpecialInfo) und Informationen, Modul-Id eines oder mehrerer Module (IncludedModule) aus denen sich der Kurs zusammensetzt.
Diagramm	
Source	<xs:element name="Course" type="CourseType" minOccurs="0" maxOccurs="unbounded"> </xs:element> <xs:complexType name="CourseType"> <xs:sequence> <xs:element name="GeneralInfo" type="GenInfoType"> <xs:annotation> <xs:documentation>Structure for general course information</xs:documentation> </xs:annotation> </xs:element> <xs:element name="SpecialInfo" type="SpecInfoType" minOccurs="0"> <xs:annotation> <xs:documentation>Information like Timestamps and physical location of the course</xs:documentation> </xs:annotation> </xs:element> <xs:element name="IncludedModule" type="ModuleID" minOccurs="0" maxOccurs="unbounded"> <xs:annotation> <xs:documentation>List of the ModuleID´s the course consists of</xs:documentation> </xs:annotation> </xs:element> </xs:sequence> <xs:attribute name="cid" type="CourseID" use="required"/> <xs:attribute name="Status" type="StatusType" use="required"/> <xs:attribute name="lang" type="xs:string" default="DE"/> </xs:complexType> <xs:simpleType name="CourseID"> <xs:restriction base="xs:ID"> <xs:pattern value="C\d{6}"/> </xs:restriction> </xs:simpleType> <xs:simpleType name="StatusType"> <xs:restriction base="xs:string"> <xs:enumeration value="Released"/> <xs:enumeration value="In Progress"/> <xs:enumeration value="Archived"/> </xs:restriction> </xs:simpleType>

Element	**GeneralInfo (für Kurse)**
Beschreibung	Generelle Informationen (GeneralInfo), die angegeben werden müssen, sind ein Titel (Title) des Kurses, zumindest ein Gebiet (Discipline) in der der Kurs eingesetzt werden kann (etwa Informatik, Rechtswissenschaften, etc.) und zumindest ein Schlagwort (Keyword) über den Inhalt des Kurses. Darüber hinaus können noch detaillierte Information, wie eine Beschreibung (Description), ein oder mehrere Zielgruppen (Audience), wie etwa Anfänger, Experten, Personen im Alter zwischen 6 und 10 Jahren, etc. und die Kursgebühr (Price) hinterlegt werden.
Diagramm	
Source	<xs:element name="GeneralInfo" type="GenInfoType"> <xs:annotation> <xs:documentation>Structure for general course information</xs:documentation> </xs:annotation> </xs:element> <xs:complexType name="GenInfoType"> <xs:sequence> <xs:element name="Title" type="xs:string"> </xs:element> <xs:element name="Description" type="xs:string" minOccurs="0"> <xs:annotation> <xs:documentation>Description of a course/module</xs:documentation> </xs:annotation> </xs:element> <xs:element name="Discipline" type="xs:string" maxOccurs="unbounded"> </xs:element> <xs:element name="Audience" type="xs:string" minOccurs="0" maxOccurs="unbounded"> </xs:element> <xs:element name="Keyword" type="xs:string" maxOccurs="unbounded"> <xs:annotation> <xs:documentation>Keywords concerning the content of the course/module</xs:documentation> </xs:annotation> </xs:element> <xs:element name="Price" type="PriceType" minOccurs="0"> <xs:annotation> <xs:documentation>The price of the course/module for participants</xs:documentation> </xs:annotation> </xs:element> </xs:sequence> </xs:complexType>

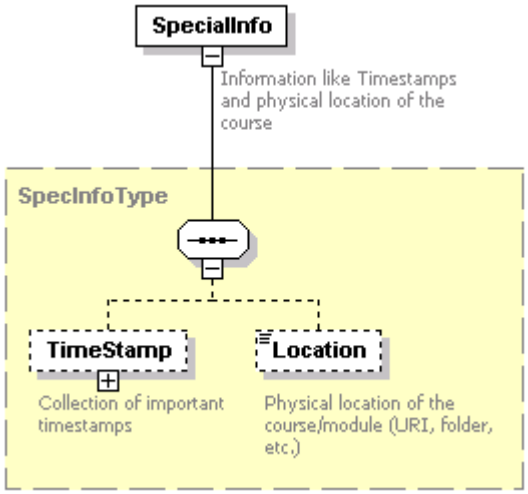
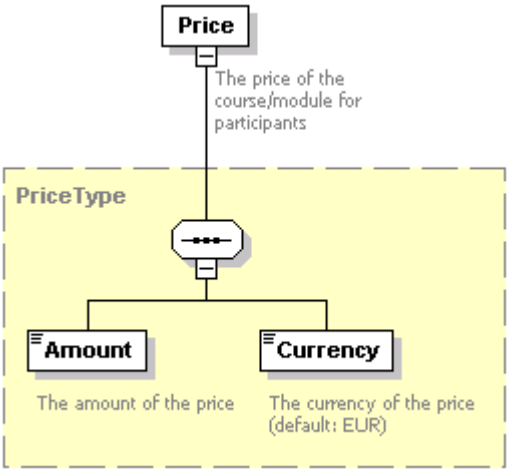
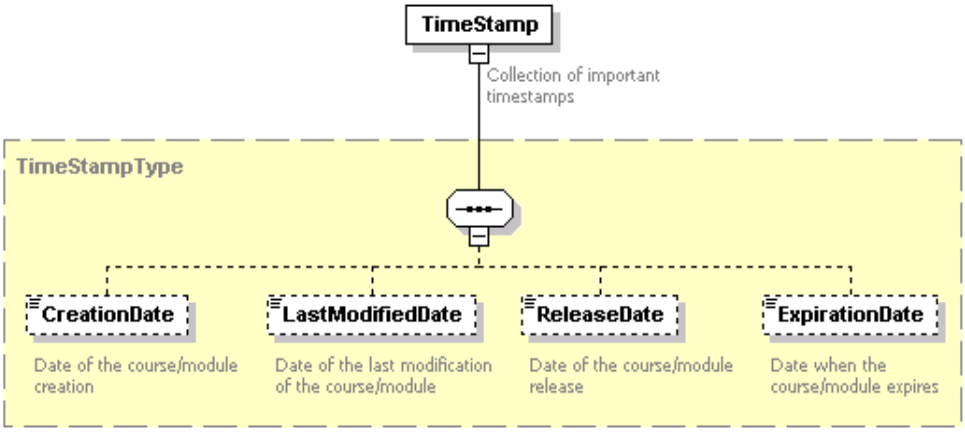
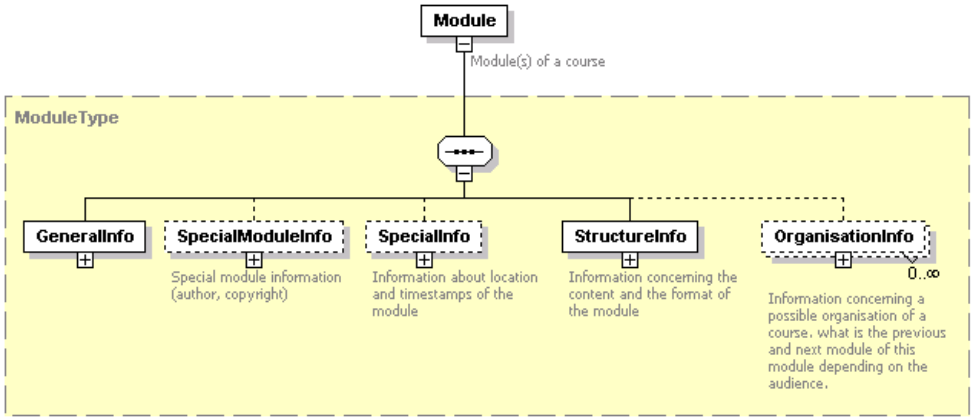
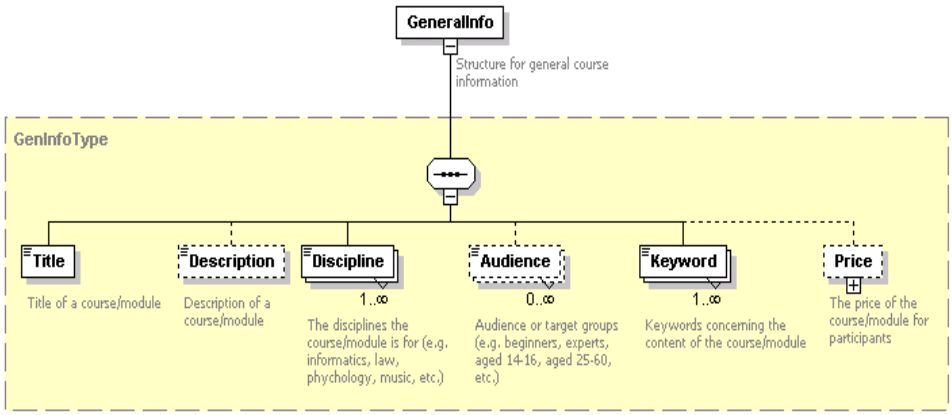
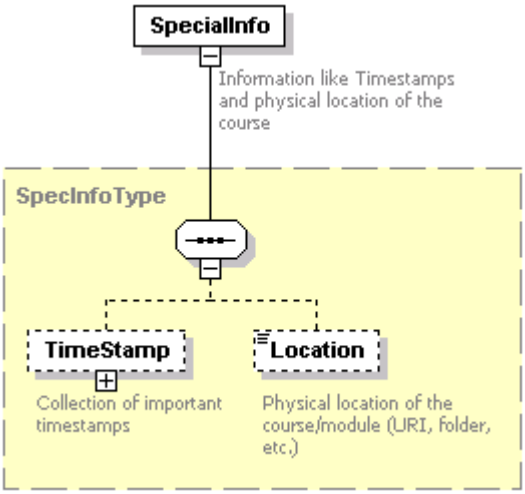
Element	**SpecialInfo (für Kurse)**
Beschreibung	Speziellere Informationen können, müssen aber nicht angeführt werden. Falls doch können hier noch Information, wie Zeitstempel (Timestamp) und/oder der physikalische Speicherort (Location) des Kurses, sprich URI, Verzeichnis auf der Festplatte, etc. spezifiziert werden.
Diagramm	
Source	<xs:element name="SpecialInfo" type="SpecInfoType" minOccurs="0"> <xs:annotation> <xs:documentation>Information like Timestamps and physical location of the course</xs:documentation> </xs:annotation> </xs:element> <xs:complexType name="SpecInfoType"> <xs:sequence> <xs:element name="TimeStamp" type="TimeStampType" minOccurs="0"> <xs:annotation> <xs:documentation>Collection of important timestamps</xs:documentation> </xs:annotation> </xs:element> <xs:element name="Location" type="xs:string" minOccurs="0"> <xs:annotation> <xs:documentation>Physical location of the course/module (URI, folder, etc.)</xs:documentation> </xs:annotation> </xs:element> </xs:sequence> </xs:complexType>
Element	**Price (für Kurse)**
Beschreibung	Wird eine Kursgebühr angegeben, so kann sowohl die Höhe (Amount) als auch die Währung (Currency) der Kursgebühr angegeben werden. Als Defaultwert der Währung ist „EUR“ angegeben.

Diagramm	
Source	<xs:element name="Price" type="PriceType" minOccurs="0"> <xs:annotation> <xs:documentation>The price of the course/module for participants</xs:documentation> </xs:annotation> </xs:element> <xs:complexType name="PriceType"> <xs:sequence> <xs:element name="Amount" type="xs:nonNegativeInteger"> <xs:annotation> <xs:documentation>The amount of the price</xs:documentation> </xs:annotation> </xs:element> <xs:element name="Currency" type="xs:string" default="EUR"> <xs:annotation> <xs:documentation>The currency of the price (default: EUR)</xs:documentation> </xs:annotation> </xs:element> </xs:sequence> </xs:complexType>
Element	**TimeStamp (für Kurse)**
Beschreibung	Als Zeitstempel können falls gewünscht ein Entwicklungsdatum (CreationDate), das Datum der letzten Änderung (LastModifiedDate), das Freigabedatum (ReleaseDate) und ein Ablaufdatum (ExpirationDate) angegeben werden.
Diagramm	

Source	<xs:element name="TimeStamp" type="TimeStampType" minOccurs="0"> <xs:annotation> <xs:documentation>Collection of important timestamps</xs:documentation> </xs:annotation> </xs:element> <xs:complexType name="TimeStampType"> <xs:sequence> <xs:element name="CreationDate" type="xs:date" minOccurs="0"> <xs:annotation> <xs:documentation>Date of the course/module creation</xs:documentation> </xs:annotation> </xs:element> <xs:element name="LastModifiedDate" type="xs:date" minOccurs="0"> <xs:annotation> <xs:documentation>Date of the last modification of the course/module</xs:documentation> </xs:annotation> </xs:element> <xs:element name="ReleaseDate" type="xs:date" minOccurs="0"> <xs:annotation> <xs:documentation>Date of the course/module release</xs:documentation> </xs:annotation> </xs:element> <xs:element name="ExpirationDate" type="xs:date" minOccurs="0"> <xs:annotation> <xs:documentation>Date when the course/module expires</xs:documentation> </xs:annotation> </xs:element> </xs:sequence> </xs:complexType>
Element	**Module**
Beschreibung	Ein Modul (Module) ist immer durch eine eindeutige ID (mid) gekennzeichnet. Diese ID ist immer durch einen führenden Buchstaben „M“ und 6 Dezimalzahlen aufgebaut. Es hat einen Status (Status), der die Werte „In Progress“, „Released“, „Archived“ annehmen kann, wobei „Released“ als Defaultwert hinterlegt ist. Ebenso muss die Modulsprache (lang) angegeben werden. Defaultwert hierfür ist „DE“. Zusätzlich dazu werden jedenfalls generelle (GeneralInfo) und strukturelle (StructureInfo) Informationen angegeben. Optional dazu können auch speziellere Informationen (SpecialInfo) und spezielle Modulinformationen (SpecialModuleInfo), sowie organisatorische Informationen (OrganisationInfo) beschrieben werden.

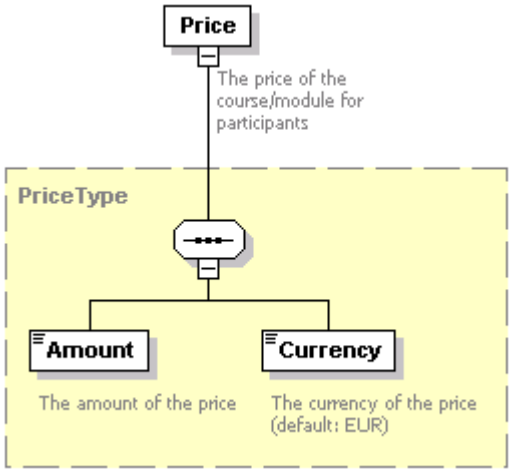
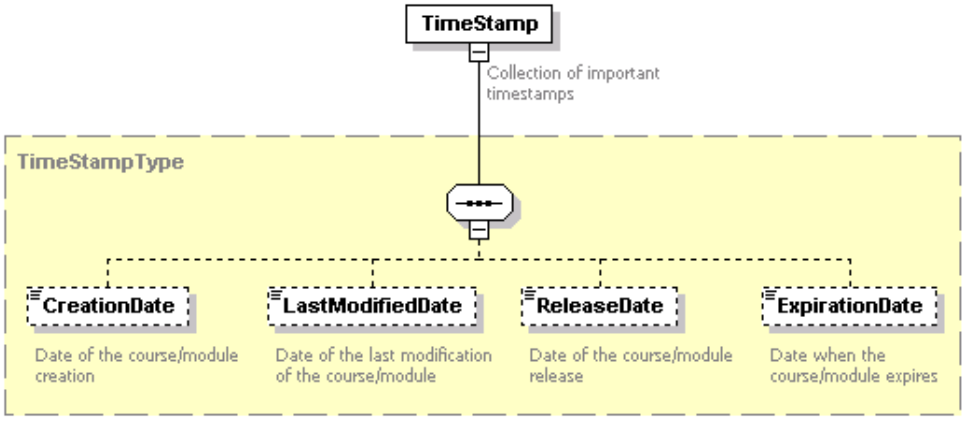
Diagramm	
Source	<xs:element name="Module" type="ModuleType" minOccurs="0" maxOccurs="unbounded"> <xs:annotation> <xs:documentation>Module(s) of a course</xs:documentation> </xs:annotation> </xs:element> <xs:complexType name="ModuleType"> <xs:sequence> <xs:element name="GeneralInfo" type="GenInfoType"/> <xs:element name="SpecialModuleInfo" type="SpecMInfoType" minOccurs="0"> <xs:annotation> <xs:documentation>Special module information (author, copyright)</xs:documentation> </xs:annotation> </xs:element> <xs:element name="SpecialInfo" type="SpecInfoType" minOccurs="0"> <xs:annotation> <xs:documentation>Information about location and timestamps of the module</xs:documentation> </xs:annotation> </xs:element> <xs:element name="StructureInfo" type="StructInfoType"> <xs:annotation> <xs:documentation>Information concerning the content and the format of the module</xs:documentation> </xs:annotation> </xs:element> <xs:element name="OrganisationInfo" type="OrgInfoType" minOccurs="0" maxOccurs="unbounded"> <xs:annotation> <xs:documentation>Information concerning a possible organisation of a course. what is the previous and next module of this module depending on the audience.</xs:documentation> </xs:annotation> </xs:element> </xs:sequence> <xs:attribute name="mid" type="ModuleID" use="required"/> <xs:attribute name="Status" type="StatusType" use="required"/> <xs:attribute name="lang" type="xs:string" default="DE"/> </xs:complexType> <xs:simpleType name="ModuleID"> <xs:restriction base="xs:string"> <xs:pattern value="M\d{6}"/> </xs:restriction> </xs:simpleType> <xs:simpleType name="StatusType"> <xs:restriction base="xs:string"> <xs:enumeration value="Released"/> <xs:enumeration value="In Progress"/> <xs:enumeration value="Archived"/> </xs:restriction> </xs:simpleType>
Element	**GeneralInfo (für Module)**
Beschreibung	Generelle Informationen (GeneralInfo), die angegeben werden müssen, sind ein Titel (Title) des Moduls, zumindest ein Gebiet (Discipline) in der das

	Modul eingesetzt werden kann (etwa Informatik, Rechtswissenschaften, etc.) und zumindest ein Schlagwort (Keyword) über den Inhalt des Moduls. Darüber hinaus können noch detaillierte Information, wie eine Beschreibung (Description), ein oder mehrere Zielgruppen (Audience), wie etwa Anfänger, Experten, Personen im Alter zwischen 6 und 10 Jahren, etc. und die Modulgebühr (Price) hinterlegt werden.
Diagramm	
Source	<xs:element name="GeneralInfo" type="GenInfoType"> <xs:annotation> <xs:documentation>Structure for general course information</xs:documentation> </xs:annotation> </xs:element> <xs:complexType name="GenInfoType"> <xs:sequence> <xs:element name="Title" type="xs:string"> <xs:annotation> <xs:documentation>Title of a course/module</xs:documentation> </xs:annotation> </xs:element> <xs:element name="Description" type="xs:string" minOccurs="0"> <xs:annotation> <xs:documentation>Description of a course/module</xs:documentation> </xs:annotation> </xs:element> <xs:element name="Discipline" type="xs:string" maxOccurs="unbounded"> <xs:annotation> <xs:documentation>The disciplines the course/module is for (e.g. informatics, law, music, etc.)</xs:documentation> </xs:annotation> </xs:element> <xs:element name="Audience" type="xs:string" minOccurs="0" maxOccurs="unbounded"> <xs:annotation> <xs:documentation>Audience or target groups (e.g. beginners, experts, aged 14-16, etc.)</xs:documentation> </xs:annotation> </xs:element> <xs:element name="Keyword" type="xs:string" maxOccurs="unbounded"> <xs:annotation> <xs:documentation>Keywords concerning the content of the course/module</xs:documentation> </xs:annotation> </xs:element> <xs:element name="Price" type="PriceType" minOccurs="0"> <xs:annotation> <xs:documentation>The price of the course/module for participants</xs:documentation> </xs:annotation> </xs:element> </xs:sequence> </xs:complexType>

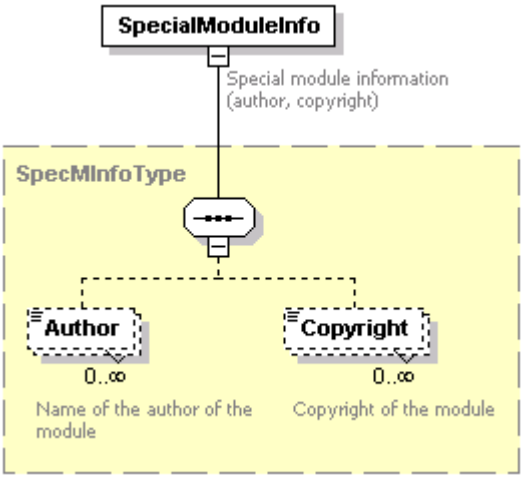
Element	**SpecialInfo (für Module)**
Beschreibung	Speziellere Informationen können, müssen aber nicht angeführt werden. Falls doch können hier noch Information, wie Zeitstempel (Timestamp) und/oder der physikalische Speicherort (Location) des Moduls, sprich URI, Verzeichnis auf der Festplatte, etc. angeführt werden.
Diagramm	

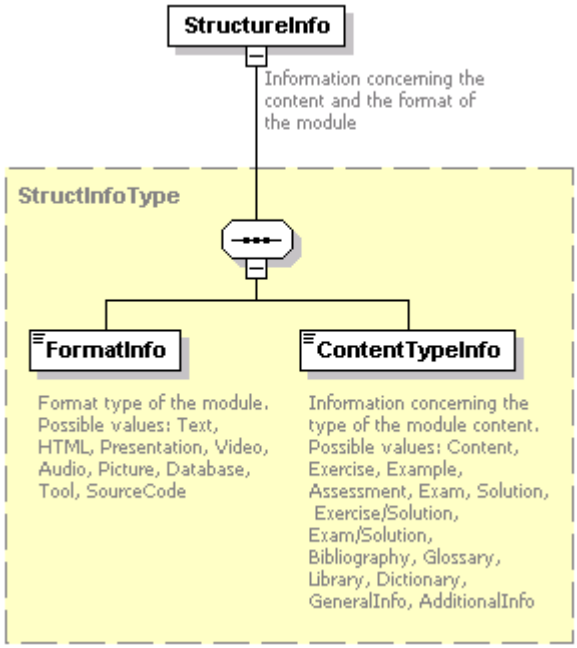
Source	`<xs:element name="SpecialInfo" type="SpecInfoType" minOccurs="0">`
	`  <xs:annotation>`
	`  <xs:documentation>Information like Timestamps and physical location of the course</xs:documentation>`
	`  </xs:annotation>`
	`</xs:element>`
	`<xs:complexType name="SpecInfoType">`
	`  <xs:sequence>`
	`  <xs:element name="TimeStamp" type="TimeStampType" minOccurs="0">`
	`  <xs:annotation>`
	`  <xs:documentation>Collection of important timestamps</xs:documentation>`
	`  </xs:annotation>`
	`  </xs:element>`
	`  <xs:element name="Location" type="xs:string" minOccurs="0">`
	`  <xs:annotation>`
	`  <xs:documentation>Physical location of the course/module (URI, folder, etc.)</xs:documentation>`
	`  </xs:annotation>`
	`  </xs:element>`
	`  </xs:sequence>`
	`</xs:complexType>`

Element	**Price (für Module)**
Beschreibung	Wird eine Modulgebühr angegeben, so kann sowohl die Höhe (Amount) als auch die Währung (Currency) der Modulgebühr angegeben werden. Als Defaultwert der Währung ist „EUR“ angegeben.

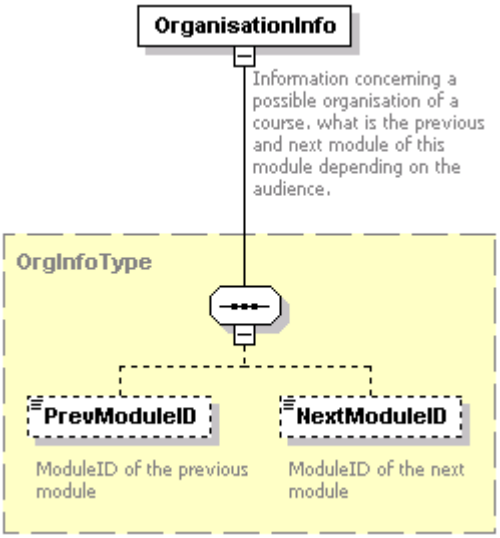
Diagramm	
Source	<xs:element name="Price" type="PriceType" minOccurs="0"> </xs:element> <xs:complexType name="PriceType"> <xs:sequence> <xs:element name="Amount" type="xs:nonNegativeInteger"> <xs:annotation> <xs:documentation>The amount of the price</xs:documentation> </xs:annotation> </xs:element> <xs:element name="Currency" type="xs:string" default="EUR"> <xs:annotation> <xs:documentation>The currency of the price (default: EUR)</xs:documentation> </xs:annotation> </xs:element> </xs:sequence> </xs:complexType>
Element	**TimeStamp (für Module)**
Beschreibung	Als Zeitstempel können, falls gewünscht ein Entwicklungsdatum (CreationDate), das Datum der letzten Änderung (LastModifiedDate), das Freigabedatum (ReleaseDate) und ein Ablaufdatum (ExpirationDate) angegeben werden.
Diagramm	
Source	<xs:element name="TimeStamp" type="TimeStampType" minOccurs="0"> <xs:annotation> <xs:documentation>Collection of important timestamps</xs:documentation> </xs:annotation> </xs:element> <xs:complexType name="TimeStampType">

```
  <xs:sequence>
    <xs:element name="CreationDate" type="xs:date" minOccurs="0">
      <xs:annotation>
        <xs:documentation>Date of the course/module creation</xs:documentation>
      </xs:annotation>
    </xs:element>
    <xs:element name="LastModifiedDate" type="xs:date" minOccurs="0">
      <xs:annotation>
        <xs:documentation>Date of the last modification of the course/module</xs:documentation>
      </xs:annotation>
    </xs:element>
    <xs:element name="ReleaseDate" type="xs:date" minOccurs="0">
      <xs:annotation>
        <xs:documentation>Date of the course/module release</xs:documentation>
      </xs:annotation>
    </xs:element>
    <xs:element name="ExpirationDate" type="xs:date" minOccurs="0">
      <xs:annotation>
        <xs:documentation>Date when the course/module expires</xs:documentation>
      </xs:annotation>
    </xs:element>
  </xs:sequence>
</xs:complexType>
```

Element	**SpecialModuleInfo**
Beschreibung	Als spezielle Modulinformationen können ein Autor (Author), sprich Ersteller des Moduls und das Copyright (Copyright) angegeben werden.
Diagramm	
Source	<xs:element name="SpecialModuleInfo" type="SpecMInfoType" minOccurs="0"> <xs:annotation> <xs:documentation>Special module information (author, copyright)</xs:documentation> </xs:annotation> </xs:element> <xs:complexType name="SpecMInfoType"> <xs:sequence> <xs:element name="Author" type="xs:string" minOccurs="0" maxOccurs="unbounded"> <xs:annotation> <xs:documentation>Name of the author of the module</xs:documentation> </xs:annotation> </xs:element> <xs:element name="Copyright" type="xs:string" minOccurs="0" maxOccurs="unbounded"> <xs:annotation> <xs:documentation>Copyright of the module</xs:documentation> </xs:annotation> </xs:element> </xs:sequence></xs:complexType>

Element	**StructureInfo (für Module)**
Beschreibung	Als Strukturinformationen können einerseits das Format des Moduls (FormatInfo) und andererseits eine Beschreibung des Modulinhalts (ContentTypeInfo) angegeben werden. Als Format kann dabei Text, HTML, Presentation, Video, Audio, Picture, Tool, Database oder SourceCode angegeben werden. Als Inhaltsbeschreibung kann Content, Exercise, Example, Assessment, Exam, Solution, Exercise/Solution, Exam/Solution, Bibliography, Glossary, GeneralInfo, AdditionalInfo, Library oder Dictionary gewählt werden.
Diagramm	
Source	<xs:element name="StructureInfo" type="StructInfoType"> </xs:element> <xs:complexType name="StructInfoType"> <xs:sequence> <xs:element name="FormatInfo" type="FormatInfoType"> </xs:element> <xs:element name="ContentTypeInfo" type="ContentTypeInfoType"> </xs:element> </xs:sequence> </xs:complexType> <xs:simpleType name="FormatInfoType"> <xs:restriction base="xs:string"> <xs:enumeration value="Text"/> <xs:enumeration value="HTML"/> <xs:enumeration value="Presentation"/> <xs:enumeration value="Video"/> <xs:enumeration value="Audio"/> <xs:enumeration value="Picture"/> <xs:enumeration value="Tool"/> <xs:enumeration value="Database"/> <xs:enumeration value="SourceCode"/> </xs:restriction> </xs:simpleType> <xs:simpleType name="ContentTypeInfoType"> <xs:restriction base="xs:string">

	`<xs:enumeration value="Content"/>`
	`<xs:enumeration value="Exercise"/>`
	`<xs:enumeration value="Example"/>`
	`<xs:enumeration value="Assessment"/>`
	`<xs:enumeration value="Exam"/>`
	`<xs:enumeration value="Solution"/>`
	`<xs:enumeration value="Exercise/Solution"/>`
	`<xs:enumeration value="Exam/Solution"/>`
	`<xs:enumeration value="Bibliography"/>`
	`<xs:enumeration value="Glossary"/>`
	`<xs:enumeration value="GeneralInfo"/>`
	`<xs:enumeration value="AdditionalInfo"/>`
	`<xs:enumeration value="Library"/>`
	`<xs:enumeration value="Dictionary"/>`
	`</xs:restriction>`
	`</xs:simpleType>`

Element	**OrganisationInfo (für Module)**
Beschreibung	Hier können zielgruppengerichtete Daten hinterlegt werden. Für eine bestimmte Zielgruppe (Audience) wird hier angegeben, welches Modul, sprich Modulid logisch als Vorgänger (PrevModuleID) bzw. als Nachfolger (NextModuleID) betrachtet wird.
Diagramm	

Source	`<xs:element name="OrganisationInfo" type="OrgInfoType" minOccurs="0" maxOccurs="unbounded">`
	`<xs:annotation>`
	`<xs:documentation>Information concerning a possible organisation of a course. what is the previous and next module of this module depending on the audience.</xs:documentation>`
	`</xs:annotation>`
	`</xs:element>`
	`<xs:complexType name="OrgInfoType">`
	`<xs:sequence>`
	`<xs:element name="PrevModuleID" type="ModuleID" minOccurs="0">`
	`<xs:annotation>`
	`<xs:documentation>ModuleID of the previous module</xs:documentation>`
	`</xs:annotation>`
	`</xs:element>`
	`<xs:element name="NextModuleID" type="ModuleID" minOccurs="0">`
	`<xs:annotation>`
	`<xs:documentation>ModuleID of the next module</xs:documentation>`
	`</xs:annotation>`
	`</xs:element>`
	`</xs:sequence>`
	`<xs:attribute name="Audience" type="xs:string" use="required"/>`
	`</xs:complexType>`

Zusammenfassend bietet die Beschreibung von Kursen und Kursmodulen nach diesem Schema folgende Möglichkeiten:

Kursrepository: Mit den Metadaten der Kurse kann ein Repository aufgebaut werden. Das hat den Vorteil für Kursanbieter, dass sie bei der Zusammenstellung ihrer Kurse auf umfangreiches bereits vorhandenes Lernmaterial zugreifen können, dieses direkt in ihre Kurse einbauen, oder entsprechend adaptieren bzw. erweitern können. Kurse können somit nach den Bausteinsystem zusammen gestellt werden. Daraus ergibt sich eine höhere Kosteneffizienz durch Wiederverwendung und einer niedrigeren Redundanz.

Kurssuche: Lernende können aus der Fülle der Kurse, Kurse bzw. Kursteile (über Agenten) abrufen, die für sie von Interesse sind und an ihre Vorkenntnisse anknüpfen. Sie bekommen dadurch eine bessere Übersicht über die vorhandenen Kurse und welche Kurse speziell für sie Informationen bieten.

Zielgruppengerechte Gestaltung: Durch die Metadaten müssen Kurse nicht für jede Zielgruppe neu geschrieben werden. Die Beschreibung der Metadaten erlaubt es zielgruppengerechte Kurse zusammenzustellen. So werden etwa bei Anfängerkursen ausschließlich Kursmodule für Anfänger verwendet und diese durch entsprechende begleitende Informationen, wie Studyguides, Nachschlagewerke, etc. erweitert. Je nach Bedarf können so auch Kurse spezifisch für den schulischen, universitären oder Erwachsenenweiterbildungsbereich erstellt werden. Die Gestaltung der Kurse kann dabei nach Zielgruppe, Themengebiet, Schlagwörtern und vor allem auch nach der bevorzugten Sprache (Deutsch, Englisch, etc.) erfolgen.

Inhaltsorientierte Gestaltung: Durch die Metadaten können Kurse nicht nur nach Zielgruppen, sondern auch nach Themengebieten punktgenau erstellt werden. So kann etwa ein Kurs zum Thema Algorithmen aus dem Kursrepository erstellt werden, der Inhaltsmodule, Beispielmodule, Hands-on Experiencemodule und Assessmentmodule beinhaltet. Je nach Bedarf kann ein Kursanbieter zur Evaluierung auf Prüfungsmodule zugreifen und ein Modul für die Prüfung auswählen.

7.3.4. Agenten

Intelligente mobile Agenten wurden auf der Basis des Agentensystems POND ([PON]) erstellt, welches am Institut für Informationsverarbeitung und Mikroprozessortechnik (FIM) entwickelt und umgesetzt wurde.

Zur besseren Erläuterung der Funktionsweise der Agenten werden hier zwei Agenten aufgezeigt, welche für WeLearn implementiert wurden. Anhand des ersten (einfacheren) Agenten sollen der genaue Kommunikationsablauf und der Nachrichtenaustausch im Agentensystem beschrieben und verdeutlicht werden. Auch die nötigen Entwicklungsschritte und der Aufbau der Strukturen werden anhand dieses Agenten erläutert.

Am Beispiel des ersten Agenten wird nicht nur der Bereich dargestellt, den der Benutzer zu sehen bekommt (Frontoffice), sondern es wird auch auf den Bereich (Backoffice) eingegangen der normalerweise ausschließlich innerhalb des Systems zu sehen und für den Endbenutzer verborgen ist.

Der zweite Agent ist ein Agent, der die Struktur des XML-Schemas WeLearnCourseMetaData umsetzt und durch den es möglich ist Kurse zu suchen, kursrelevante Daten abzufragen bzw. Kurse zielgruppengerecht zu erstellen.

Genauer wird dabei auf die Funktionalität der Kurssuche eingegangen. Hier wurde ein Prototyp für ein Web-Interface für WeLearn erstellt, welches den neuen Frontofficebereich für die Benutzer bildet. Der Backoffice-Bereich ist für alle in dieser Arbeit beschriebenen mobilen intelligenten Agenten gleich und wird daher nur bei Agent 1 erläutert.

7.3.4.1. Agent 1: WEMemberDataAgent

Dieser Agent ist ein Beispiel für die direkte Benutzerunterstützung.

Ablauf der Kommunikation

Der Benutzer sieht und arbeitet mit einem Agenten vom Typ WEMemberDataAgent (Benutzeragent). Dieser Typ von Agent hat 2 Funktionen. Zum einen kann jedes Mitglied einer Lernplattform (z.B. WeLearn), seine eingetragenen Daten überprüfen. Dafür startet der Benutzer einen WEMemberDataAgent, der in einem POND-Agentensystem läuft.

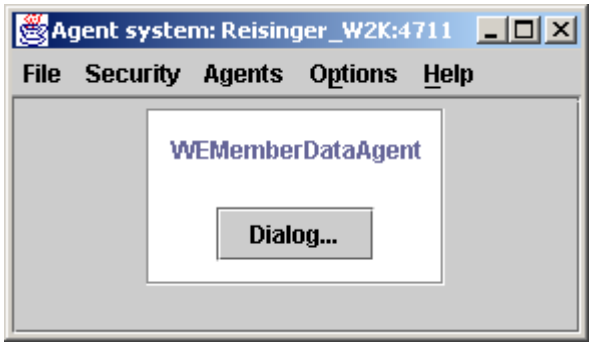


Abbildung 74: WEMemberDataAgent im POND Agentensystem

Danach wählt der Benutzer die Option „Mitgliedsdaten anzeigen“ aus.

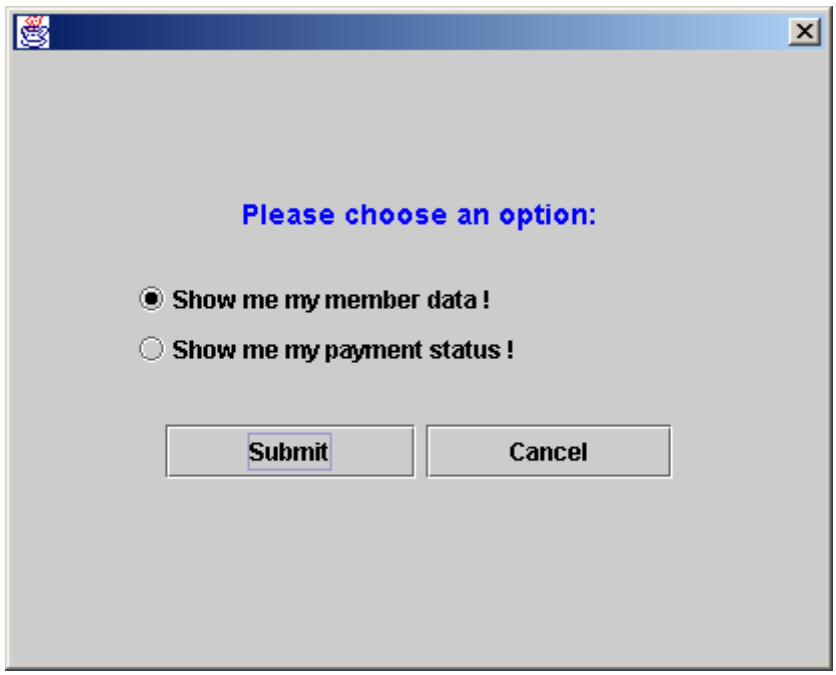


Abbildung 75: Auswahl der Funktionen des Agenten WEMemberDataAgent

Anschließend erscheint ein Dialog, indem der Benutzer seine Mitgliedsidentität (MemberID) eingibt und diese bestätigt.

Abbildung 76: Dialog für ID-Eingabe

Abgeschirmt von den Benutzern läuft in einem zweiten Agentensystem ein anderer Agent vom Typ PortalServerAgent (WeLearn-Framework Agent), der auf die Datenbank des Portals bzw. der Lernplattform zugreifen darf. Aus Sicherheitsgründen hat nur dieser Typ von Agenten die Berechtigung auf die Daten der Datenbank zuzugreifen. Andere Agenten müssen diese Agenten ansprechen und sie nach den gewünschten Daten fragen.

Das heißt, dass zuerst ein Agent vom Typ WEMemberDataAgent und ein Agent vom Typ PortalServerAgent getrennt voneinander in zwei unterschiedlichen Agentensystemen laufen.

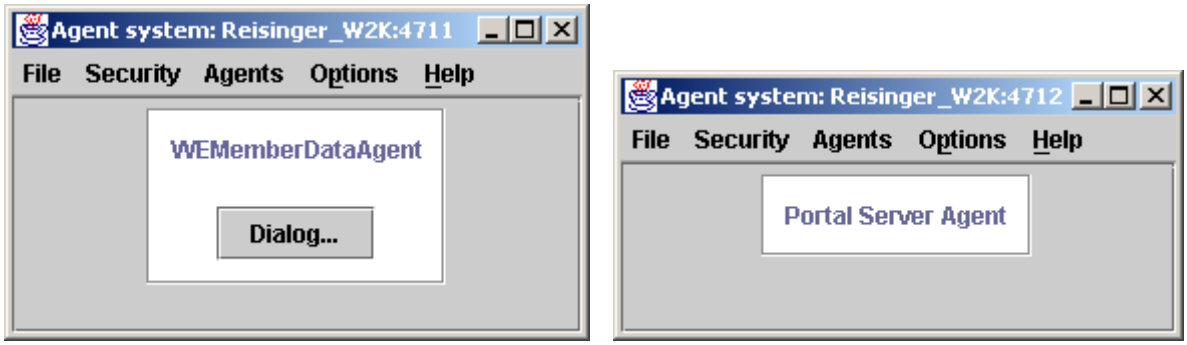


Abbildung 77: Zwei getrennte Agentensysteme

Um die gewünschten Daten abfragen zu können, migriert der WEMemberDataAgent zum Agentensystem, in dem der PortalServerAgent läuft.

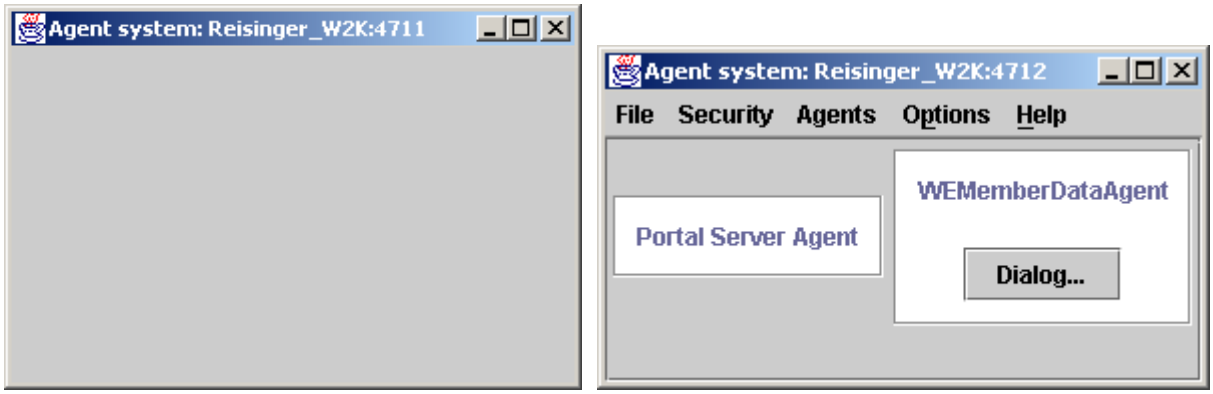


Abbildung 78: WEMemberDataAgent ist zum anderen Agentensystem migriert

Dort übergibt er dem PortalServerAgent seine Daten. Der PortalServerAgent kopiert die gewünschten Daten, für die der Benutzer zum Abfragen berechtigt ist aus der Datenbank und übergibt diese dem WEMemberDataAgent. Anschließend migriert der WEMemberDataAgent wieder zu seinem Ausgangsagentensystem zurück.

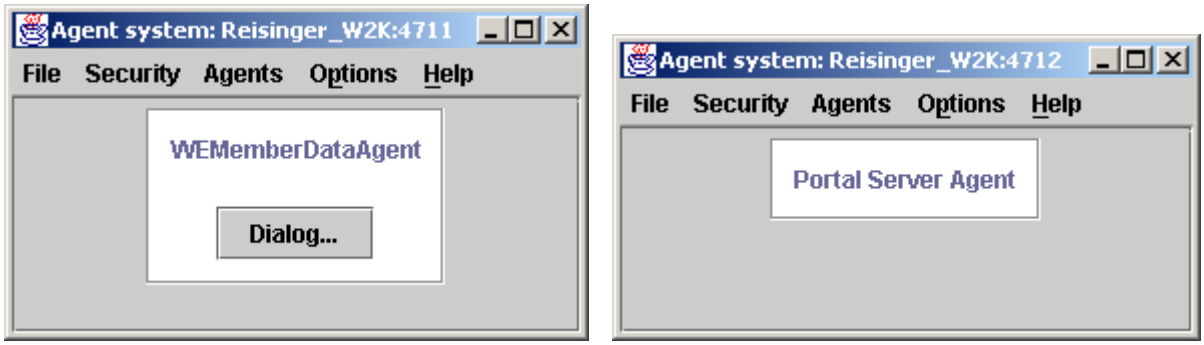


Abbildung 79: Agent ist zum Ausgangsagentensystem zurück migriert

Der Agent kann nun die gewünschten Daten dem Benutzer präsentieren.

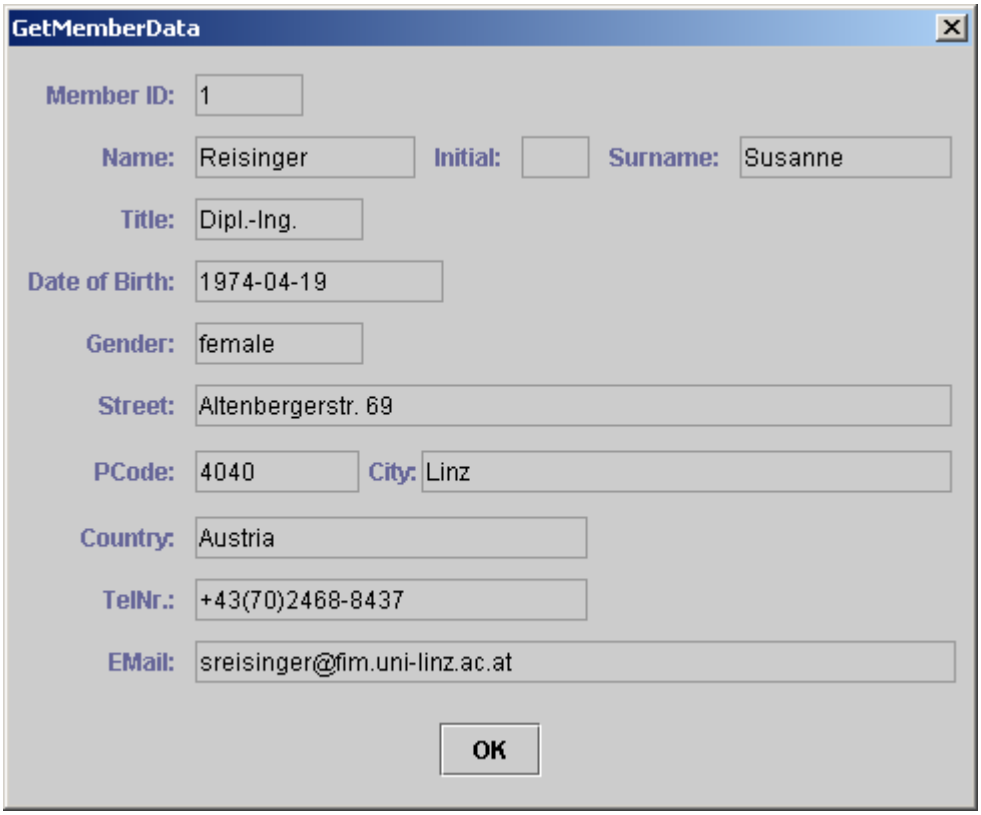


Abbildung 80: Dialog mit den gewünschten Benutzerdaten

Wird die zweite Funktion ausgewählt, so kann der Benutzer prüfen, welche Kurs- und Kursveranstaltungsgebühren noch nicht bezahlt wurden. Weiters kann er dem Agenten auch Zahlungsdaten bekannt geben und der Agent übernimmt die Bezahlung der offenen Gebühren.

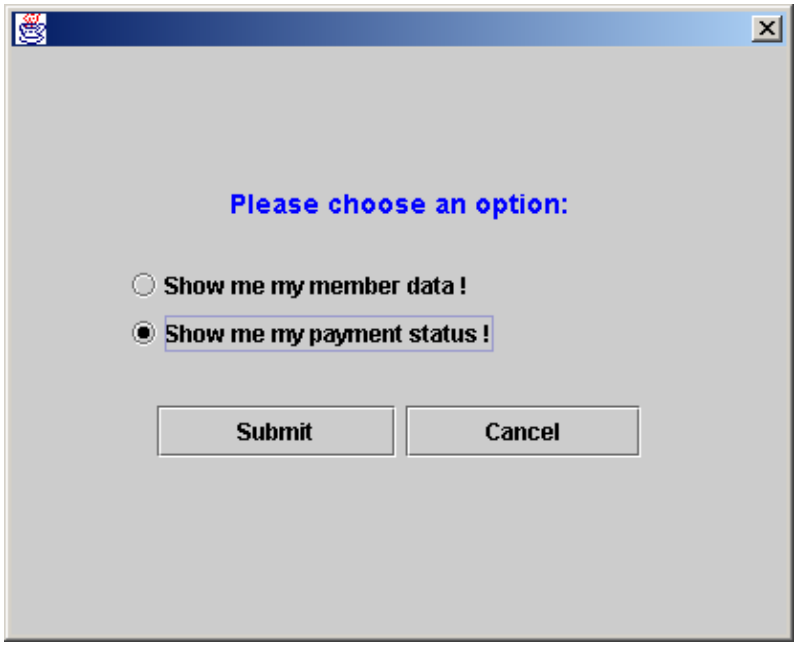


Abbildung 81: Zweite Funktion des Agenten

Der Benutzer gibt seine Mitgliedsidentität (MemberID) und Kursidentität (CourseID) ein.

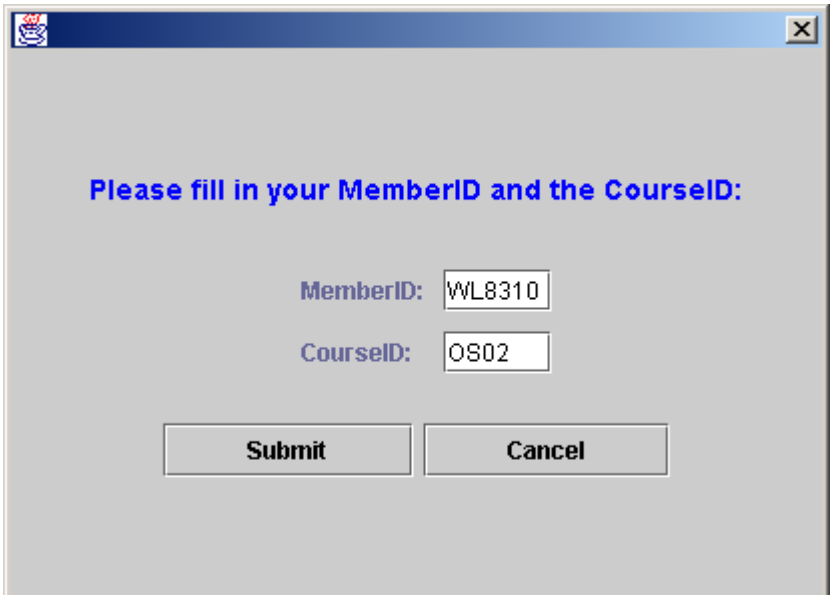


Abbildung 82: Eingabe der ID´s

Der Ablauf ist hier gleich. Der WEMemberDataAgent migriert zum Agentensystem auf dem ein PortalServerAgent läuft. Diesem übergibt er die Daten und teilt ihm mit, welche Daten er benötigt. Der PortalServerAgent sucht und kopiert je nach Berechtigung des Benutzers die

entsprechenden Daten aus der Datenbank und übergibt sie dem WEMemberDataAgent. Der WEMemberDataAgent migriert zurück und zeigt dem Benutzer offene Gebühren an.

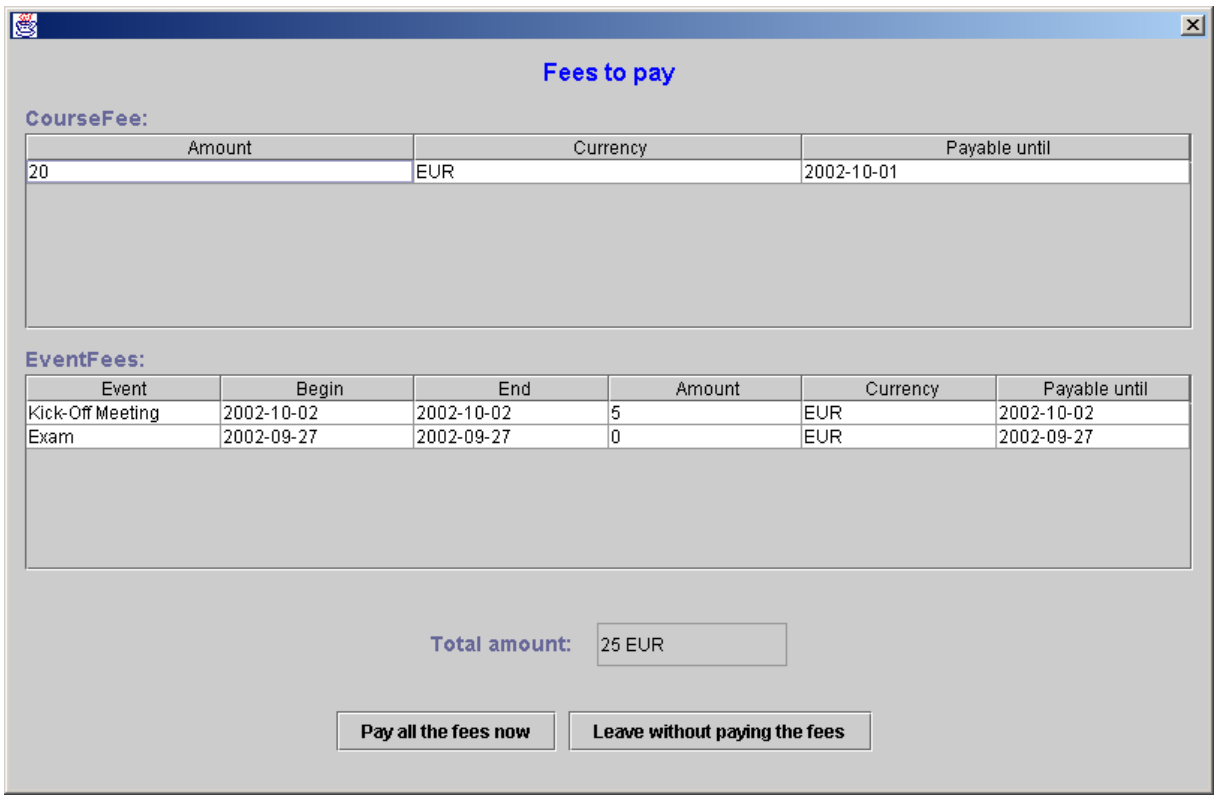


Abbildung 83: Dialog mit den noch offenen Gebühren eines Benutzers

Der Benutzer kann nun den Agenten stoppen oder ihn beauftragen, die Gebührenbezahlung für ihn durchzuführen.

Soll der Agent die Gebührenbezahlung durchführen, kann der Benutzer seine Kreditkartendaten eingeben.

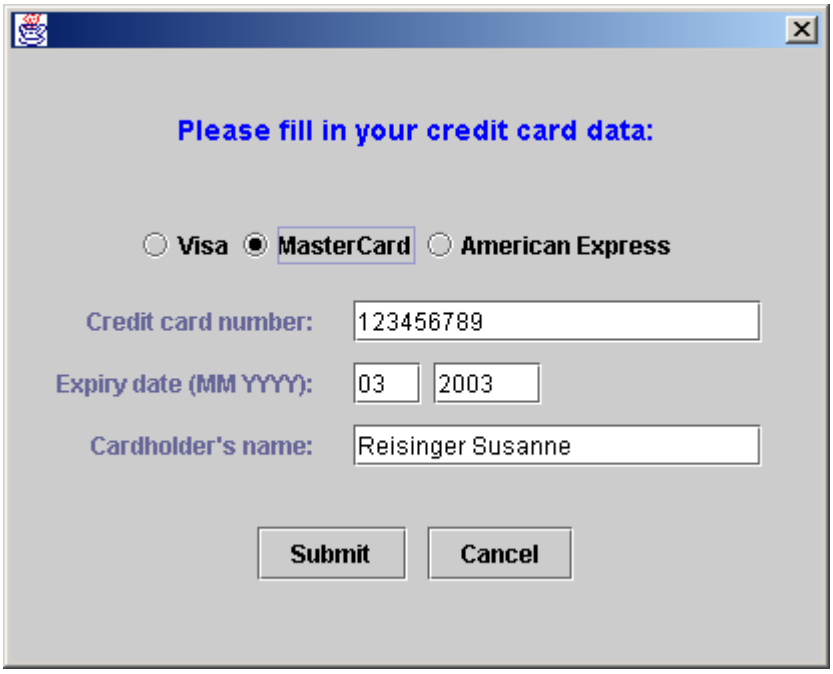


Abbildung 84: Eingabe der Kreditkartendaten des Benutzers

Der Agent migriert mit diesen Daten zum PortalServerAgent zurück und übergibt ihm diese Daten. Der PortalServerAgent setzt nach der Bezahlung die offenen Gebühren in der Datenbank auf bezahlt. Der WEMemberDataAgent erhält eine Statusmeldung über die Bezahlung und meldet entweder einen Fehler oder die erfolgreiche Durchführung aller offenen Gebühren.

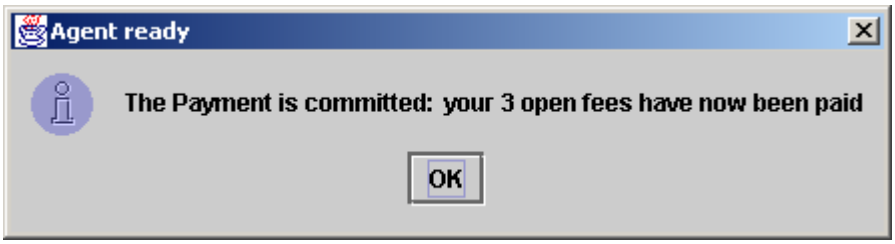


Abbildung 85: Rückmeldung durch den WEMemberDataAgent über den Status der Bezahlung der offenen Gebühren

Abstrahiert von der Funktion des Agenten verläuft der Lebenszyklus der Agenten und die Kommunikation immer wie in Abbildung 86 dargestellt:

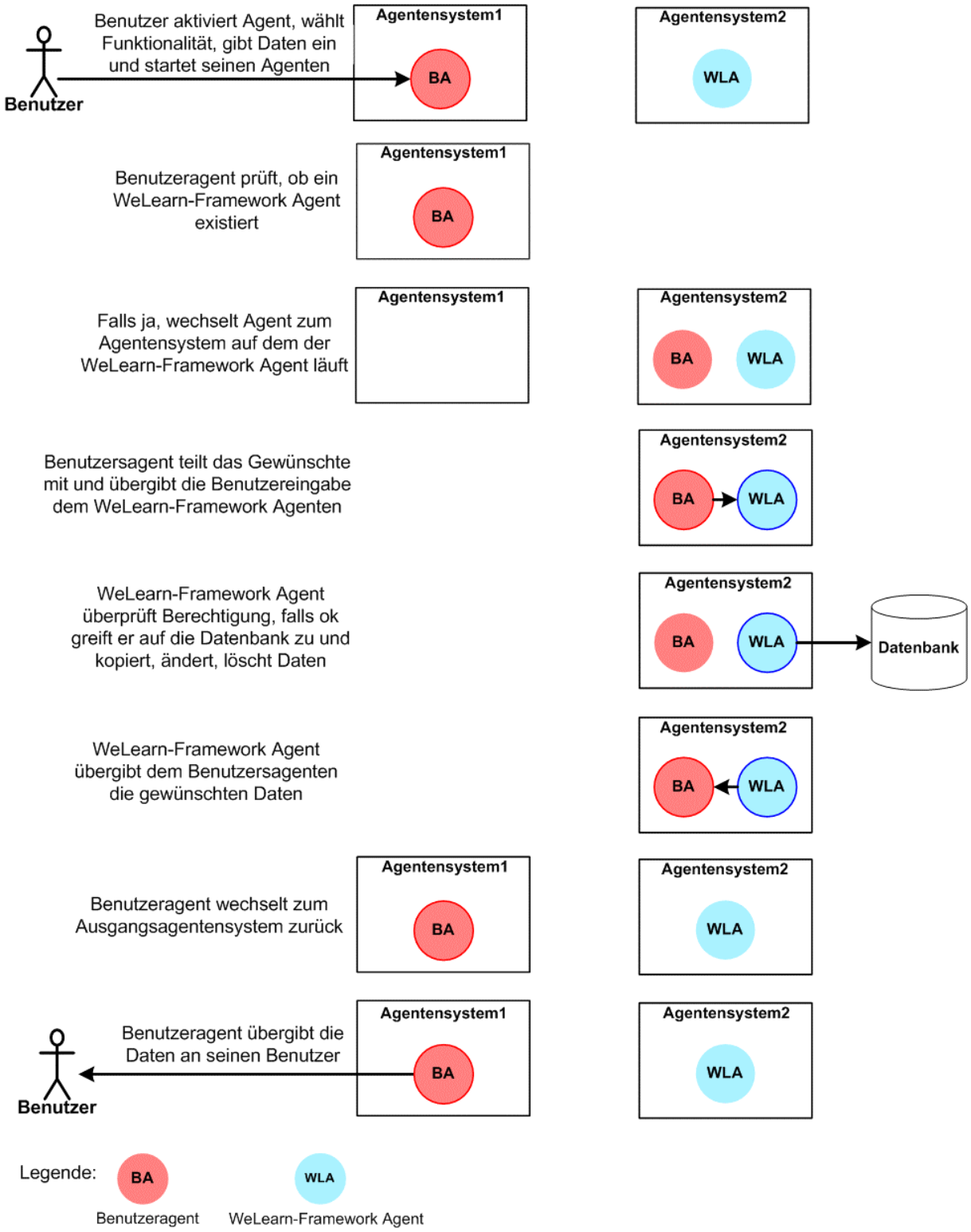


Abbildung 86: Lebenszyklus und Kommunikationsablauf der Agenten

Ablauf des Datenaustauschs

Die Kommunikation zwischen den Agenten basiert dabei auf XML-Protokollen, die Bestandteil des Agentensystems POND sind. Weiters wurden zum Austausch von Daten ebXML-Protokolle implementiert ([EBX]). Deren genauer Aufbau und deren Funktionsweise werden ausführlich in [PON02b] erläutert. Durch die genaue Definition der Struktur der Daten (DTD's/Schemata) und des Inhalts können verschiedenste Agenten auf die Daten zugreifen und mit dem Portalagenten interagieren.

Am konkreten Beispiel der Mitgliedsdaten sieht ein Datenaustausch folgendermaßen aus:

Schritt 1: Der Entwickler erstellt XML-Schemata (DTD´s), die die Struktur (Aufbau und Typ) der Daten definieren. Etwa wurde für die Mitgliedsdaten folgendes XML-Schema[1] erstellt:

```
<?xml version="1.0" encoding="UTF-8"?>
<!--XML Schema generated by Reisinger Susanne)-->
<xsd:schema targetNamespace="http://www.fim.uni-linz.ac.at" xmlns:xsd="http://www.w3.org/2000/10/XMLSchema"
xmlns="http://www.fim.uni-linz.ac.at" elementFormDefault="qualified">
        <xsd:element name="MemberDataDoc">
            <xsd:complexType>
                <xsd:sequence>
                    <xsd:element name="MemberData" type="MemberData" minOccurs="0" maxOccurs="unbounded"/>
                </xsd:sequence>
            </xsd:complexType>
        </xsd:element>
        <xsd:complexType name="MemberData">
            <xsd:sequence>
                <xsd:element name="Name">
                    <xsd:complexType>
                        <xsd:sequence>
                            <xsd:element name="FName" type="xsd:string"/>
                            <xsd:element name="SName" type="xsd:string"/>
                            <xsd:element name="MInitial" type="xsd:string" minOccurs="0"/>
                            <xsd:element name="Title" type="xsd:string" minOccurs="0"/>
                        </xsd:sequence>
                    </xsd:complexType>
                </xsd:element>
                <xsd:element name="BDate" type="xsd:date" minOccurs="0"/>
                <xsd:element name="Address">
                    <xsd:complexType>
                        <xsd:sequence>
                            <xsd:element name="Street" type="xsd:string"/>
                            <xsd:element name="PCode" type="xsd:integer"/>
                            <xsd:element name="City" type="xsd:string"/>
                            <xsd:element name="Country" type="xsd:string" minOccurs="0"/>
                            <xsd:element name="EMail" type="xsd:string" minOccurs="0"/>
                            <xsd:element name="TelNr" type="xsd:string" minOccurs="0"/>
                        </xsd:sequence>
```

[1] Dieses XML-Schema wurde bereits in Kapitel 4.3.3 näher erläutert.

```
                    </xsd:complexType>
                </xsd:element>
            </xsd:sequence>
            <xsd:attribute name="id" use="required"/>
            <xsd:attribute name="Gender" use="required">
                <xsd:simpleType>
                    <xsd:restriction base="xsd:string">
                        <xsd:enumeration value="Female"/>
                        <xsd:enumeration value="Male"/>
                    </xsd:restriction>
                </xsd:simpleType>
            </xsd:attribute>
        </xsd:complexType>
</xsd:schema>
```

Schritt 2: Der Entwickler bildet dieses Schema in der Datenbank ab.

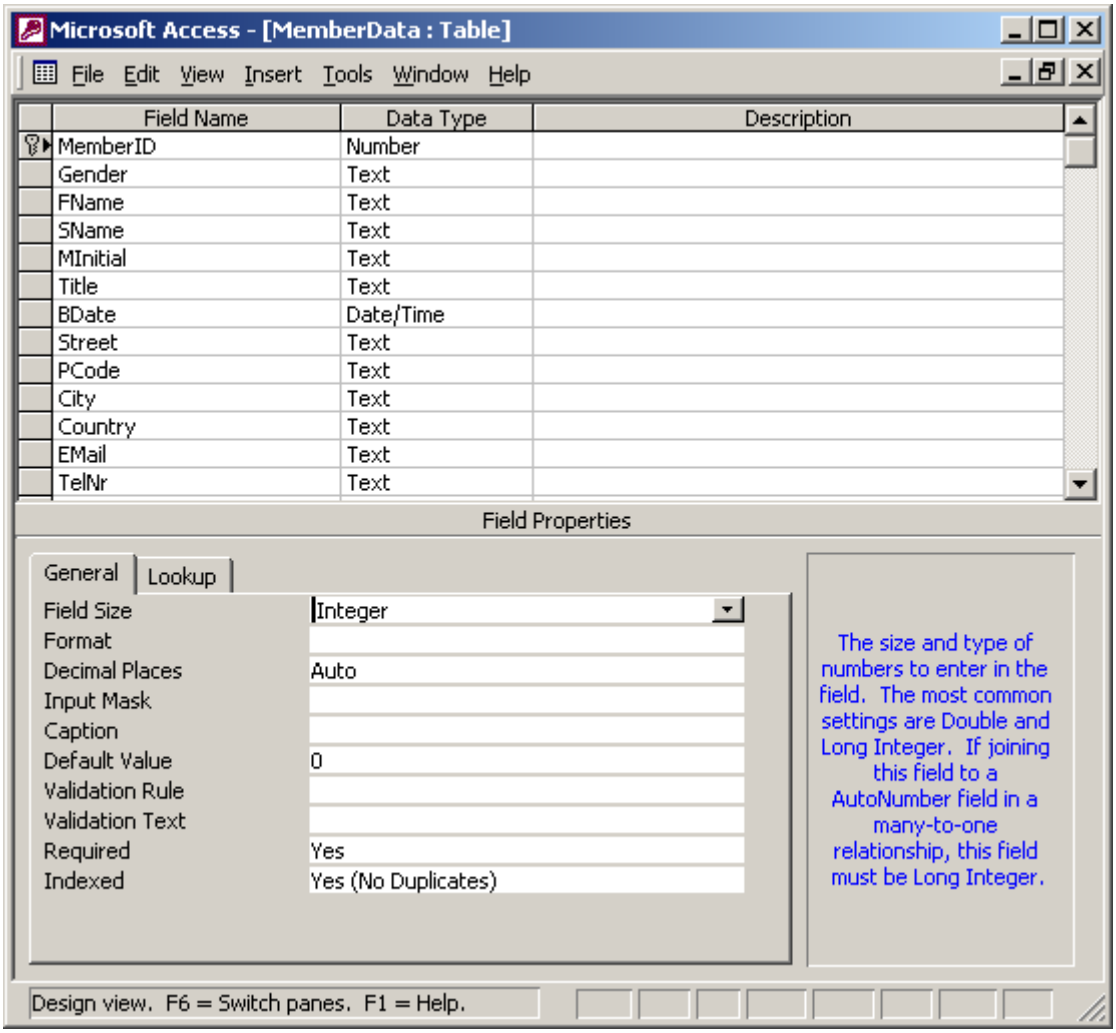


Abbildung 87: Datenbank-Tabelle für die Mitgliedsdaten

Schritt 3: Daten werden entsprechend diesem Schema in der Datenbank abgelegt.

Microsoft Access - [MemberData : Table]

MemberID	Gender	FName	SName	MInitial	Title	BDate	Street	PCode	City	Country	EMail	TelNr
1	female	Reisinger	Susanne		Dipl.-Ing.	19.04.1974	Altenbergerstr.	4040	Linz	Austria	sreisinger@fim.	+43(70)2468
2	male	Loidl	Reinhard	K	Dipl.-Ing.	14.06.1967	Altenbergerstr.	4040	Linz	Austria	loidl@utanet.at	+43(70)2468
0												

Record: 2 of 2

Datasheet View

Abbildung 88: Datenbankeinträge der Mitgliedsdaten

Schritt 4: Der Entwickler erstellt Agententypen und Agenten für die Benutzer (hier: Benutzeragent) und die Plattformseite (hier: WeLearn-Framework Agent) entsprechend den Strukturvorgaben.

Schritt 5: Der Benutzer aktiviert seinen Benutzeragenten und übergibt ihm die für die gewählte Funktion erforderlichen Daten.

Schritt 6: Der Benutzeragent erstellt ein XML-Dokument. Laut Funktion ist in diesem XML-Dokument die Mitgliedsidentifikation (MemberID) enthalten. Dieses schickt er dem WeLearn-Framework Agenten als Inhalt einer Nachricht.

Auszug aus dem Source-Code: Erstellen des XML-Dokuments mit der MemberID als Inhalt:

```
protected Document getXMLEncoded() throws SAXException
{
   try
   {
      String str="<?xml version=\"1.0\" encoding=\"UTF-
      8\"?>\n<!DOCTYPE GetMemberDataMessage SYSTEM
      \"file:/GetMemberDataMessage.dtd\">\n<GetMemberDataMessage/>";
      DocumentBuilderFactory dFactory =
      DocumentBuilderFactory.newInstance();
      DocumentBuilder dBuilder=dFactory.newDocumentBuilder();
      MessageParsingHandler handler=new MessageParsingHandler();
      dBuilder.setEntityResolver(handler);
      Document document = dBuilder.parse(new
      java.io.ByteArrayInputStream(str.getBytes()));
      Element memberIDEl=document.createElement("MemberID");
      document.getDocumentElement().appendChild(memberIDEl);
      memberIDEl.appendChild(document.createTextNode(memberID));
      return document;
   }
```

```
    catch(ParserConfigurationException t)
    {   // Should not happen
      throw new SAXException(t.toString());
    }
    catch(IOException t)
    {   // Should not happen
      throw new SAXException(t.toString());
    }
  }
```

Schritt 7: Der WeLearn-Framework Agent prüft die Berechtigung. Falls der Benutzer Daten abrufen darf, ruft der Agent die entsprechenden Daten aus der Datenbank ab und erzeugt ein XML-Dokument mit den Daten als Inhalt. Der Aufbau des XML-Dokuments entspricht der Struktur die das XML-Schema für Mitgliedsdaten vorgibt.

Auszug aus dem Source-Code: Erstellen eines XML-Dokuments mit den Daten der Datenbank als Inhalt und strukturiert nach dem XML-Schema für Mitgliedsdaten.

```
public org.w3c.dom.Document getMemberData(String id) throws
IllegalArgumentException,IllegalStateException, SQLException
{
  if(id==null || id.length()<1)
    throw new IllegalArgumentException("id must be provided");
  ResultSet rs=getSingleRow("SELECT * FROM MemberData WHERE
  MemberID="+id+";");
  org.w3c.dom.Document doc=createXMLDocument("MemberData");
  // Insert data
  insertMemberData(rs,doc.getDocumentElement(),doc);
  // Print the resulting XML document
  ByteArrayOutputStream bos=new ByteArrayOutputStream();
  try
  {
    OutputProperties op=new OutputProperties
    (outputProperties.getDefaultMethodProperties("xml"));
    op.setProperty(javax.xml.transform.OutputKeys.DOCTYPE_SYSTEM,
    "MemberData.dtd");
    Serializer serializer = SerializerFactory.getSerializer
    (op.getProperties());
    serializer.setOutputStream(bos);
    serializer.asDOMSerializer().serialize(doc);
  }
  catch(IOException e)
  {
    throw new IllegalStateException(e.toString());
  }
```

```
   System.out.println(new String(bos.toByteArray()));
   return doc;
}
protected org.w3c.dom.Document createXMLDocument(String name) throws
IllegalStateException
{
   try
   {
      String str="<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n<!DOCTYPE"
      +name+"Doc SYSTEM \"file:/"+name+".dtd\">\n<"+name+"Doc/>";
      DocumentBuilderFactory dFactory =
      DocumentBuilderFactory.newInstance();
      DocumentBuilder dBuilder=dFactory.newDocumentBuilder();
      MessageParsingHandler handler=new MessageParsingHandler();
      dBuilder.setEntityResolver(handler);
      org.w3c.dom.Document doc=dBuilder.parse(new
      java.io.ByteArrayInputStream(str.getBytes()));
      return doc;
   }
   catch(SAXException t)
   {
      if(t instanceof SAXParseException)
      {
         SAXParseException sp=(SAXParseException)t;
         System.out.println("Line: "+sp.getLineNumber());
         System.out.println("Column: "+sp.getColumnNumber());
         System.out.println("PublicID: "+sp.getPublicId());
         System.out.println("SystemID: "+sp.getSystemId());
      }
         throw new IllegalStateException("Error creating XML document
         (SAX): "+t.toString());
   }
   catch(ParserConfigurationException t)
   {   // Should not happen
      throw new IllegalStateException("Error creating XML document (Parser
      config): "+t.toString());
   }
   catch(IOException t)
   {   // Should not happen
      throw new IllegalStateException("Error creating XML document (IO):
      "+t.toString());
   }
}

private void insertMemberData(ResultSet rs,Node
parentNode,org.w3c.dom.Document doc) throws SQLException
{
   Element el=doc.createElement("MemberData");
   el.setAttribute("MemberID",rs.getString("MemberID"));
```

```
el.setAttribute("Gender",rs.getString("Gender"));
parentNode.appendChild(el);
Element nameNode=doc.createElement("Name");
el.appendChild(nameNode);
String datastr=rs.getString("FName");
if(datastr!=null)
{
  Element node=doc.createElement("FName");
  node.appendChild(doc.createTextNode(datastr));
  nameNode.appendChild(node);
}
datastr=rs.getString("SName");
if(datastr!=null)
{
  Element node=doc.createElement("SName");
  node.appendChild(doc.createTextNode(datastr));
  nameNode.appendChild(node);
}
datastr=rs.getString("MInitial");
if(datastr!=null)
{
  Element node=doc.createElement("MInitial");
  node.appendChild(doc.createTextNode(datastr));
  nameNode.appendChild(node);
}
datastr=rs.getString("Title");
if(datastr!=null)
{
  Element node=doc.createElement("Title");
  node.appendChild(doc.createTextNode(datastr));
  nameNode.appendChild(node);
}
java.sql.Date d=rs.getDate("BDate");
if(d!=null)
{
  Element node=doc.createElement("BDate");
  node.appendChild(doc.createTextNode(d.toString()));
  el.appendChild(node);
}
Element adrNode=doc.createElement("Address");
el.appendChild(adrNode);
datastr=rs.getString("Street");
if(datastr!=null)
{
  Element node=doc.createElement("Street");
  node.appendChild(doc.createTextNode(datastr));
  adrNode.appendChild(node);
```

```
    }
    datastr=rs.getString("PCode");
    if(datastr!=null)
    {
      Element node=doc.createElement("PCode");
      node.appendChild(doc.createTextNode(datastr));
      adrNode.appendChild(node);
    }
    datastr=rs.getString("City");
    if(datastr!=null)
    {
      Element node=doc.createElement("City");
      node.appendChild(doc.createTextNode(datastr));
      adrNode.appendChild(node);
    }
    datastr=rs.getString("Country");
    if(datastr!=null)
    {
      Element node=doc.createElement("Country");
      node.appendChild(doc.createTextNode(datastr));
      adrNode.appendChild(node);
    }
    datastr=rs.getString("EMail");
    if(datastr!=null)
    {
      Element node=doc.createElement("EMail");
      node.appendChild(doc.createTextNode(datastr));
      adrNode.appendChild(node);
    }
    datastr=rs.getString("TelNr");
    if(datastr!=null)
    {
      Element node=doc.createElement("TelNr");
      node.appendChild(doc.createTextNode(datastr));
      adrNode.appendChild(node);
    }
  }
```

Schritt 8: Der WeLearn-Framework Agent schickt danach an den Benutzeragenten eine Nachricht mit dem XML-Dokument als Inhalt.

Schritt 9: Der Benutzeragent extrahiert die Daten aus dem XML-Dokument und präsentiert sie seinem Benutzer.

Auszug aus dem Source-Code: Auslesen der Daten aus dem XML-Dokument

```
public MemberData extractMemberDataFromDoc (Document doc)
{
  MemberData data = new MemberData();
  NodeList nlist = doc.getElementsByTagName("FName");
  if (nlist.getLength() > 0 &&
  nlist.item(0).getFirstChild().getNodeValue()!=null)
    data.fName = (String) nlist.item(0).getFirstChild().getNodeValue();
  nlist = doc.getElementsByTagName("SName");
  if (nlist.getLength() > 0 &&
  nlist.item(0).getFirstChild().getNodeValue()!=null)
    data.sName = (String) nlist.item(0).getFirstChild().getNodeValue();
  nlist = doc.getElementsByTagName("MInitial");
  if (nlist.getLength() > 0 &&
  nlist.item(0).getFirstChild().getNodeValue()!=null)
      data.mInitial = (String)
  nlist.item(0).getFirstChild().getNodeValue();
  nlist = doc.getElementsByTagName("Title");
  if (nlist.getLength() > 0 &&
  nlist.item(0).getFirstChild().getNodeValue()!=null)
    data.title = (String) nlist.item(0).getFirstChild().getNodeValue();
  nlist = doc.getElementsByTagName("BDate");
  if (nlist.getLength() > 0 &&
  nlist.item(0).getFirstChild().getNodeValue()!=null)
    data.bDate = (String) nlist.item(0).getFirstChild().getNodeValue();
  nlist = doc.getElementsByTagName("Street");
  if (nlist.getLength() > 0 &&
  nlist.item(0).getFirstChild().getNodeValue()!=null)
    data.street = (String) nlist.item(0).getFirstChild().getNodeValue();
  nlist = doc.getElementsByTagName("PCode");
  if (nlist.getLength() > 0 &&
  nlist.item(0).getFirstChild().getNodeValue()!=null)
    data.pCode = (String) nlist.item(0).getFirstChild().getNodeValue();
  nlist = doc.getElementsByTagName("City");
  if (nlist.getLength() > 0 &&
  nlist.item(0).getFirstChild().getNodeValue()!=null)
    data.city = (String) nlist.item(0).getFirstChild().getNodeValue();
  nlist = doc.getElementsByTagName("Country");
  if (nlist.getLength() > 0 &&
  nlist.item(0).getFirstChild().getNodeValue()!=null)
    data.country = (String)
    nlist.item(0).getFirstChild().getNodeValue();
  nlist = doc.getElementsByTagName("EMail");
  if (nlist.getLength() > 0 &&
  nlist.item(0).getFirstChild().getNodeValue()!=null)
    data.email = (String) nlist.item(0).getFirstChild().getNodeValue();
  nlist = doc.getElementsByTagName("TelNr");
```

```
    if (nlist.getLength() > 0 &&
    nlist.item(0).getFirstChild().getNodeValue()!=null)
      data.telNr = (String) nlist.item(0).getFirstChild().getNodeValue();
    nlist = doc.getElementsByTagName("MemberData");
    if (nlist.getLength() > 0 &&
    nlist.item(0).getAttributes().getNamedItem("MemberID").getNodeValue()!
    =null)
     data.memberID = (String)
     nlist.item(0).getAttributes().getNamedItem("MemberID").getNodeValue()
     ;
    if (nlist.getLength() > 0 &&
    nlist.item(0).getAttributes().getNamedItem("Gender").getNodeValue()!=
    null)
     data.gender = (String)
     nlist.item(0).getAttributes().getNamedItem("Gender").getNodeValue();
    return data;
  }
```

Abstrahiert von spezifischen Agentenfunktionen, verläuft der Aufbau der Strukturen, der Austausch und die Gewinnung von Daten allgemein immer nach folgendem Schema (siehe Abbildung 89):

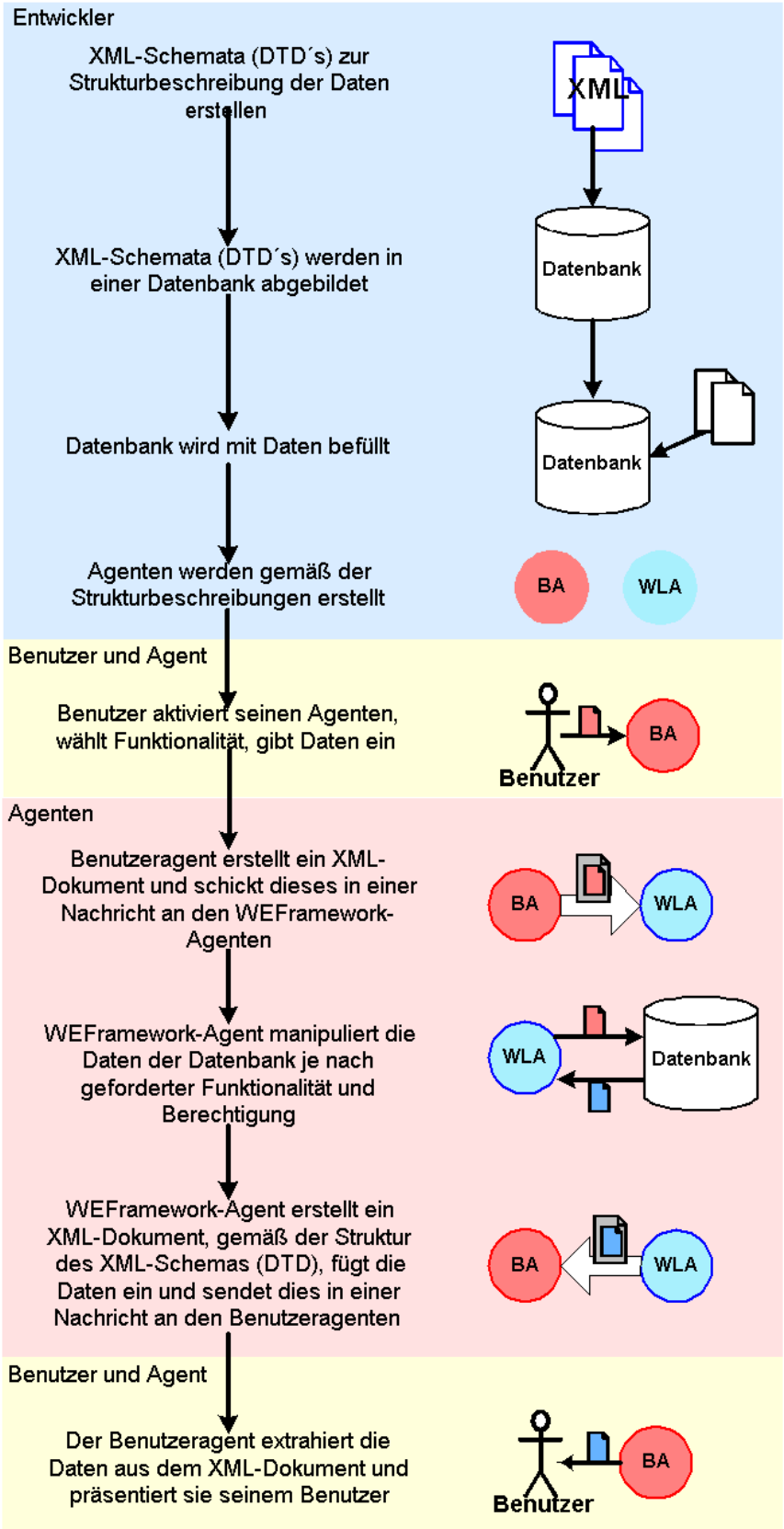


Abbildung 89: Aufbau der Strukturen, Ablauf des Datenaustauschs und der -gewinnung

7.3.4.2. Agent 2: CourseMDAgent

Dieselbe Backoffice-Funktionsweise, wie in Abbildung 86 dargestellt, wird auch bei Agenten des Typs CourseMDAgent eingesetzt. Auch der generelle Auflauf der Datenstrukturierung und –aufbereitung, des Datenaustauschs und der Datenmanipulation in der Datenbank ist deckungsgleich mit der in Abbildung 89 erläuterten Methodik.

Wesentlich für diesen Typ von Agenten ist, dass hierfür ein neues Web-Interface für WeLearn geschaffen wurde, welches dem Benutzer den Umgang mit den Agenten erheblich erleichtert. Der Agent wird dadurch für den Benutzer intuitiv bedienbar.

Generell arbeiten Agenten des Typs CourseMDAgent auf Basis des XML-Schemas „WeLearnCourseMetaData“. Dieses Schema definiert die Struktur von Kursmetadaten und wurde in 7.3.3.1.1 detailliert beschrieben.

Das prototypische Webinterface kann mittels eines beliebigen Webbrowsers aufgerufen und angezeigt werden. Für Benutzer des WeLearn-Frameworks ist das Webinterface direkt eingebettet. Sucht der Benutzer nun einen bestimmten Kurs, so kann er über dieses Webinterface die Daten für den gewünschten Kurs eintragen.

Im Hintergrund laufen Agenten des Typs CourseMDAgent (Benutzeragenten) in einem oder mehreren POND-Agentensystemen und Agenten vom Typ WeLearnFramework-Agent laufen getrennt von den Benutzeragenten in anderen POND-Agentensystemen.

Die Agentensysteme und die darin laufenden Agenten arbeiten vollkommen unabhängig von WeLearn. Für den Benutzer ist dies jedoch nicht ersichtlich. Er arbeitet ausschließlich mit und innerhalb seines gewohnten Systems, in diesem Fall mit WeLearn. Dies hat den Vorteil, dass er sich nicht auf mehrere unterschiedliche Systeme einstellen und deren Bedienung erlernen muss, sondern wie bisher weiterarbeiten kann, jedoch mit einer wesentlich erweiterten Funktionalität.

Ruft der Benutzer schließlich das Web-Interface für den CourseMDAgent auf, so kann er die gewünschten Kursdaten dort eintragen. Derzeit[2] ist es möglich einen genauen Kurstitel, mehrere Schlüsselwörter, ein Fachgebiet, die Zielgruppe, die gewünschte Sprache und eine Preisspanne anzugeben.

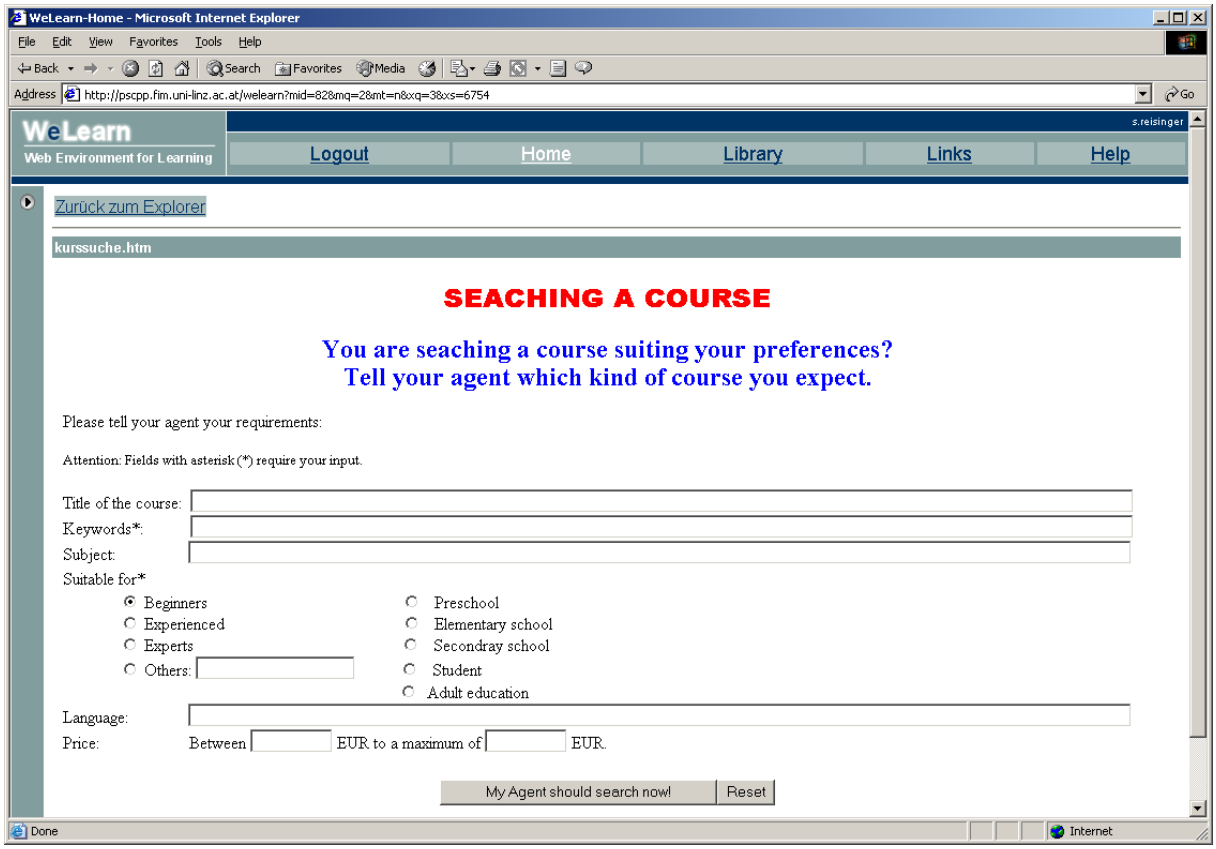


Abbildung 90: Web-Interface des CourseMDAgent

Nach Eingabe der gewünschten Daten kann der Benutzer seinen Agenten mit der Suche beauftragen. In diesem Beispiel (Abbildung 91) sucht der Benutzer einen Anfänger-Kurs zum Thema Algorithmen, im Fachgebiet der Informatik. Als gewünschte Sprache hat er Deutsch angegeben. Zusätzlich darf der Kurs nicht mehr als 100 Euro kosten.

---

[2] An dieser Stelle sei vermerkt, dass es sich hierbei zur Zeit um einen Prototypen handelt, der noch wenig Rücksicht auf graphische Aufbereitung und Benutzerführung nimmt.

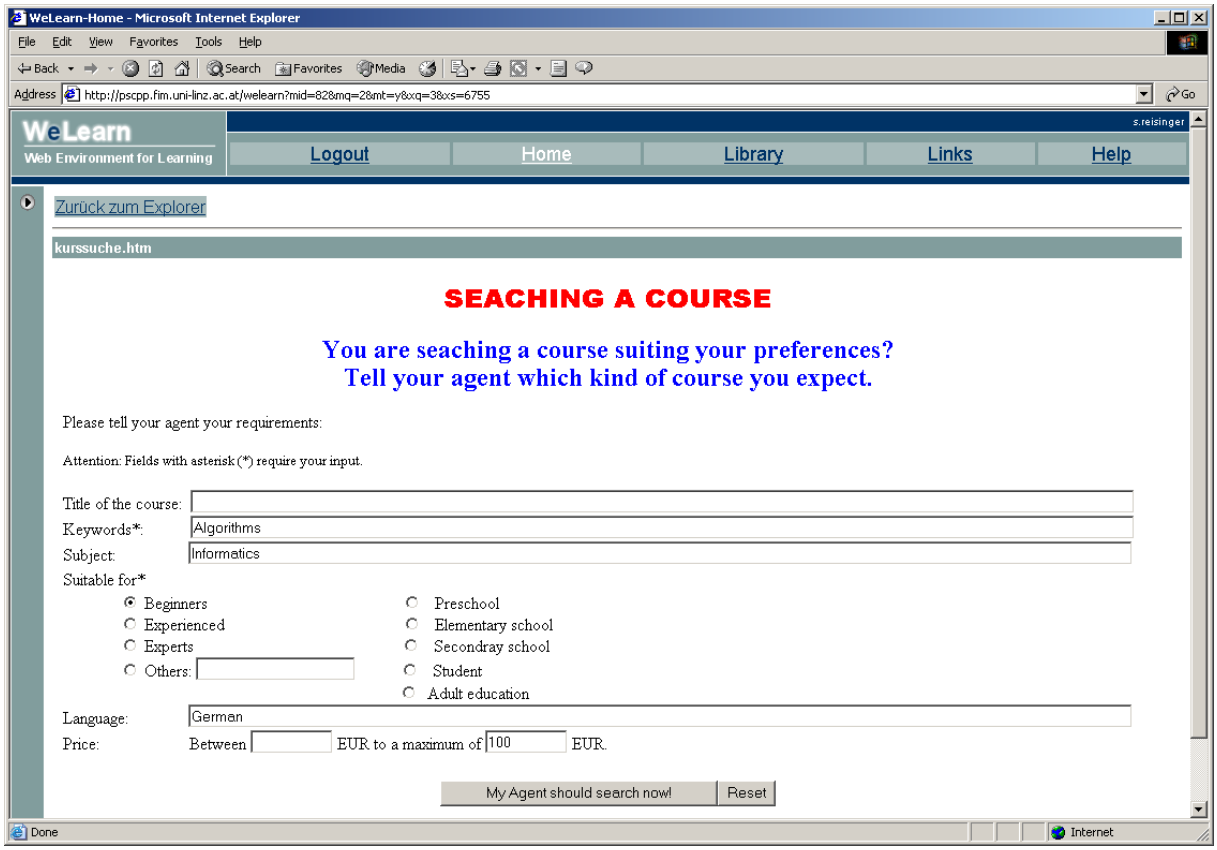


Abbildung 91: Web-Interface des CourseMDAgent mit den gewünschten Benutzerdaten

Der Benutzeragent (ein Exemplar des Typs CourseMDAgent) migriert anschließend zu einem Agentensystem auf dem ein WeLearn-Framework Agent läuft. Dieser sucht entsprechend der Struktur, die durch das XML-Schema (WeLearnCourseMetaData) vorgegeben ist, in der Datenbank nach den gewünschten Daten und gibt alle Kurse an den Benutzeragenten zurück, die auf die Angaben des Benutzers zutreffen.

Der Benutzeragent (CourseMDAgent) stellt anschließend das Ergebnis - die Daten aller gefundenen Kurse - im Web-Interface dar.

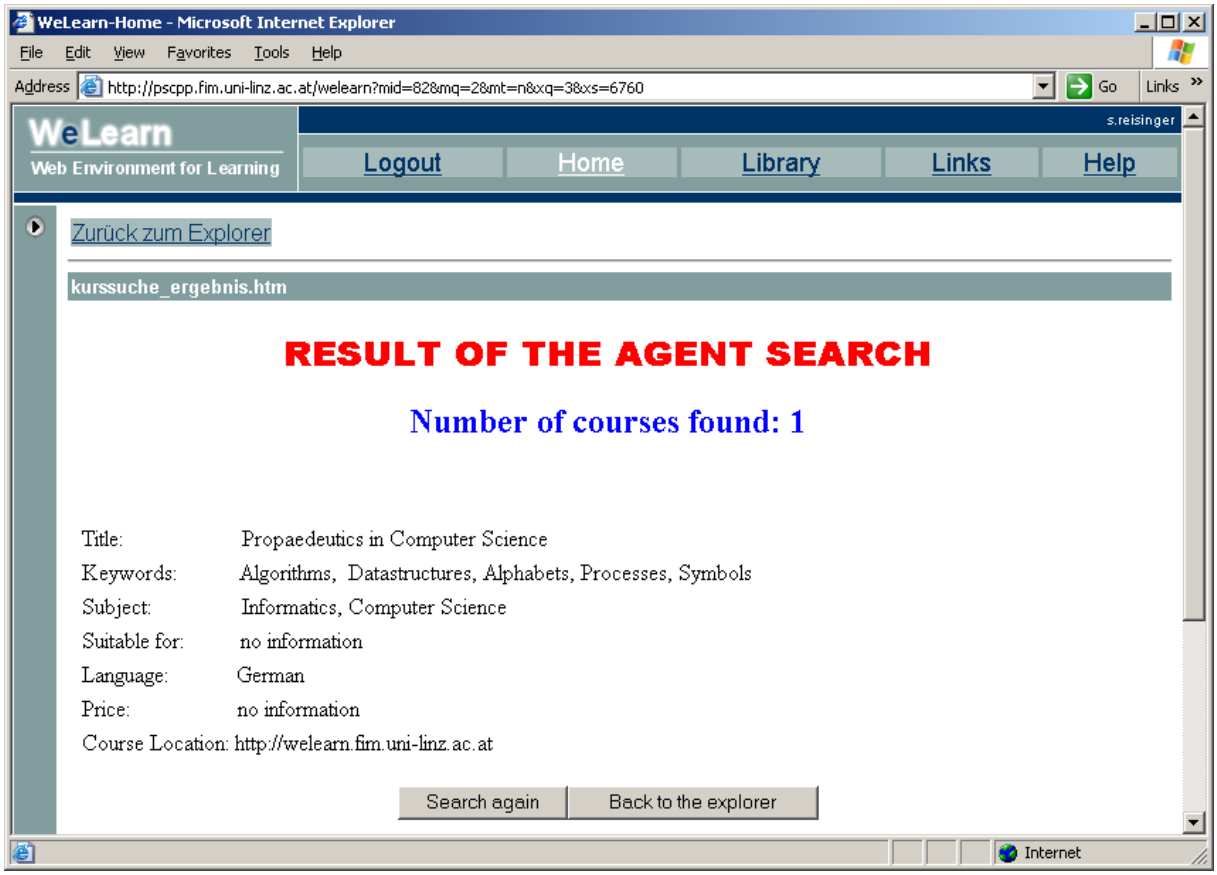


Abbildung 92: Web-Interface mit den gefundenen Kursen

Bei der durchgeführten Agentensuche wurde im Repository ein Kurs gefunden, der zu den Benutzerangaben passt. Dieser Kurs ist „Propaedeutics in Computer Science“, beinhaltet Informationen zum Thema Algorithmen, ist in Deutsch und unter http://welearn.fim.uni-linz.ac.at zu finden.

8. ZUSAMMENFASSUNG

Agenten und Distance Education haben bereits in unser Leben Einzug gehalten und werden in Zukunft noch viel stärker Bestandteil unseres Lebens sein: Auf der einen Seite Agenten als Helfer, als persönliche Assistenten, die uns direkt oder versteckt im Hintergrund unterstützen und auf der anderen Seite Distance Education als Notwendigkeit aufgrund einer immer kürzer werdenden Halbwertszeit von (produktspezifischen) Wissen. Lebensbegleitendes Lernen als fixer Bestandteil unseres Lebens setzt voraus, dass wir selbst aktiver im Lernprozess werden, uns selbst Informationen beschaffen und selbstorganisiert Lernen. Hier spielt Distance Education eine wesentliche Rolle.

Warum sollte man daher nicht Agenten und Distance Education gemeinsam einsetzen: Der Einsatz von Agenten ergibt letztendlich eine Win-Win-Win Situation für Benutzer, Distance Education als solches und für Agenten. Personen (Kursanbieter, Coaches, Lernende, etc.), die mit Distance Education zu tun haben, können durch Agenten bei vielen unterschiedlichen Tätigkeiten (Kurssuche, Wartung, Lernmaterialassemblierung, etc.) unterstützt und entlastet werden. Distance Education gewinnt dadurch an Attraktivität und nicht zuletzt werden für den Einsatz von Agenten entsprechende Strukturen geschaffen, auf denen Agenten aufsetzen und von denen sie aktiv profitieren können.

In dieser Arbeit wurden zunächst grundlegende Informationen zum Thema Agenten zusammengestellt und aufgezeigt. Definitionen, Anwendungsgebiete, geschichtliche Entwicklung, Agentensysteme und -plattformen, agentenorientierte Programmierung, Agentensprachen, etc. wurden dabei umfassend beleuchtet. In diesem Zusammenhang wurde auch auf XML (DTD´s, Schemata, XSLT, Parser, etc.) eingegangen.

Weiters wurden Aspekte der Distance Education (DE) beleuchtet. Unterschiedliche Lernmodelle, Vor- und Nachteile, wichtige Elemente, Durchführung von DE-Kursen, etc. wurden dabei erläutert. Zusätzlich wurde auf eine konkrete Umsetzung von Distance Education im Rahmen eines Projekts (PIAF) eingegangen. Speziell aufbereitetes Lernmaterial, Settings, die Lernplattform WeLearn und Pilotstudien wurden dabei vorgestellt.

Aufgrund der Erfahrungen im Bereich der Agenten und der Distance Education wurde eine Zusammenführung dieser beiden Themen in Angriff genommen. Zunächst wurde ein Prozess-

modell für Distance Education Kurse entwickelt. Dieses Prozessmodell zeigt notwendige Prozessschritte und Aufgaben von der Kursvorbereitung, über die Kursdurchführung bis hin zur Kursnachbereitung auf. Darin werden die Aufgaben im Lichte einer Rollenverteilung (Kursorganisator, Coach, Lernenden, etc.) präsentiert und konkrete Umsetzungshinweise gegeben. Im Prozessmodell wurde auch die Rolle von Agenten und deren Einsatzmöglichkeiten detailliert dargelegt.

Weiters wurde eine agenten-orientierte Wissensplattform (knowledge platform for agents – KNOWPAG) entwickelt. Die Basis dafür bietet das XML-Schema WeLearnCourseMetaData. Dieses legt die Struktur und den Typ von Metadaten fest. Diese Metadaten geben Aufschluss über den Aufbau, Inhalt und die Module eines DE-Kurses. Darüber hinaus gibt dieses Schema den Aufbau von Datenbanken zur Informationsspeicherung vor und ist Richtlinie für Agentenprotokolle und Agententypen, die auf diese Daten zugreifen.

Die generelle Funktionsweise, der Lebenszyklus der Agenten, die Kommunikation zwischen den Agenten und der Aufbau der Strukturen werden anhand von konkreten Agenten dargestellt und anschließend allgemeingültig abstrahiert.

Die hier vorgestellten Agenten sind auf Basis des POND-Agentensystems entworfen und entwickelt worden. Um diese Benutzern der Lernplattform WeLearn zugänglich zu machen, wurde ein Prototyp eines Web-Interfaces entwickelt. Dadurch können die Benutzer in ihrem gewohnten System weiterarbeiten ohne sich mit anderen Systemen auseinander setzen zu müssen und dennoch von den Agenten und deren Funktionalität profitieren.

Es zeigt sich also, dass Agenten im Bereich der Distance Education auf vielfältigste Weise eingesetzt werden können. Sinnvoll ist dies jedoch nur, wenn dafür auch entsprechende Strukturen und Definitionen geschaffen werden, sodass der Einsatz der Agenten auch den gewünschten Effekt bringt.

Literatur

[ALB01] Albayrak, S.: Agent-oriented technology for communications. Communications of the ACM, Vol. 44, No. 4, 2001

[AND00] Anderson, R.: Professional XML. Wrox Press, Birmingham, 2000

[ARC01] Archer, T.: Inside C#. Microsoft Press, Redmond, USA, 2001

[AST99] Astleitner, H., Sindler, A.: Pädagogische Grundlagen virtueller Ausbildung. Wien: WUV, 1999

[AUS62] Austin, J. L.: How to do things with Words. Harvard University Press, 1962

[BAT95] Bates, A. W.: Technology, Open Learning and Distance Education New York/London: Routledge/Melbourne: Thomas Nelson Australia, 1995

[BAT96] Bates, A. W.: The Impact of Technological Change on Open and Distance Learning. Keynote Presentation in Proceedings of the 2nd International Conference on Open Learning. December 1996. Brisbane, Qld Australia, 1996

[BAU97] Baumgartner, P.: Evaluation vernetzten Lernens: 4 Thesen. In S. Hartmut (Hrsg.), Virtueller Campus – Forschung und Entwicklung für neues Lehren und Lernen (S. 131-144). Münster: Waxmann, 1997

[BAU01] Bauer, B., Müller, J. P., Odell, J.: Agent UML: A Formalism for Specifying Multiagent Interaction. Agent-Oriented Software Engineering, Paolo Ciancarini and Michael Wooldridge eds., Springer-Verlag, Berlin, pp. 91-103, 2001. (Held at the 22nd International Conference on Software Engineering (ISCE)).

[BIG01] Bigus, J. P., Bigus, J.: Constructing Intelligent Agents using Java. John Wiley & Sons, Inc., New York, 200

[BOE99] Böhme, T.: Agentensysteme - aktueller Entwicklungsstand und Konzeption eines universellen News-Watcher-Agent. Diplomarbeit gefördert durch

Fraunhofer-Institut für Arbeitswirtschaft und Organisation, Stuttgart, Institut für Informatik, Universität Leipzig, 1999

[BOX00] Box, D., Skonnard, A., Lam J.: Essential XML: beyond markup. Addison-Wesley, Boston, 2000

[BRE98] Brenner, W., Zarnekow, R., Wittig, H.: Intelligente Softwareagenten: Grundlagen und Anwendungen. Springer-Verlag, Berlin, 1998

[BRI92] Briggs, L.J., Gagné, R., Wagner, W.W.: Principles of instructional design. 4. Auflage, Harcourt, Brace & Javanovich, Orlando, 1992

[BRO96] Brockhaus. Enzyklopädie. 24 Bände. (20. Auflage, erschienen zwischen 1996 und 1999, unter Berücksichtigung der neuen Rechtschreibreform)., F. A. Brockhaus, Mhm., 1996

[CAG97] Caglayan, A., Snorrason, M., Mazzu, J., Jacoby, J., Jones, R., Kumar, K.: Open sesame - a learning agent engine. Applied Artificial Intelligence, 11(5):393-412, 1997

[CAG98] Caglayan, A. K., Harrison, C. G.: Intelligente Software-Agenten. Grundlagen, Technik und praktische Anwendung im Unternehmen. Hanser Verlag München Wien, 1998.

[CAM83] Cammarata, S., McArthur, D., Steeb, R.: Strategies of cooperation in distributed problem solving. IJCAI-83, pages 767-770, 1983

[CHA96] Chang, J. W., Scott, C. T.: Agents based Workflow: TRP Support Environment (TSE). Computer Networks and ISDN Systems, 1996, im WWW http://orgwis.gmd.de/projects/W4G/papers/chang.html

[CHA00a] Chaudhri, A. B., Zicari, R.: Succeeding with object databases: a practical look at today's implementations with Java & XML. Wiley, New York, 2000

[CHA00b] Charlton, P., Cattoni, R., Potrich, A., Mamdani, E.H.: Evaluating the FIPA Standards and Its Role in Achieving Cooperation in Multi-Agent Systems. Multiagent Systems, Internet and Applications (HICSS-33 Minitrack), 2000

[CHE95] Chess, D., Grosof, B., Harisson, C., Levine, D., Parris, C., Tsudik, G.: Itinerant Agents for Mobile Computing. Research Paper des IBM T. J. Watson Research Center, Yorktown Heights, USA, 1995.

[CIA01] Ciancarini, P., Wooldridge, M.: Agent-Oriented Software Engineering: The State of the Art. Lecture Notes in AI Volume 1957, Springer-Verlag, 2001

[COH90a] Cohen, P. R., Levesque, H. J.: Intention is choice with commitment. Artificial Intelligence, Vol. 42, 1990

[COH90b] Cohen, P. R., Levesque, H. J.: Rational interaction as the basis for communication. In Cohen, P. R., Morgan, J., and Pollack, M. E., editors, Intentions in Communication, The MIT Press: Cambridge, MA., 1990

[CRA01] Cranefield, S., Haustein, S., Purvis, M.: UML-based ontology modelling for software agents. Proceedings of the Workshop on Ontologies in Agent Systems, 5th International Conference on Autonomous Agents, 2001.

[DAV83] Davis D., Smith, R. G.: Negotiation as a Metaphor for Distributed Problem Solving. Artificial Intelligence, Vol. 20, 1983

[DIK01] Dikaiakos, M., Kyriakou, M., Samaras, G.: Benchmarking Mobile-Agent Systems. Technical Report TR-2001-2a, Department of Computer Science, University of Cyprus, November 2001

[DIV02a] Divotkey R., Remplbauer D.: Design and Implementation of a web-based Learning Environment. Diplomarbeit, Institut für Informationsverarbeitung und Mikroprozessortecknik, J.K. Universität Linz, 2002

[DIV02b] Divotkey, R., Mühlbacher, J.R., Reisinger, S., Remplbauer, D.: The WeLearn Distance Teaching Framework. Granada: EDEN (European Distance Education Network), 2002 Eden Annual Conference, Open and Distance Learning in Europe and Beyond Rethinking International Co-operation, Conference Proceedings, 2002, 308-314 (auch als SYSPRO 78/02, Mai 2002)

[DIX92] Dix A. J., Miles, V. C.: Version control for asynchronous group work. YCS 181, Department of Computer Science, University of York, (Poster presentation HCI'92: People and Computers VII), 1992

[DOB00] Doberkat, E.: Agenten – Ein kleiner Überblick. LOG IN, Nr. 3-4/2000, LOG IN Verlag, Berlin, 2000

[DOU99] Doug, L.: Concurrent Programming in Java: Design Principles and Patterns by. Second edition, Addison-Wesley, 1999

[DUR87] Durfee, E., Lesser, V.: Using partial global plans to coordinate distributed problem solvers. Proceedings of the Tenth International Joint Conference on Artificial Intelligence, pages 875-883, 1987

[DUR01] Durfee, E.: Scaling up agent coordination strategies. IEEE Computer Vol. 34, No. 7, 2001

[EDE01] Eder, J., Strametz, W.: Composition of XML-Transformations. EC-Web (Electronic Commerce and Web Technologies) 2001, München, 2001

[EIC94] Eichmann, D.: Ethical Web Agents. Second International World-Wide Web Conference: Mosaic and the Web, Chicago, IL, pages 3-13, 1994

[EXC02] Excelente-Toledo, C. B., Jennings, N.R.: Learning to select a coordination mechanism. Proc. 1st Int Joint Conf. on Autonomous Agents and Multi-Agent Systems, Bolgna, Italy, 1106-1113, 2002

[FAR96] Farmer, W. M., Guttman, J. D., Swarup, V.: Security for Mobile Agents: Issues and Requirements. Proceedings of the 19th National Information System Security Conference, Baltimore, 1996

[FIN94] Finin, T., Fritzson, R., McKay, D. McEntire, R.: KQML as an Agent Communication Language. Proceedings of the 3rd International Conference on Information and Knowledge Management (CIKM´94), ACM Press, 1994

[FIS00] Das Fischer Wirtschaftslexikon. 3., aktualisierten und erweiterten Ausgabe, Fischer Taschenbuch Verlag, Frankfurt am Main, 2000

[FLE01] Flehmig, M.: Integration und Personalisierung - Zur Realisierung essentieller Aspekte von Portal-Systemen. in: Proceedings Int. Workshop "E-Commerce: Systemunterstützung für den Umgang mit Daten und Prozessen in vernetzten Anwendungsumgebungen", Wien, 2001

[FRI01] Fricke, S., Bsufka, K., Keiser, J., Schmidt, T., Sesseler, R., Albayrak S.: Agent-based telematic services and telecom applications. Communications of the ACM, Vol. 44, No. 4, 2001

[GEN92] Gensereth, M. R., Fikes, R. E.: Knowledge Interchange Format, version 3.0. Reference Manual, Standford University, 1992

[GEO83] Georgeff, M: Communication and interaction in Multi-agent Planning, Proceedings of the 3rd National Conference on Artificial Intelligence, pp.125-129, 1983

[GEO84] Georgeff, M.: A theory of action for multiagent planning. Proceedings AAAI-84, pages 121-125, Los Altos, CA. Morgan Kaufmann, 1984

[GIE97] Giegerich, R., Füllen, G.: Neue Formen der Internet-basierten Lehre in der Bioinformatik. In S. Hartmut, (Hrsg.), Virtueller Campus – Forschung und Entwicklung für neues Lehren und Lernen (S. 77-89). Münster: Waxmann, 1997

[GOE01] Goethe, J. W.: Götz von Berlichingen mit der eisernen Hand. Suhrkamp, 2001

[GOL98] Golfarb, C. F., Prescod, P.: The XML handbook. Prentice-Hall, New York, 1998

[GRA96] Gray, R. S.: Agent Tcl: A flexible and secure mobile-agent system, Proceedings of the Fourth Annual Tcl/Tk Workshop (TCL 96), 1996

[GRA98] Gray, R. S., Kotz, D., Cybenko, G., Rus, D.: D'Agents: Security in a multiple-language, mobile-agent system, 1998

[GUI97] Guilfoyle, C., Jeffcoate, J., Stark, H.: Agents on the Web: Catalyst for E-Commerce. Ovum Reports, Ovum Ltd., London, 1997

[GUE96] Gürer, D.W., Khan, I, Ogler, R.: An Artificial Intelligence Approach to Network Fault Management. SRO International, Menlo Park, USA, 1996.

[HAY85] Hayes-Roth, B.: A blackboard architecture for control. Artificial Intelligence, Vol. 26, 1985

[HER01] Herzog, O., Timm, I.J., Tönshoff, H.K., Woelk, P.O.: Adaptives und kooperatives Verhalten in Agentensystemen. Proceedings des 3. Kolloquiums des DFG-Schwerpunktprogramms "Intelligente Softwareagenten und betriebswirtschaftliche Anwendungsszenarien", Hameln, 10.-12.05.2001. TZI-Bericht Nr. 23/2001. Technologie-Zentrum Informatik, Bremen, 2001.

[HEW77] Hewitt, C.: Viewing Control Structures as Patterns of Passing Messages. Artificial Intelligence, Vol. 8, Nr.3, North-Holland Publishing Company, Amsterdam, 1977

[HOH99] Hohl, F.: A Protocol to Detect Malicious Hosts Attacks by Using Reference States. Fakultät Informatik Fakultätsbericht 1999/09 1999

[HOL98] Holzner, S.: XML Complete. McGraw-Hill, New York, 1998

[HOL01] Holzner, S.: Inside XML. New Riders, USA, 2001

[IBM96] IBM: Intelligent Agent Resource Manager. Open Blueprint, G325-6592-0, 1996

[JAC01] Jacobson, I., Övergaard, G.: Making the Software Process Transparent by Using Intelligent Agents. 15th European Conference on Object-Oriented Programming, Hungary, 2001

[JEN96] Jennings, N. R., Wooldridge, M.: Software Agents. IEEE Review, 42(1), 1996

[JEN98] Jennings, N. R., Sycara, K. P., Wooldridge, M.: A Roadmap of Agent Research and Development. Journal of Autonomous Agents and Multi-Agent Systems, 1(1), 1998

[JEN00] Jennings, N. R.: On Agent-Based Software Engineering. Artificial Intelligence, 117(2), 277-296, 2000

[KAY84] Kay, A.: Computer Software. Scientific American 251(3): 53–59, 1984

[KER02] Kerres, M.: Online- und Präsenzelemente in hybriden Lernarrangements kombinieren. In A. Hohenstein und K. Wilbers (Hg.), Handbuch E-Learning, Fachverlag Deutscher Wirtschaftsdienst, Köln, 2002

[KOE99] Koenemann, J.: Intelligente Agenten: Eine Kurzübersicht. DFN-Symposium „Fortgeschrittene Kommunikationstechnik“ Berlin, 1999

[LAB97] Labrou, Y., Finin, T.: A Proposal for a new KQML Specification. Technical report TR CS-07-03, University of Maryland, Baltimore, 1997

[LAN94] Langer, G., Wolff, M.: Objektorientierte Unternehmensmodellierung mit AENEIS. Stuttgard, ipro Tool GmbH, 1994

[LED01] Leder, E.: E-Learning in der Wirtschaft am Beispiel von UBS, In Web-basiertes Lernen, Fortbildungsseminar in Informatik, Institut für Informatik, Universität Zürich, 2001

[LIE01a] Lieberman, H., Fry, C., Weitzman, L.: Exploring the reconnaissance agents. Communications of the ACM, Vol. 44, No. 8, 2001

[LIE01b] Lieberman, H., Rosenzweig, E., Singh, P.: Aria: An Agent for annotating and retrieving images. IEEE Computer Vol. 34, No. 7, 2001

[LES81] Lesser, V., Corkill, D. D.: Functionally-Accurate Cooperative Distributed Systems. IEEE Trans. Systems, Man, and Cybernetics, special issue on distributed problem-solving, 11(1), pp. 81- 96, 1981

[LIB01] Liberty, J.: Programming C#. O´Reilly Associates, Sebastopol, USA, 2001

[LIN95] Lingnau, A., Drobnik, O., Dömel, P.: An HTTP-based Infrastructure for Mobile Agents. Fourth International World Wide Web Conference “The Web Revolution” Boston, Massachusetts, USA, 1995

[LOM01] Lomuscio, A., Wooldridge, M., Jennings, N. R.: A Classification Scheme for Negotiation in Electronic Commerce. Dignum F., Sierra, C., editors, Agent-

Mediated Electronic Commerce: A European AgentLink Perspective. Springer-Verlag, Lecture Notes in AI, Vol. 1991, 2001.

[LUC68] Lucas, P., Lauer, P., Stigleitner, H.: Method and Notation for Formal Definition of Programming Languages. TR 25.087, IBM Laboratory, Vienna, 1968

[LUC69] Lucas, P. Walk, K.: On the Formal Description of PL/I. Ann. Review in Automatic Programming, Vol. 6, Part 3, pp. 105—182, 1969

[MAA01] Maamar, Z., Mansoor, W., Yahyaoui H., Van den Heuvel, WJ.: Software Agents and Wireless E-Commerce, SIGecom Exchanges, Vol. 2.3, ACM, 2001.

[MAE94] Maes, P.: Agents that reduce work and information overload. Communications of the ACM, 37(7), ACM-Press, New York, USA, 1994

[MAE97] Maes, P.: Agents that Reduce Work and Information Overload. In: Software Agents, edited by Jeffrey M. Bradshaw, AAAI Press/MIT Press, 1997.

[MAG96] Magedanz, T., Rothermel, K., Krause, S.: Intelligent Agents: An Emerging Technology for Next Generation Telecommunications?. INFOCOM´96, San Francisco, USA, 1996

[MAN01] Mandl, H.: Auf dem Weg zu einer neuen Kultur des Lehrens und Lernens?, In Web-basiertes Lernen, Fortbildungsseminar in Informatik, Institut für Informatik, Universität Zürich, 2001

[MAY98] Mayer, R. E.: Multimedia aids to problem-solving transfer. Paper presented at Symposium on “Transfer: Recent findings and future perspectives”. San Francisco: International Congress of Applied Psychology, 1998 [zitiert nach REI01]

[MCG69] McGee, W.C.: Gerneralized File Processing. In Annual Review in Automatic Programming 5 (Hrsg.: M.I. Halpern, C.J. Shaw), Pergamon Press, Oxford, 1969

[MIC01] Michelbacher, K.: Web-basierte Unterstützung selbstorganisierter Lernprozesse, In Web-basiertes Lernen, Fortbildungsseminar in Informatik, Institut für Informatik, Universität Zürich, 2001

[MOO90] Moore, R. C.: A formal theory of knowledge and action. Allen, J. F., Hendler, J., and Tate, A., editors, Readings in Planning, Morgan Kaufmann Publishers: San Mateo, CA., 1990

[MUE73] Mühlbacher, J.R.: Bäume und Arboreszenzen, ihre Notation und Anwendung. in: Probleme beim Aufbau betrieblicher Informationssysteme, Hrgsg. von H. Hansen, Verlag moderne Industrie, 1973

[MUE75] Mühlbacher, J.R.: Datenstrukturen. Carl Hanser Verlag, München Wien, 1975

[MUE01a] Mühlbacher, J. R., Reisinger, S., Sonntag, M.: Intelligent Agents and XML – A method for accessing webportals in both B2C and B2B E-Commerce. in: R. Corchuelo, A. Ruiz, J. Mühlbacher, J.D. García-Consuegra: Workshop on Object-Oriented Business Solutions Proceedings book. Budapest, 2001

[MUE01b] Mühlbacher, J. R.: Concepts, Aims and Problems which Modelling in OR and OOP have in Common; Essays in Honor of Paul Stähly; in: Models, Methods and Decision Support for Management; P.Kischka, U.Leopold-Wildburger, R.H.Möhring, F-J.Radermacher (Hrsg.), Physica-Verlag Heidelberg 2001, pp.139-146, ISBN 3-7908-1373-7

[MUE02a] Mühlbacher, J.R., Mühlbacher, S.C., Reisinger, S.: Learning Arrangements and Settings for Distance Teaching/Coaching/Learning: Best practice report. In: Hofer Christian, Chroust Gerhard (Ed.): IDIMT-2002. 10th Interdisciplinary Information Management Talks. Linz: Universitätsverlag Rudolf Trauner 2002, (auch als SYSPRO 79/02, Juni 2002)

[MUE02b] Mühlbacher, S.C.: Distance Learning, Ablauf und Evaluierung einer Lehrveranstaltung aus Informatik. Diplomarbeit, Institut für Psychologie, University of Vienna, Austria, 2002

[MTC01] Das M+T Computerlexikon 2001, Verlag Markt+Technik, München, 2000

[NOR94] Norman, D. A.: How might people interact with agents. Communications of the ACM, 37(7), ACM-Press, New York, USA, 1994

[NWA96a] Nwana, H. S.: Software Agents: An Overview. Knowledge Engineering Review, 11(3), 1996, im WWW: http://www.cs.umbc.edu/agents/introduction/ao/

[NWA96b] Nwana, H. S., Lee, L. C., Jennings, N. R.: Coordination in Software Agent Systems. The British Telecom Technical Journal, 14(4), 1996

[OBR98] O`Brien, P., Nicol, R.: FIPA - Towards a Standard for Software Agents BT Technology Journal, Vol 16:3, pp. 51-59, 1998

[ODE00] Odell, J., Van Dyke Parunak, H., Bauer, B.: Extending UML for Agents. Proc. of the Agent-Oriented Information Systems Workshop at the 17th National conference on Artificial Intelligence (AOIS Workshop at AAAI 2000), 2000

[PAL99] Pallmann, D.: Programming Bots, Spiders and Intelligent Agents in Microsoft Visual C++. Microsoft Press, Redmond, 1999

[PAP01] Papazoglou, M. P.: Agent-oriented technology in support of e-business. Communications of the ACM, Vol. 44, No. 84 2001

[PAR01] Van Dyke Parunak, H., Odell, J.: Representing Social Structures in UML. Proceedings of the Agent-Oriented Software Engineering Workshop, Agents 2001, 2001

[PAY02] Payne, T.R., Singh, R. Sycara. K.: Calendar Agents on the Semantic Web. IEEE Intelligent Systems, Vol. 17, No. 3, 2002

[RAO95] Rao, A., Georgeff, M.: BDI Agents: From Theory to Practice. Technical Report, Australian Artificial Intelligence Institute, Melbourne, 1995

[REE02] Reed, C., Norman, T. J., Jennings, N.R.: Negotiating the semantics of agent communication languages. Computational Intelligence 18 (2) 229-252, 2002

[REI01] Reinmann-Rothmeier, G., Mandl, H. (Hrsg.): Virtuelle Seminare in Hochschule und Weiterbildung – Drei Beispiele aus der Praxis. Bern: Hans Huber, 2001

[REI02] Reisinger, S.: The WeLearn Distance Teaching Framework. Vortrag im Rahmen der Annual Eden (European Distance Education Network) Conference 2002, Granada, Juni, 2002

[ROB02] Robinson, S., Allen, K.S., Cornes O., et al.: Professional C# 2nd ed. Wrox Press, Chicago, 2002.

[ROT97] Rothermel, K.: Mobile Software-Agenten: Chancen und Risiken. Universität Stuttgart, Folien zu einem Vortrag im Rahmen der Tagung „Kommunikation in verteilten Systemen“, Braunschweig, 1997

[ROT01] Rothenfluh, T., Aeppli, J.: Didaktisches Design für web-basierte Lern-umgebungen, In Web-basiertes Lernen, Fortbildungsseminar in Informatik, Institut für Informatik, Universität Zürich, 2001

[RUM91] Rumbaugh, J.: Object oriented modeling and design. Prentice Hall, New York, 1991

[RUS95] Russel, S. J., Norvig, P.: Artificial Intelligence: A Modern Approach. Englewood Cliffs, 1995.

[SCH01] Schauer, H.: Web-basiertes Lernen in Aus- und Weiterbildung – ein Paradigmenwechsel?, In Web-basiertes Lernen, Fortbildungsseminar in Informatik, Institut für Informatik, Universität Zürich, 2001

[SEA69] Searle, J. R.: Speech Acts. Cambridge University Press, 1969

[SMI80] Smith, R. G.: The contract net protocol: High-level communication and control in distributed problem solver. IEEE Transactions on Computers, 29(12), 1980

[SON02a] Sonntag, M., Mühlbacher, J.R., Reisinger, S.: Personalization of Web-Based Interfaces for Humans and Agents, Applied to E-Government Portals. In: Maria A. Wimmer (Ed.): Knowledge Management in e-Government. KMGov-2002. Linz: Universitätsverlag Rudolf Trauner 2002, 210-222

[SON02b] Sonntag. M.: POND - Ein Agentensystem mit Fokus auf Sicherheit und Bezahlung für Leistungen unter Berücksichtigung rechtlicher Aspekte (POND -

An agentsystem with focus on security and payment for services with consideration of legal aspects). Dissertation, Linz, 2002

[STU00] Sturm, J.: Developing XML Solutions. Microsoft Press, Redmond, 2000

[SYC96] Sycara, K., Zeng, D.: Coordination of multiple intelligent software agents. International Journal of Intelligent and Cooperative Information Systems, 5(2), 1996

[TEL98] Tella, S.: The poor relation of the education system? Aspects of distance education and open and distance learning, in: T. Nummi, A. Rönkä, J. Sariola et al., Virtuality and digital nomadism: introduction to the Live Project (1997-2000), "Media education publications" n 6, Department of teacher education, Media education centre, University of Helsinki, 1998, http://www.edu.helsinki.fi/media/mep6/Tella.pdf

[TUR50] Turing, A.M.: Computing machinery and intelligence. Mind, 59, 433-460, 1950

[TUR51] Turing, A.M.: Intelligent machinery, a heretical theory, a lecture given to '51 Society' at Manchester, 1951. Scan of the original Turing typescript available at http://www.turingarchive.org/catalogue.php3?category=AMT/B/4

[VET01] Vetter, M., Pitsch, S.: Towards a Flexible Trading Process over the Internet. in Agent Mediated Electronic Commerce, Dignum Frank, Sierra Carles (Eds.). Springer 2001

[WED73] Wedekind, H.: Der logische und physische Entwurf einer Datenbank. in: Probleme beim Aufbau betrieblicher Informationssysteme, Hrgsg. von H. Hansen, Verlag moderne Industrie, 1973

[WEG72] Wegner, P.: The Vienna Definition Language. ACM Comp Surveys 4(1):5-63, 1972

[WEI00] Weiß, G.: Multiagent Systems. The MIT Press, 2000

[WHI94a] White, J. E.: Telescript Technologie: The Foundation for the Electronic Marketplace, General Magic White Paper, 1994

[WHI94b] White, J. E.: Telescript Technologie: Scenes from the Electronic Marketplace, General Magic White Paper, 1994

[WIE97] Wiemer, W.: Miles – Multimediales Datenbank-Autorensystem für die Physiologie. In S. Hartmut (Hrsg.), Virtueller Campus – Forschung und Entwicklung für neues Lehren und Lernen (S. 179-197). Münster, 1997

[WOO92a] Wooldridge, M.: The Logical Modelling of Computational Multi-Agent Systems. PhD thesis, Department of Computation, UMIST, Manchester, UK, 1992

[WOO92b] Wooldridge, M., Fisher, M.: A first-order branching time logic of multi-agent systems. Proceedings of the Tenth European Conference on Artificial Intelligence (ECAI-92), Vienna, Austria, 1992

[WOO97] Wooldridge, M.: Agent-based Software Engineering. IEEE Proceedings on Software Engineering, 144(1), 1997

[WOO98] Wooldridge, M.: Agents and software engineering. AI*IA Natizie XI, Vol. 3, 1998

[WOO99] Wooldridge M., Jennings, N.: Software Engineering with Agents: Pitfalls and Pratfalls. IEEE Internet Computing 3 (3) 20-27, 1999

[WOO01] Wooldridge, M., Ciancarini, P.: Agent-Oriented Software Engineering: The State of the Art. In P. Ciancarini and M. Wooldridge, editors, Agent-Oriented Software Engineering. Springer-Verlag Lecture Notes in AI Volume 1957, 2001.

[WOO02] Wooldridge, M.: An Introduction to Multiagent Systems. John Wiley and Sons Ltd, February 2002.

[WUR01] Wurmann, P. R.: Dynamic Pricing in the Virtual Marketplace. IEEE Internet Computing, March/April (Vol. 5, Nr. 2), 2001

[ZAM00] Zambonelli, F., Jennings, N. R., Omicini, A. Wooldridge M.: Agent-Oriented Software Engineering for Internet Applications. In A. Omicini, F. Zambonelli,

M. Klusch and R. Tolksdorf, editors, Coordination of Internet Agents, Springer Verlag, 2000

[ZIM97] Zimmer, G.: Mit Multimedia vom Fernunterricht zum Offenen Fernlernen. In L. Issing, P. Klimsa (Hrsg.), Information und Lernen mit Multimedia (2. überarbeitete Auflage) (S. 337-352). Weinheim: Psychologie Verlags Union, 1997

Elektronische Literatur

[AEN] AENEIS - Die umfassende Software zur Unternehmensgestaltung. http://www.atoss.com/produkte/AENEIS/index.html (Stand September 2002)

[AGL] Aglets Software Development Kit Home, http://www.trl.ibm.com/aglets/ (Stand März 2002)

[ASS] Van Assem, M. F. J.: Agent-Oriented Software Engineering: An Evaluation of Current Tools and Theories. http://www.cs.vu.nl/~mark/projects/AOSE-final_version.htm (Stand Februar 2002)

[AUM] Agent UML-Homepage, http://www.auml.org (Stand April 2002)

[BOD] Bodendorf, F.: Office Management Systeme. http://teleteaching.wi2.uni-erlangen.de/mmtt/vod/om980731/sld001.htm (Stand Februar 2002)

[BOT] BotSpot, http://www.botspot.com/ (Stand April 2002)

[BURa] Burger, C.: Vorlesung Kooperation in verteilten Systemen, http://www.informatik.uni-stuttgart.de/ipvr/vs/lehre/ws0001/vorlesungen/KVS/koopver.pdf (Stand Februar 2002)

[BURb] Bursell M., Ugai, T.: Comparison of autonomous mobile agent technologies. http://www.ansa.co.uk/ANSATech/97/Primary/198901.pdf (Stand November 2001)

[CAV] Cavelti, S., et al.: Tools zur Entwicklung von Lernplattformen. http://www.ifi.unizh.ch/groups/se/teaching/ss01/webbased_learning/praesent_2/ToolszurEntwicklungLernp.ppt (Stand Juli 2002)

[CDR] Center for Design Research, Stanford University, http://www-cdr.stanford.edu/CDR/projects.html (Stand April 2002)

[COS] Cossi, G., Dasen, A., Enz, M., Kraner, S., Tan, B.: Didaktisches Design. Universität Zürich http://www.ifi.unizh.ch/se/teaching/ss01/webbased_learning/ergeb/gruppen/didgroup1.htm (Stand Juli 2002)

[CRE] Strauss, H.: What Is a Portal, Anyway? CREN, Tech Talk, http://www.cren.net/know/techtalk/events/portals.html (Stand März 2002)

[DAG] D´Agents, http://agent.cs.dartmouth.edu/general/overview.html (Stand Februar 2002)

[DUB] Dublin Core Metadata Initiative, http://www.dublincore.org/ (August 2002)

[DUE] Fachhochschule Düsseldorf, Fachbereich Medien, Praktikum Java. http://swlab.et.fh-duesseldorf.de/java/PraktikumJAVA/Abschn4.html (Stand Februar 2002)

[EBX] ebXML, http://www.ebxml.org (Stand August 2002)

[END] Ender, C.: „High-Definition Multimedia (Agenten und Avatare), http://www.moondock.org/chrender/Agenten/ Stand Januar 2002

[FIM] WeLearn-Testaccount: http://trywelearn.fim.uni-linz.ac.at/welearn, FIM – Institut für Informationsverarbeitung und Mikroprozessortechnik, J.K. Universität Linz, (Stand August 2002)

[FIP] Foundation for Intelligent Physical Agents, http://www.fipa.org (Stand April 2002)

[GEN] General Magic Inc.: Telescript Language Reference. http://citeseer.nj.nec.com/cache/papers/cs/815/http:zSzzSzcuiwww.unige.chzSzOSGzSzpeoplezSzjvitekzSzResourceszSzArchivezSztslanguageref.pdf/inc95telescript.pdf (Stand Dezember 2001)

[GNU] The GNU project, http://www.gnu.org, 2002

[GOE] Goethe, J. W.: Zauberlehrling. http://www.fln.vcu.edu/goethe/zauber.html (Stand April 2002)

[GRA] Grasshopper 2 – The agent platform, http://www.grasshopper.de/ (Stand März 2002)

[GRU] Gruler, C.: Agentenbasierte Digitale Bibliotheken. http://www.stud.fernuni-hagen.de/q4559630/seminar1914/index.html (Stand Februar 2001)

[HOR] Horn, E., Rose, Th.: Intelligente Softwareagenten und betriebswirtschaftliche Anwendungsszenarien. 2. Kolloquium zum DFG-Schwerpunktprogramm, November

2000, http://www.cs.uni-potsdam.de/soft/veroeffentlichung/publik/ai_005_1.pdf (Stand Februar 2002)

[IEE] IEEE Learning Technology Standards Committee (LTSC), Learning Object Metadata Working Group, Draft Standard for Learning Object Metadata: http://ltsc.ieee.org/doc/wg12/LOM_1484_12_1_v1_Final_Draft.pdf (August 2002)

[IMSa] IMS Content Packaging Information Model, Final Specification Version 1.0, IMS Global Learning Consortium, Inc. http://www.imsproject.com, (Stand Juli 2002)

[IMSb] IMS Meta-Data, http://www.imsproject.org/metadata/index.cfm, (Stand August 2002)

[JAV] Java-Homepage, http://java.sun.com (Stand Dezember 2001)

[JEN] Jennings, N.: Negotiating Agents for Business Process Management http://www.ecs.soton.ac.uk/~nrj/download-files/adept.pdf (Stand Feburar 2002)

[MAG] Magedanz, T.: Grasshopper - A Universal Agent Platform Based on OMG MASIF and FIPA Standards, http://www.cordis.lu/infowin/acts/analysys/products/thematic/agents/ch4/ch4.htm (Stand März 2002)

[MAS] MAS Laboratory, Department of Computer Science, University of Massachusetts at Amherst, Multi Agent Survivability Simulator (MASS). http://dis.cs.umass.edu/research/mass/api/utilities/utilities/KQML.html (Stand Februar 2002)

[MEI] Meier, M., et al: Fachliche & soziale Betreuung, Motivation und Feedback. http://www.ifi.unizh.ch/se/teaching/ss01/webbased_learning/ergeb/gruppen/distgroup1.htm, Universität Zürich (Stand Juli 2002)

[MIT] MIT Media Lab, Software Agents Group, http://www.hivecell.net/ (Stand März 2002)

[MOB] Mobile Agenten - Security und Anwendungen. Technische Hochschule Darmstadt, Fachbereich Informatik, http://www.ito.tu-darmstadt.de/edu/sem-kivs-s97/aus4-html/agent02.html, Stand Januar 2001

[OMGa] Object Management Group, http://www.omg.org, (Stand April 2002)

[OMGb] Object Management Group, UML Resource Page. http://www.omg.org/uml/ (Stand April 2002)

[PON] POND: The agent system POND. http://www.fim.uni-linz.ac.at/research/telework/agents.htm (Stand August 2002)

[POR] Portals from the Higher Education Perspective, A Short Introduction, http://ict.emich.edu/intro.htm (Stand März 2002)

[POS] Poslad, S., Buckle, P., Hadingham, R.: The FIPA-OS Agent Platform:Open Source for Open Standards. Manchester, http://fipa-os.sourceforge.net/docs/papers/FIPAOS.pdf (Stand August 2002).

[POT] Fachhochschule Potsdam, XML-Einführungskurs, Transformationen: http://www.informatik.hu-berlin.de/~obecker/Lehre/XML-Kurs/xml4.html (Stand November 2001)

[SAX] Saxproject: Simple API for XML, http://www.saxproject.org/ (Stand August 2002)

[SCH] XML Schema Language, http://www.fh-wedel.de/~si/seminare/ws00/Ausarbeitung/2.schema/abgabe_01.htm (Stand Februar 2002)

[SIM] Shoppingagent mySimon, http://www.mysimon.com (Stand März 2002)

[SOF] Software Agents group of the MIT Media Laboratory, http://agents.media.mit.edu/projects/ (Stand April 2002)

[SOM] Sommer, M.: Gemeinsame Homepages und Portale: Möglichkeiten und Voraussetzungen, http://www.uni-marburg.de/archivschule/sommer.pdf (Stand März 2002)

[SUNb] Java Technologie and XML. http://java.sun.com/xml/ (Stand März 2002)

[TCLa] Tcl Developer Xchange, Tcl/Tk Documentation. http://tcl.activestate.com/doc/ (Stand Mai 2002)

[TCLb] Tcl Developer Xchange: Safe-Tcl. tcl.activestate.com/software/plugin/safetcl.html (Stand Mai 2002)

[TSC] Tschiemer, S. Svoboda, M. Schleinofer, M., Schaller, F.: Web-basiertes Lernen: eLarning und Wissensmanagement. Universität Zürich, http://www.ifi.unizh.ch/se/teaching/ss01/webbased_learning/ergeb/gruppen/wissgroup1/start.htm (Stand Juli 2002)

[TVE] Tveit, A.: A Survey of Agent-Oriented Engineering. http://www.jfipa.org/publications/AgentOrientedSoftwareEngineering/ (Stand Februar 2002)

[UNI] unicmind.com AG: eLearning und Wissensmanagement in deutschen Grossunternehmen. http://www.unicmind.com/elearningstudie.pdf (Stand Juli 2002)

[W3Ca] World Wide Web Consortium (W3C), XML: http://www.w3.org/XML/ (Stand Mai 2002)

[W3Cb] Extensible Markup Language (XML) 1.0, W3C Recommendation 10-February-1998, http://www.w3.org/TR/1998/REC-xml-19980210 (Stand November 2001)

[W3Cc] World Wide Web Consotrtium: Document Object Model (DOM), http://www.w3.org/DOM/ (Stand August 2002)

[WEB] Wepopedia, http://webopedia.internet.com/TERM/v/vortal.html (Stand Mai 2002)

[XML] XML Standard, http://www.xml.com (Stand November 2001)

Anhang A: Sourcecode Agenten (Auszug)

PortalServerAgent.java

```
/*
 * @(#) PortalServerAgent.java
 *
 * PortalServerAgent
 * Author: Susanne Reisinger
 * Developed at the Institute for Information Processing and Microprocessor
Technology (FIM)
 * Johannes Kepler University Linz
 * Altenbergerstr. 69, A-4040 Linz, Austria
 */

package Portal;

import java.util.*;
import java.io.*;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import PkgAgentSystem.*;
import PkgAgentSystem.Crypto.*;
import PkgAgentSystem.GUI.*;
import PkgAgentSystem.Messaging.Message;
import PkgAgentSystem.Messaging.BroadcastMessage;
import PkgAgentSystem.Messaging.MessageException;
import FIM.Util.Crypto.NamedKeyAndCertificate;
import Protocol.Inquiry.IdentifiedInquiryProtocol;
import Protocol.Contact.ContactProtocol;
import Protocol.Identification.IdentificationProtocol;
import Protocol.Identification.SignData.SignDataIdentificationProtocol;
import Protocol.Identification.Passphrase.PassphraseIdentificationProtocol;
import Protocol.Identification.Passphrase.WordpairChallengeResponse;
import Protocol.Identification.SendChallengeMessage;
import Protocol.ebXMLAgent;
import Protocol.ebXMLMessage;
import Protocol.ebXMLBroadcastMessage;
import Protocol.MessageParsingHandler;
import Protocol.Sell.*;
import java.security.PrivateKey;
import java.sql.*;
import FIM.payment.*;
import org.apache.xalan.serialize.Serializer;
import org.apache.xalan.serialize.SerializerFactory;
import org.apache.xalan.templates.OutputProperties;
import javax.xml.messaging.JAXMException;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.DocumentBuilder;
```

```
import javax.xml.parsers.ParserConfigurationException;
import org.xml.sax.SAXException;
import org.xml.sax.SAXParseException;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.Node;
import org.w3c.dom.Text;
import MetaDataAgents.CourseMDAgent.*;

/**
@version 1.0
*/
public class PortalServerAgent extends ebXMLAgent
{
    private IdentifiedInquiryProtocol inq=null;
    private PassphraseIdentificationProtocol ppConv=null;
    private SignDataIdentificationProtocol signConv=null;
    private GetMemberDataProtocol getMemberDataProtocolConv=null;
    private GetPaymentInfoProtocol getPaymentInfoProtocolConv=null;
    private DoFeePaymentProtocol doFeePaymentProtocol=null;

    public PortalServerAgent()
    {
        ppConv=new PassphraseIdentificationProtocol(this);
              ppConv.addPassPhrase("The night?","One's best friend");
            ppConv.addPassPhrase("The stars","Like dust");
        registerConversation(ppConv);
        signConv=new SignDataIdentificationProtocol(this);
        registerConversation(signConv);
        registerConversation(new GetMemberDataProtocol(this));
        registerConversation(new GetPaymentInfoProtocol(this));
        registerConversation(new FindAgentBroadcastProtocol(this));

    }

    private void registerSellConversation()
    {
        // Sell protocol configuration
        IdentificationProtocol[] idProts=new IdentificationProtocol[3];
        idProts[0]=signConv;
        idProts[1]=ppConv;
        idProts[2]=new Protocol.Identification.None.NoIdentificationProtocol(this);
        DeliveryProtocol[] delProts=new DeliveryProtocol[1];
        delProts[0]=new XMLDocDeliveryProtocol(this);
        PaymentProtocol[] payProts=new PaymentProtocol[2];
        Calendar cal=Calendar.getInstance();
        cal.setTime(new java.util.Date());
        cal.add(Calendar.YEAR,1);
        byte[] number={4,0,0,0,0,8,1,5,4,7,1,1,0,3,3,3,3};
        payProts[0]=new CreditCardPaymentProtocol(this,"ACME-Card","Jane Doe",
        number,cal.getTime());
        VoucherWrapper[] vouchers=new VoucherWrapper[1];
```

```
    vouchers[0]=new VoucherWrapper("Special Discount",null,new
    Price(Currency.findCurrency("EUR"),1.0));
    payProts[1]=new VoucherPaymentProtocol(this,vouchers);
    doFeePaymentProtocol=new DoFeePaymentProtocol(this);
    FeeSellProtocol sellProt=new FeeSellProtocol(this,idProts,
    doFeePaymentProtocol,delProts,payProts,new NamedKeyAndCertificate("Sell-
    Protocol",getKeyPair("Agents own keys").getKeyPair().getPrivate(),
    getIdentity()));
    registerConversation(sellProt);
}

  protected void showDialog() {}

  protected ResultSet getSingleRow(String query) throws SQLException
  {
      if(query==null || query.length()<1)
          throw new IllegalArgumentException("Query must be provided");
        try
        {
           Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
        }
        catch(java.lang.ClassNotFoundException e)
        {
           throw new IllegalStateException("Could not load database driver");
        }
          Connection con=DriverManager.getConnection
        ("jdbc:odbc:ParticipantsData", "","");
              Statement stmt=con.createStatement();
        ResultSet rs=stmt.executeQuery(query);
    rs.next();
        return rs;
  }

  protected ResultSet getMultipleRows(String query) throws SQLException
  {
      if(query==null || query.length()<1)
          throw new IllegalArgumentException("Query must be provided");
        try
        {
           Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
        }
        catch(java.lang.ClassNotFoundException e)
        {
           throw new IllegalStateException("Could not load database driver");
        }
                Connection
        con=DriverManager.getConnection("jdbc:odbc:ParticipantsData","","");
              Statement stmt=con.createStatement();
        ResultSet rs=stmt.executeQuery(query);
    return rs;
  }
```

```
protected int updateDatabase(String query) throws SQLException
{
    if(query==null || query.length()<1)
        throw new IllegalArgumentException("Query must be provided");
      try
      {
          Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
      }
      catch(java.lang.ClassNotFoundException e)
      {
          throw new IllegalStateException("Could not load database driver");
      }
              Connection
      con=DriverManager.getConnection("jdbc:odbc:ParticipantsData","","");
            Statement stmt=con.createStatement();

      int nrOfUpdatedRows=stmt.executeUpdate(query);
  return nrOfUpdatedRows;
}

protected org.w3c.dom.Document createXMLDocument(String name) throws
   IllegalStateException
{
    try
    {
     String str="<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n<!DOCTYPE
     "+name+"Doc SYSTEM \"file:/"+name+".dtd\">\n<"+name+"Doc/>";
     DocumentBuilderFactory dFactory=DocumentBuilderFactory.newInstance();
     DocumentBuilder dBuilder=dFactory.newDocumentBuilder();
     MessageParsingHandler handler=new MessageParsingHandler();
     dBuilder.setEntityResolver(handler);
      org.w3c.dom.Document doc=dBuilder.parse(new
     java.io.ByteArrayInputStream(str.getBytes()));
     return doc;
  }
  catch(SAXException t)
  {
      if(t instanceof SAXParseException)
      {
          SAXParseException sp=(SAXParseException)t;
      }
      throw new IllegalStateException("Error creating XML document (SAX):
      "+t.toString());
  }
  catch(ParserConfigurationException t)
  {   // Should not happen
      throw new IllegalStateException("Error creating XML document (Parser
      config): "+t.toString());
  }
  catch(IOException t)
  {   // Should not happen
      throw new IllegalStateException("Error creating XML document (IO):
      "+t.toString());
```

```
  }
}

private void insertMemberData(ResultSet rs,Node
parentNode,org.w3c.dom.Document doc) throws SQLException
{
 Element el=doc.createElement("MemberData");
 el.setAttribute("MemberID",rs.getString("MemberID"));
 el.setAttribute("Gender",rs.getString("Gender"));
 parentNode.appendChild(el);
 Element nameNode=doc.createElement("Name");
 el.appendChild(nameNode);
 String datastr=rs.getString("FName");
 if(datastr!=null)
 {
     Element node=doc.createElement("FName");
     node.appendChild(doc.createTextNode(datastr));
     nameNode.appendChild(node);
 }
 datastr=rs.getString("SName");
 if(datastr!=null)
 {
     Element node=doc.createElement("SName");
     node.appendChild(doc.createTextNode(datastr));
     nameNode.appendChild(node);
 }
 datastr=rs.getString("MInitial");
 if(datastr!=null)
 {
     Element node=doc.createElement("MInitial");
     node.appendChild(doc.createTextNode(datastr));
     nameNode.appendChild(node);
 }
 datastr=rs.getString("Title");
 if(datastr!=null)
 {
     Element node=doc.createElement("Title");
     node.appendChild(doc.createTextNode(datastr));
     nameNode.appendChild(node);
 }
 java.sql.Date d=rs.getDate("BDate");
 if(d!=null)
 {
     Element node=doc.createElement("BDate");
     node.appendChild(doc.createTextNode(d.toString()));
     el.appendChild(node);
 }
 Element adrNode=doc.createElement("Address");
 el.appendChild(adrNode);
 datastr=rs.getString("Street");
 if(datastr!=null)
 {
```

```
        Element node=doc.createElement("Street");
        node.appendChild(doc.createTextNode(datastr));
        adrNode.appendChild(node);
    }
    datastr=rs.getString("PCode");
    if(datastr!=null)
    {
        Element node=doc.createElement("PCode");
        node.appendChild(doc.createTextNode(datastr));
        adrNode.appendChild(node);
    }
    datastr=rs.getString("City");
    if(datastr!=null)
    {
        Element node=doc.createElement("City");
        node.appendChild(doc.createTextNode(datastr));
        adrNode.appendChild(node);
    }
    datastr=rs.getString("Country");
    if(datastr!=null)
    {
        Element node=doc.createElement("Country");
        node.appendChild(doc.createTextNode(datastr));
        adrNode.appendChild(node);
    }
    datastr=rs.getString("EMail");
    if(datastr!=null)
    {
        Element node=doc.createElement("EMail");
        node.appendChild(doc.createTextNode(datastr));
        adrNode.appendChild(node);
    }
    datastr=rs.getString("TelNr");
    if(datastr!=null)
    {
        Element node=doc.createElement("TelNr");
        node.appendChild(doc.createTextNode(datastr));
        adrNode.appendChild(node);
    }
  }

public void insertCourseEventFeePaymentInfo(ResultSet rs,Node
parentNode,org.w3c.dom.Document doc) throws SQLException
  {
    Element el=doc.createElement("CourseEventFeePaymentInfo");
    el.setAttribute("EventID",rs.getString("EventID"));
    el.setAttribute("MemberID",rs.getString("MemberID"));
    parentNode.appendChild(el);
    Element feeNode=doc.createElement("EventFee");
    String paid=rs.getString("Event_FeePaid");
    if(paid!=null && paid.compareTo("0")!=0)
       feeNode.setAttribute("Event_FeePaid","Yes");
```

```
    else
       feeNode.setAttribute("Event_FeePaid","No");
    el.appendChild(feeNode);
    java.sql.Date d=rs.getDate("EventFee_Date");
    if(d!=null)
    {
        Element node=doc.createElement("EventFee_Date");
        node.appendChild(doc.createTextNode(d.toString()));
        feeNode.appendChild(node);
    }
    String datastr=rs.getString("EventFee_Amount");
    if(datastr!=null)
    {
        Element node=doc.createElement("EventFee_Amount");
        node.appendChild(doc.createTextNode(datastr));
        feeNode.appendChild(node);
    }
    datastr=rs.getString("EventFee_Currency");
    if(datastr!=null)
    {
        Element node=doc.createElement("EventFee_Currency");
        node.appendChild(doc.createTextNode(datastr));
        feeNode.appendChild(node);
    }
    d=rs.getDate("BEventDate");
    if(d!=null)
    {
        Element node=doc.createElement("BEventDate");
        node.appendChild(doc.createTextNode(d.toString()));
        el.appendChild(node);
    }
    d=rs.getDate("EEventDate");
    if(d!=null)
    {
        Element node=doc.createElement("EEventDate");
        node.appendChild(doc.createTextNode(d.toString()));
        el.appendChild(node);
    }
    datastr=rs.getString("ETitle");
    if(datastr!=null)
    {
        Element node=doc.createElement("ETitle");
        node.appendChild(doc.createTextNode(datastr));
        el.appendChild(node);
    }
  }

public void insertCoursePaymentInfo(ResultSet rs,Node
parentNode,org.w3c.dom.Document doc) throws SQLException
  {
    Element el=doc.createElement("CoursePaymentInfo");
    el.setAttribute("MemberID",rs.getString("MemberID"));
```

```
 el.setAttribute("CourseID",rs.getString("CourseID"));
 parentNode.appendChild(el);
 Element feeNode=doc.createElement("Fee");
 String paid=rs.getString("FeePaid");
 if(paid!=null && paid.compareTo("0")!=0)
    feeNode.setAttribute("FeePaid","Yes");
 else
    feeNode.setAttribute("FeePaid","No");
 el.appendChild(feeNode);
 java.sql.Date d=rs.getDate("Date");
 if(d!=null)
 {
     Element node=doc.createElement("Date");
     node.appendChild(doc.createTextNode(d.toString()));
     feeNode.appendChild(node);
 }
 String datastr=rs.getString("Amount");
 if(datastr!=null)
 {
     Element node=doc.createElement("Amount");
     node.appendChild(doc.createTextNode(datastr));
     feeNode.appendChild(node);
 }
 datastr=rs.getString("Currency");
 if(datastr!=null)
 {
     Element node=doc.createElement("Currency");
     node.appendChild(doc.createTextNode(datastr));
     feeNode.appendChild(node);
 }
}

public void insertMemberCourseIDs(String memberID, String courseID, Node
parentNode,org.w3c.dom.Document doc) throws SQLException
{
 Element el=doc.createElement("CoursePaymentInfo");
 el.setAttribute("MemberID",memberID);
 el.setAttribute("CourseID",courseID);
 parentNode.appendChild(el);
}

/**
@param id the id of the member to find
@return an XML document with the data
@throws IllegalArgumentException if the parameter id is erroneous or not
found
@throws IllegalStateException if the database driver could not be loaded or
an XML exception occured
@throws SQLException if an SQL exception occured
*/
public org.w3c.dom.Document getMemberData(String id) throws
IllegalArgumentException,IllegalStateException, SQLException
```

```
{
    if(id==null || id.length()<1)
        throw new IllegalArgumentException("id must be provided");
      ResultSet rs=getSingleRow("SELECT * FROM MemberData WHERE
      MemberID="+id+";");
      org.w3c.dom.Document doc=createXMLDocument("MemberData");
 // Insert data
 insertMemberData(rs,doc.getDocumentElement(),doc);
 // Print the resulting XML document
 ByteArrayOutputStream bos=new ByteArrayOutputStream();
 try
 {
     OutputProperties op=new
     OutputProperties(OutputProperties.getDefaultMethodProperties("xml"));

     op.setProperty(javax.xml.transform.OutputKeys.DOCTYPE_SYSTEM,"MemberData
     .dtd");
     Serializer
     serializer=SerializerFactory.getSerializer(op.getProperties());
     serializer.setOutputStream(bos);
     serializer.asDOMSerializer().serialize(doc);
 }
 catch(IOException e)
 {
     throw new IllegalStateException(e.toString());
 }
 return doc;
}
/**
@param id the id of the member and the course to find
@return an XML document with the data
@throws IllegalArgumentException if the parameter id is erroneous or not
found
@throws IllegalStateException if the database driver could not be loaded or
an XML exception occured
@throws SQLException if an SQL exception occured
*/
public org.w3c.dom.Document getPaymentInfo(String memberID, String courseID)
throws IllegalArgumentException,IllegalStateException, SQLException
{
    if(memberID==null || memberID.length()<1 || courseID==null ||
courseID.length()<1)
        throw new IllegalArgumentException("id must be provided");
      org.w3c.dom.Document doc =createXMLDocument("PaymentInfo");
ResultSet rs= getMultipleRows("SELECT * FROM CourseMemberFee WHERE
FeePaid=False  AND MemberID="+memberID +" AND CourseID="+courseID+";");
while (rs.next()) {
    insertCoursePaymentInfo(rs, doc.getDocumentElement(),doc);
}
      rs=getMultipleRows("SELECT DISTINCTROW
EventParticipantsMember.MemberID AS MemberID, EventParticipantsMember.FeePaid
AS Event_FeePaid, EventData.EventID AS EventID, EventData.BEventDate AS
BEventDate, EventData.EEventDate AS EEventDate, EventData.EventFee_Amount AS
EventFee_Amount, EventData.EventFee_Currency AS EventFee_Currency,
EventData.EventFee_Date AS EventFee_Date, EventData.ETitle AS ETitle"+" FROM
```

```
CourseEventData INNER JOIN (EventData LEFT JOIN EventParticipantsMember ON
EventData.EventID = EventParticipantsMember.EventID) ON
CourseEventData.EventID = EventData.EventID"+" WHERE
((EventParticipantsMember.FeePaid=False) AND
(EventParticipantsMember.EventID=CourseEventData.EventID) AND
(CourseEventData.CourseID="+courseID+") AND
(EventParticipantsMember.MemberID="+memberID+"));");
  // Insert data
  while (rs.next()) {
      insertCourseEventFeePaymentInfo(rs,doc.getDocumentElement(),doc);
  }
  // Print the resulting XML document
  ByteArrayOutputStream bos=new ByteArrayOutputStream();
  try
  {
      OutputProperties op=new
OutputProperties(OutputProperties.getDefaultMethodProperties("xml"));

op.setProperty(javax.xml.transform.OutputKeys.DOCTYPE_SYSTEM,"PaymentInfo.dtd
");
      Serializer
      serializer=SerializerFactory.getSerializer(op.getProperties());
      serializer.setOutputStream(bos);
      serializer.asDOMSerializer().serialize(doc);
  }
  catch(IOException e)
  {
      throw new IllegalStateException(e.toString());
  }
  return doc;
}

public org.w3c.dom.Document getPrice(String memberID, String courseID,
String[] eventIDs, int nrOfEvents, Price price, String currency) throws
IllegalArgumentException,IllegalStateException, SQLException
{
  ResultSet rs = null;
  double fprice = 0;

     if ((!courseID.equals("")) && (!memberID.equals("")) && (courseID!=null))
{
        rs = getSingleRow("SELECT SUM(CourseMemberFee.Amount) AS Expr1 FROM
        CourseMemberFee WHERE CourseMemberFee.MemberID="+memberID+" AND
        CourseMemberFee.CourseID="+courseID+" AND
        CourseMemberFee.FeePaid=False;");
      fprice = Double.parseDouble(rs.getString("Expr1"));
  }
  if (nrOfEvents > 0) {
      for (int i = 0; i < nrOfEvents; i++) {
          rs = getSingleRow("SELECT SUM(EventData.EventFee_Amount) AS Expr1
        FROM EventData LEFT JOIN EventParticipantsMember ON EventData.EventID
        = EventParticipantsMember.EventID WHERE
        EventParticipantsMember.EventID="+eventIDs[i]+" AND
        EventParticipantsMember.MemberID="+memberID+" AND
        EventParticipantsMember.FeePaid=False;");
          fprice += Double.parseDouble(rs.getString("Expr1"));
```

```
        }
    }
    Price feePrice = new Price(Currency.findCurrency(currency), fprice);
    String priceOK = "No";
    if (feePrice.equals(price)) {
        priceOK = "Yes";
    }
    org.w3c.dom.Document doc =createXMLDocument("Price");
    Element node=doc.createElement("PriceOK");
    node.appendChild(doc.createTextNode(priceOK));
    doc.getDocumentElement().appendChild(node);


    // Print the resulting XML document
    ByteArrayOutputStream bos=new ByteArrayOutputStream();
    try
    {
        OutputProperties op=new
        OutputProperties(OutputProperties.getDefaultMethodProperties("xml"));

        op.setProperty(javax.xml.transform.OutputKeys.DOCTYPE_SYSTEM,"Price.dtd"
        );
        Serializer
        serializer=SerializerFactory.getSerializer(op.getProperties());
        serializer.setOutputStream(bos);
        serializer.asDOMSerializer().serialize(doc);
    }
    catch(IOException e)
    {
        throw new IllegalStateException(e.toString());
    }
    return doc;
}

/**
@param id of the course and the events to pay and the nrOfEvents
@return an XML document with the data
@throws IllegalArgumentException if the parameter id is erroneous or not
found
@throws IllegalStateException if the database driver could not be loaded or
an XML exception occured
@throws SQLException if an SQL exception occured
*/
public org.w3c.dom.Document doFeePayment(String memberID, String courseID,
String[] eventIDs, int nrOfEvents) throws
IllegalArgumentException,IllegalStateException, SQLException
{
    int nrOfUpdatedRows = 0;
       if ((!courseID.equals("")) && (!memberID.equals("")) {
          nrOfUpdatedRows = updateDatabase("UPDATE CourseMemberFee SET
          CourseMemberFee.FeePaid = True WHERE FeePaid=False AND
          MemberID="+memberID+" AND CourseID="+courseID+";");
    }
    if (nrOfEvents > 0) {
```

```
        for (int i = 0; i < nrOfEvents; i++) {
             nrOfUpdatedRows += updateDatabase("UPDATE EventParticipantsMember
         SET EventParticipantsMember.FeePaid = True WHERE FeePaid=False  AND
         EventParticipantsMember.EventID="+eventIDs[i]+" AND
         EventParticipantsMember.MemberID="+memberID+";");
         }
    }
         org.w3c.dom.Document doc =createXMLDocument("FeePayment");
    insertFeesPaid(nrOfUpdatedRows, doc.getDocumentElement(),doc);
    // Print the resulting XML document
    ByteArrayOutputStream bos=new ByteArrayOutputStream();
    try
    {
        OutputProperties op=new
         OutputProperties(OutputProperties.getDefaultMethodProperties("xml"));

         op.setProperty(javax.xml.transform.OutputKeys.DOCTYPE_SYSTEM,"FeePayme
         nt.dtd");
        Serializer
         serializer=SerializerFactory.getSerializer(op.getProperties());
        serializer.setOutputStream(bos);
        serializer.asDOMSerializer().serialize(doc);
    }
    catch(IOException e)
    {
        throw new IllegalStateException(e.toString());
    }
    return doc;
  }

  private void insertFeesPaid (int nrOfUpdatedRows, Node
  parentNode,org.w3c.dom.Document doc) throws SQLException
  {
    Element el=doc.createElement("FeePayment");
    if (nrOfUpdatedRows > 0)
        el.setAttribute("FeesPaid","Yes");
    else
        el.setAttribute("FeesPaid","No");
    parentNode.appendChild(el);
    String datastr = ""+nrOfUpdatedRows;
    if (datastr!=null) {
        Element node=doc.createElement("NrOfPaidFees");
        node.appendChild(doc.createTextNode(datastr));
        el.appendChild(node);
    }
}

  private void insertPortalMemberData(ResultSet rs,Node
  parentNode,org.w3c.dom.Document doc,String memberID) throws SQLException
  {
    Element el=doc.createElement("PortalMemberData");
    el.setAttribute("MemberID",memberID);
    parentNode.appendChild(el);
    String datastr=rs.getString("Amount");
```

```
 if(datastr!=null)
 {
     Element feeNode=doc.createElement("Fee");
     Element node=doc.createElement("Amount");
     node.appendChild(doc.createTextNode(datastr));
     feeNode.appendChild(node);
     node=doc.createElement("Currency");
     node.appendChild(doc.createTextNode(rs.getString("Currency")));
     feeNode.appendChild(node);
     String paid=rs.getString("FeePaid");
     if(paid!=null && paid.compareTo("0")!=0)
         feeNode.setAttribute("FeePaid","Yes");
     else
         feeNode.setAttribute("FeePaid","No");
     el.appendChild(feeNode);
 }
 datastr=rs.getString("DEntrance");
 if(datastr!=null)
 {
     Element node=doc.createElement("DEntrance");
     node.appendChild(doc.createTextNode(datastr));
     el.appendChild(node);
 }
 datastr=rs.getString("DWithdrawal");
 if(datastr!=null)
 {
     Element node=doc.createElement("DWithdrawal");
     node.appendChild(doc.createTextNode(datastr));
     el.appendChild(node);
 }
 datastr=rs.getString("UserID");
 if(datastr!=null)
 {
     Element node=doc.createElement("UserID");
     node.appendChild(doc.createTextNode(datastr));
     el.appendChild(node);
 }
 datastr=rs.getString("Password");
 if(datastr!=null)
 {
     Element node=doc.createElement("Password");
     node.appendChild(doc.createTextNode(datastr));
     el.appendChild(node);
 }
}

protected org.w3c.dom.Document getPortalMemberData(String id) throws
IllegalArgumentException,IllegalStateException, SQLException
{
    if(id==null || id.length()<1)
        throw new IllegalArgumentException("id must be provided");
      org.w3c.dom.Document doc=createXMLDocument("PortalMemberData");
```

```
        ResultSet rs=getSingleRow("SELECT PortalMemberData.*,
        PortalMemberFee.* FROM PortalMemberData LEFT JOIN PortalMemberFee ON
        PortalMemberData.MemberID=PortalMemberFee.MemberID WHERE
        PortalMemberData.MemberID="+id+";");
      // Insert data
      insertPortalMemberData(rs,doc.getDocumentElement(),doc,id);
            rs=getSingleRow("SELECT * FROM MemberData WHERE
        MemberID="+id+";");
      insertMemberData(rs,doc.getDocumentElement(),doc);
      // Print the resulting XML document
      ByteArrayOutputStream bos=new ByteArrayOutputStream();
      try
      {
          OutputProperties op=new
        OutputProperties(OutputProperties.getDefaultMethodProperties("xml"));

        op.setProperty(javax.xml.transform.OutputKeys.DOCTYPE_SYSTEM,"PortalMe
        mberData.dtd");
          Serializer
        serializer=SerializerFactory.getSerializer(op.getProperties());
       serializer.setOutputStream(bos);
       serializer.asDOMSerializer().serialize(doc);
    }
    catch(IOException e)
    {
       throw new IllegalStateException(e.toString());
    }
    return doc;
   }

protected void initialize(Object param) throws InitializationException {
    signConv.setPrivateKey(getKeyPair("Agents own
keys").getKeyPair().getPrivate());
    registerSellConversation();
}

protected javax.swing.JPanel createVisualization()
{
    javax.swing.JPanel visualization=new javax.swing.JPanel();
    visualization.setLayout(new BorderLayout());
    visualization.setBorder(new javax.swing.border.EtchedBorder());
    javax.swing.JPanel pane=new javax.swing.JPanel();
    pane.setLayout(new GridBagLayout());
    pane.setBackground(Color.white);
    pane.setOpaque(true);
    javax.swing.JLabel label=new javax.swing.JLabel("Portal Server Agent");
    GridBagConstraints cons=new GridBagConstraints();
    cons.insets=new Insets(10,10,10,10);
    cons.fill=GridBagConstraints.HORIZONTAL;
    cons.anchor=GridBagConstraints.CENTER;
    cons.weightx=1.0;
    cons.weighty=0.0;
    cons.gridx=0;
    cons.gridy=0;
```

```
        pane.add(label,cons);
        cons.insets=new Insets(10,10,10,10);
        cons.fill=GridBagConstraints.NONE;
        cons.anchor=GridBagConstraints.CENTER;
        cons.ipadx=10;
        cons.ipady=10;
        cons.weightx=1.0;
        cons.weighty=0.0;
        cons.gridx=0;
        cons.gridy=1;
        visualization.add(pane);
        return visualization;
    }
}

class PortalTestSystem extends GUIAgentSystem
{
    public PortalTestSystem(int port,String library_dir,String homedir,String
     systemDir,boolean exitOnLast,String savedFile,boolean useSecurity)
    {
        super(port,library_dir,homedir,systemDir,exitOnLast,savedFile,useSecurity);
    }

    public static void main(String[] args)
    {
        PortalServerAgent ag=new PortalServerAgent();
        try
        {
            ag.getPaymentInfo("1", "1");
        }
        catch(Exception e)
        {
            e.printStackTrace();
            System.out.println(e.toString());
        }
        System.exit(0);
    }
}
```

CourseMDAgent

```
/*
 * @(#) CourseMDAgent.java
 *
 * CourseMDAgent
 * Author: Susanne Reisinger
 * Developed at FIM
 * Johannes Kepler University Linz
 * Altenbergerstr. 69, A-4040 Linz, Austria
 */

package MetaDataAgents.CourseMDAgent;
import java.util.*;
import java.io.*;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.net.URL;
import PkgAgentSystem.*;
import PkgAgentSystem.GUI.*;
import PkgAgentSystem.Messaging.Message;
import PkgAgentSystem.Messaging.SignedMessage;
import PkgAgentSystem.Crypto.AgentIdentity;
import PkgAgentSystem.Crypto.AgentIdentityExtension;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.NodeList;
import org.w3c.dom.Node;
import org.w3c.dom.CharacterData;
import org.w3c.dom.Text;
import Protocol.*;
import Protocol.Sell.*;
import Protocol.Identification.SignData.SignDataIdentificationProtocol;
import Protocol.Identification.Passphrase.PassphraseIdentificationProtocol;
import Protocol.Identification.IdentificationProtocol;
import FIM.payment.*;
import FIM.Util.Crypto.NamedKeyAndCertificate;
import org.apache.xalan.serialize.Serializer;
import org.apache.xalan.serialize.SerializerFactory;
import org.apache.xalan.templates.OutputProperties;

public class CourseMDAgent extends ebXMLAgent implements ActionListener
{
    transient private MemberDataDlg memberDataDlg=null;
    transient private ChooseFunctionDlg chooseFuncDlg=null;
    protected URL moveToURL = null;
    protected URL startFromURL = null;
    protected int moveState = 0;
    protected String memberID = null;
    protected String courseID = null;
```

```
protected MemberData data = null;
protected PaymentInfo paymentData = null;
protected String feePaid = "No";
protected int nrOfPaidFees = 0;
protected GetMemberDataProtocol getMemberDataProtocol=null;
protected GetPaymentInfoProtocol getPaymentInfoProtocol=null;
protected SellProtocol doSellProtocol=null;
protected DoFeePaymentProtocol doFeePaymentProtocol=null;
protected PassphraseIdentificationProtocol ppConv = null;
protected SignDataIdentificationProtocol signConv = null;
protected boolean isBroadcasted=false;
protected int state=0;

public CourseMDAgent() {
    registerConversation(new GetMemberDataProtocol(this));
    registerConversation(new GetPaymentInfoProtocol(this));
    registerConversation(new FindAgentBroadcastProtocol(this));
    signConv = new SignDataIdentificationProtocol(this);
    registerConversation(signConv);
    ppConv=new PassphraseIdentificationProtocol(this);
          ppConv.addPassPhrase("The night?","One's best friend");
        ppConv.addPassPhrase("The stars","Like dust");
    registerConversation(ppConv);
    try
    {
        moveToURL = new URL("amp://susi_wxp:4712");
    }
    catch(java.net.MalformedURLException e)
    {
        System.err.println(e.toString());
    }
}

protected void registerSellConversation(String cardCompany,String
nameOnCard,byte[] number,Date expiry)
{
    // Sell protocol configuration
    IdentificationProtocol[] idProts=new IdentificationProtocol[2];
    idProts[0]=signConv;
    idProts[1]=ppConv;
    DeliveryProtocol[] delProts=new DeliveryProtocol[1];
    delProts[0]=new XMLDocDeliveryProtocol(this);
    PaymentProtocol[] payProts=new PaymentProtocol[2];
    payProts[0]=new
    CreditCardPaymentProtocol(this,cardCompany,nameOnCard,number,expiry);
    VoucherWrapper[] vouchers=new VoucherWrapper[1];
    vouchers[0]=new VoucherWrapper("Special Discount",null,new
    Price(Currency.findCurrency("EUR"),1.0));
    payProts[1]=new VoucherPaymentProtocol(this,vouchers);
    doFeePaymentProtocol=new DoFeePaymentProtocol(this);
    FeeSellProtocol sellProt=new FeeSellProtocol(this,idProts,
    doFeePaymentProtocol,delProts,payProts,new NamedKeyAndCertificate("Sell-
```

```
    Protocol",getKeyPair("Agents own
    keys").getKeyPair().getPrivate(),getIdentity()));
    registerReplaceConversation(sellProt);
}

protected void initialize(Object param) throws InitializationException
{
    signConv.setPrivateKey(getKeyPair("Agents own
    keys").getKeyPair().getPrivate());
}

protected void onBeforeMoving() {
    if (memberDataDlg!=null) {
        memberDataDlg.setVisible(false);
        memberDataDlg=null;
    }
    if (chooseFuncDlg!=null) {
        chooseFuncDlg.setVisible(false);
        chooseFuncDlg=null;
    }
}

protected void goServer() {
    try {
        startFromURL = getAgentSystem().getLocation();
        move(moveToURL);

    }
    catch (ReceiveAgentDeniedException e){
        System.err.println(e.toString());
    }

    catch (CannotSendException e) {
        System.err.println(e.toString());
    }
}
protected void goBack() {
    try {
        move(startFromURL);
    }
    catch (ReceiveAgentDeniedException e){
        System.err.println(e.toString());
    }

    catch (CannotSendException e) {
        System.err.println(e.toString());
    }
}

protected void onAfterMoving() {
   signConv.setPrivateKey(getKeyPair("Agents own
   keys").getKeyPair().getPrivate());
```

```
    FindAgentBroadcastProtocol conv = (FindAgentBroadcastProtocol)
   newConversation(FindAgentBroadcastProtocol.conversationTypeID,null);
    switch (moveState) {
         case 1:
         // Starting conversation in order to find an agent to receive the
         expected data
                 logMessage("Sending memberid to get member data ...");
                 conv.setMemberID(this.memberID);
                 startConversation(conv);
                 isBroadcasted=true;
                 logMessage("Broadcast for searchig agent sent...");
                 break;
         case 3:
         // Starting conversation in order to find an agent to receive the
         expected data
                 logMessage("Sending memberid and courseid in order to get
                 payment info ...");
                 conv.setMemberID(this.memberID);
                 conv.setCourseID(this.courseID);
                 startConversation(conv);
                 isBroadcasted=true;
                 logMessage("Broadcast for searchig agent sent...");
                 break;
         case 5:
         // Starting conversation in order to find an agent to pay the fees
                 logMessage("Sending memberID, courseID and eventIDs in order to
                 pay the fees ...");
                 String[] eventIDs=new String[paymentData.nrOfEvents];
                 for (int i = 0; i < paymentData.nrOfEvents; i++) {
                   eventIDs[i]=paymentData.paymentEventInfos[i].eventID;
                 }
                 conv.setMemberID(paymentData.memberID);
                 conv.setCourseID(paymentData.courseID);
                 conv.setPrice(paymentData.totAmount);
                 conv.setCurrency(paymentData.currency);
                 conv.setEventIDs(eventIDs, paymentData.nrOfEvents);
                 startConversation(conv);
                 isBroadcasted=true;
                 logMessage("Broadcast for searchig agent sent... state3");
                 break;
     }
 }

 protected void onBeforePersist() throws PersistingDeniedException {
     onBeforeMoving();
 }

 protected void onAfterDePersist() {
     onAfterMoving();
 }

protected javax.swing.JPanel createVisualization() {
     javax.swing.JPanel visualization=new javax.swing.JPanel();
```

```
    visualization.setLayout(new BorderLayout());
    visualization.setBorder(new javax.swing.border.EtchedBorder());
    javax.swing.JPanel pane=new javax.swing.JPanel();
    pane.setLayout(new GridBagLayout());
    pane.setBackground(Color.white);
    pane.setOpaque(true);
    javax.swing.JLabel label=new javax.swing.JLabel(getAgentName());
    GridBagConstraints cons=new GridBagConstraints();
    cons.insets=new Insets(10,10,10,10);
    cons.fill=GridBagConstraints.NONE;
    cons.anchor=GridBagConstraints.CENTER;
    cons.gridwidth=GridBagConstraints.REMAINDER;
    pane.add(label,cons);
    javax.swing.JButton doDlgButton=new javax.swing.JButton( "Dialog...");
    doDlgButton.addActionListener(this);
    pane.add(doDlgButton,cons);
    visualization.add(pane);
    return visualization;
}

public void actionPerformed (ActionEvent e) {
    showDialog();
}

protected void showDialog() {
    switch (state) {
        case 0: if (!this.isBroadcasted) {
                    chooseFuncDlg= new ChooseFunctionDlg(this, null);
                    chooseFuncDlg.setVisible(true);
                }
                break;
        case 1: if (this.moveState == 2) {
                    memberDataDlg= new MemberDataDlg(this, null, data);
                    memberDataDlg.setVisible(true);
                      this.state=0;
                    this.isBroadcasted=false;
                }
                else {
                    // working
                    JOptionPane.showMessageDialog(memberDataDlg, "Your agent is
                    searching the data", "Agent working",
                    JOptionPane.INFORMATION_MESSAGE);
                }
                break;
        case 2: if (this.moveState == 4) {
                        PaymentInfoDlg paymentInfoDlg= new PaymentInfoDlg(this,
                    null, paymentData);
                        paymentInfoDlg.setVisible(true);
                    }
                else {
                    // working
```

```
                    JOptionPane.showMessageDialog(memberDataDlg, "Your agent is
                    searching the data", "Agent working",
                    JOptionPane.INFORMATION_MESSAGE);
                }
                break;
        case 3: if (this.moveState == 6) {
                    MemberDataDlg memberDataDlg= new MemberDataDlg(this, null,
                    null);
                    if (this.feePaid.equals("Yes")) {
                        if (this.nrOfPaidFees == 1) {
                            JOptionPane.showMessageDialog(memberDataDlg, "The
                    Payment is committed:  your open fee has now been paid",
                    "Agent ready", JOptionPane.INFORMATION_MESSAGE);
                        }
                        else {
                            JOptionPane.showMessageDialog(memberDataDlg, "The
                    Payment is committed:  your "+this.nrOfPaidFees+" open fees
                    have now been paid", "Agent ready",
                    JOptionPane.INFORMATION_MESSAGE);
                        }
                    }
                    else {
                        JOptionPane.showMessageDialog(memberDataDlg, "The
                    Payment is failed: An error occured during payment", "Agent
                    ready", JOptionPane.INFORMATION_MESSAGE);
                    }
                }
                else {
                    // working
                    JOptionPane.showMessageDialog(memberDataDlg, "Your agent is
                    trying to pay", "Agent working",
                    JOptionPane.INFORMATION_MESSAGE);
                }
                this.state=0;
                this.isBroadcasted=false;
                break;
    }
}
public MemberData extractMemberDataFromDoc (Document doc) {
    MemberData data = new MemberData();
    NodeList nlist = doc.getElementsByTagName("FName");
    if (nlist.getLength() > 0 &&
    nlist.item(0).getFirstChild().getNodeValue()!=null)
       data.fName = (String) nlist.item(0).getFirstChild().getNodeValue();
    nlist = doc.getElementsByTagName("SName");
    if (nlist.getLength() > 0 &&
    nlist.item(0).getFirstChild().getNodeValue()!=null)
       data.sName = (String) nlist.item(0).getFirstChild().getNodeValue();
    nlist = doc.getElementsByTagName("MInitial");
    if (nlist.getLength() > 0 &&
    nlist.item(0).getFirstChild().getNodeValue()!=null)
       data.mInitial = (String) nlist.item(0).getFirstChild().getNodeValue();
    nlist = doc.getElementsByTagName("Title");
    if (nlist.getLength() > 0 &&
    nlist.item(0).getFirstChild().getNodeValue()!=null)
```

```
        data.title = (String) nlist.item(0).getFirstChild().getNodeValue();
    nlist = doc.getElementsByTagName("BDate");
    if (nlist.getLength() > 0 &&
    nlist.item(0).getFirstChild().getNodeValue()!=null)
        data.bDate = (String) nlist.item(0).getFirstChild().getNodeValue();
    nlist = doc.getElementsByTagName("Street");
    if (nlist.getLength() > 0 &&
    nlist.item(0).getFirstChild().getNodeValue()!=null)
        data.street = (String) nlist.item(0).getFirstChild().getNodeValue();
    nlist = doc.getElementsByTagName("PCode");
    if (nlist.getLength() > 0 &&
    nlist.item(0).getFirstChild().getNodeValue()!=null)
        data.pCode = (String) nlist.item(0).getFirstChild().getNodeValue();
    nlist = doc.getElementsByTagName("City");
    if (nlist.getLength() > 0 &&
    nlist.item(0).getFirstChild().getNodeValue()!=null)
        data.city = (String) nlist.item(0).getFirstChild().getNodeValue();
    nlist = doc.getElementsByTagName("Country");
    if (nlist.getLength() > 0 &&
    nlist.item(0).getFirstChild().getNodeValue()!=null)
        data.country = (String) nlist.item(0).getFirstChild().getNodeValue();
    nlist = doc.getElementsByTagName("EMail");
    if (nlist.getLength() > 0 &&
    nlist.item(0).getFirstChild().getNodeValue()!=null)
        data.email = (String) nlist.item(0).getFirstChild().getNodeValue();
    nlist = doc.getElementsByTagName("TelNr");
    if (nlist.getLength() > 0 &&
    nlist.item(0).getFirstChild().getNodeValue()!=null)
        data.telNr = (String) nlist.item(0).getFirstChild().getNodeValue();
    nlist = doc.getElementsByTagName("MemberData");
    if (nlist.getLength() > 0 &&
    nlist.item(0).getAttributes().getNamedItem("MemberID").getNodeValue()!=null)
        data.memberID = (String)
    nlist.item(0).getAttributes().getNamedItem("MemberID").getNodeValue();
    if (nlist.getLength() > 0 &&
    nlist.item(0).getAttributes().getNamedItem("Gender").getNodeValue()!=null)
        data.gender = (String)
    nlist.item(0).getAttributes().getNamedItem("Gender").getNodeValue();
    return data;
}
public PaymentInfo extractPaymentInfoFromDoc (Document doc) {
    PaymentInfo data = new PaymentInfo();

    NodeList nlist = doc.getElementsByTagName("Date");
    if (nlist.getLength() > 0 &&
    nlist.item(0).getFirstChild().getNodeValue()!=null)
        data.date = (String) nlist.item(0).getFirstChild().getNodeValue();
    nlist = doc.getElementsByTagName("Amount");
    if (nlist.getLength() > 0 &&
    nlist.item(0).getFirstChild().getNodeValue()!=null)
        data.amount = (String) nlist.item(0).getFirstChild().getNodeValue();
    nlist = doc.getElementsByTagName("Currency");
    if (nlist.getLength() > 0 &&
    nlist.item(0).getFirstChild().getNodeValue()!=null)
        data.currency = (String) nlist.item(0).getFirstChild().getNodeValue();
```

```
nlist = doc.getElementsByTagName("CoursePaymentInfo");
if (nlist.getLength() > 0 &&
nlist.item(0).getAttributes().getNamedItem("MemberID").getNodeValue()!=null)
   data.memberID = (String)
nlist.item(0).getAttributes().getNamedItem("MemberID").getNodeValue();
if (nlist.getLength() > 0 &&
nlist.item(0).getAttributes().getNamedItem("CourseID").getNodeValue()!=null)
   data.courseID = (String)
nlist.item(0).getAttributes().getNamedItem("CourseID").getNodeValue();

nlist = doc.getElementsByTagName("Fee");
if (nlist.getLength() > 0 &&
nlist.item(0).getAttributes().getNamedItem("FeePaid").getNodeValue()!=null)
   data.feePaid = (String)
nlist.item(0).getAttributes().getNamedItem("FeePaid").getNodeValue();
nlist = doc.getElementsByTagName("EventFee_Date");
data.paymentEventInfos = new PaymentEventInfos[nlist.getLength()];
for (int i=0; i<nlist.getLength(); i++)
        data.paymentEventInfos[i] = new PaymentEventInfos();
data.nrOfEvents=nlist.getLength();
nlist = doc.getElementsByTagName("CourseEventFeePaymentInfo");
if (nlist.getLength() > 0) {
    for (int i=0; i<nlist.getLength(); i++) {
        if (nlist.item(i).getAttributes().getNamedItem(
"EventID").getNodeValue()!=null)
            data.paymentEventInfos[i].eventID =
nlist.item(i).getAttributes().getNamedItem("EventID").getNodeValue();
        else data.paymentEventInfos[i].eventID = "";
        }
        if (data.memberID.equals("")) {
            data.memberID =
nlist.item(0).getAttributes().getNamedItem("MemberID").getNodeValue();
   }
}
nlist = doc.getElementsByTagName("EventFee_Date");
if (nlist.getLength() > 0) {
    for (int i=0; i<nlist.getLength(); i++) {
        if (nlist.item(i).getFirstChild().getNodeValue()!=null) {
           data.paymentEventInfos[i].event_date = (String)
nlist.item(i).getFirstChild().getNodeValue();
        }
        else data.paymentEventInfos[i].event_date = "";
    }
}
nlist = doc.getElementsByTagName("EventFee_Amount");
if (nlist.getLength() > 0) {
    for (int i=0; i<nlist.getLength(); i++) {
        if (nlist.item(i).getFirstChild().getNodeValue()!=null)
            data.paymentEventInfos[i].event_amount = (String)
nlist.item(i).getFirstChild().getNodeValue();
        else data.paymentEventInfos[i].event_amount = "";
    }
}
nlist = doc.getElementsByTagName("EventFee_Currency");
```

```
        if (nlist.getLength() > 0) {
            for (int i=0; i < nlist.getLength(); i++) {
                if (nlist.item(i).getFirstChild().getNodeValue()!=null)
                   data.paymentEventInfos[i].event_currency = (String)
        nlist.item(i).getFirstChild().getNodeValue();
                else data.paymentEventInfos[i].event_currency = "";
            }
            if (data.currency.equals(""))
        data.currency=data.paymentEventInfos[0].event_currency;
        }
        nlist = doc.getElementsByTagName("BEventDate");
        if (nlist.getLength() > 0) {
            for (int i=0; i < nlist.getLength(); i++) {
                if (nlist.item(i).getFirstChild().getNodeValue()!=null)
                   data.paymentEventInfos[i].event_bdate = (String)
        nlist.item(i).getFirstChild().getNodeValue();
                else data.paymentEventInfos[i].event_bdate = "";
            }
        }
        nlist = doc.getElementsByTagName("EEventDate");
        if (nlist.getLength() > 0) {
            for (int i=0; i < nlist.getLength(); i++) {
                if (nlist.item(i).getFirstChild().getNodeValue()!=null)
                   data.paymentEventInfos[i].event_edate = (String)
        nlist.item(i).getFirstChild().getNodeValue();
                else data.paymentEventInfos[i].event_edate = "";
            }
        }
        nlist = doc.getElementsByTagName("ETitle");
        if (nlist.getLength() > 0) {
            for (int i=0; i < nlist.getLength(); i++) {
                if (nlist.item(i).getFirstChild().getNodeValue()!=null)
                   data.paymentEventInfos[i].event_title = (String)
        nlist.item(i).getFirstChild().getNodeValue();
                else data.paymentEventInfos[i].event_title = "";
            }
        }
        nlist = doc.getElementsByTagName("EventFee");
        if (nlist.getLength() > 0) {
            for (int i=0; i < nlist.getLength(); i++) {
                if
                (nlist.item(i).getAttributes().getNamedItem("Event_FeePaid").getNode
                Value()!=null)
                    data.paymentEventInfos[i].event_feePaid = (String)
                nlist.item(i).getAttributes().getNamedItem("Event_FeePaid").getNodeV
                alue();
                else data.paymentEventInfos[i].event_feePaid = "";
            }
        }
        return data;
    }
}
```

CURRICULUM VITAE

Persönliche Daten

Name:	Dipl. Ing. Reisinger Susanne
Geburtsdatum:	19.04.1974
Geburtsort:	Linz
Staatsbürgerschaft:	Österreich
Email:	sreisinger@fim.uni-linz.ac.at

Schulausbildung

1980/81 - 1983/84	Volksschule, 4020 Linz
1984/85 - 1992/93	Bundesrealgymnasium mit besonderer Berücksichtigung der Informatik, 4020 Linz (Reifeprüfung: 7. Juni 1993)

Studium

21.09.1993 - 06.10.1998	Studium der Informatik, Johannes Kepler Universität Linz (Erste Diplomprüfung: 10. Februar 1997) (Zweite Diplomprüfung: 6. Oktober 1998) Diplomarbeit: Integration der Telearbeit im Unternehmen - Einführung von Telearbeit in Unternehmen einschließlich Fallstudie im Architekturbüro
09.09.1994 - 07.09.1998	Kurzstudium der Datentechnik, Johannes Kepler Universität
Seit 22.10.1998	Dr.-Studium der technischen Wissenschaften, Johannes Kepler Universität

Berufspraxis

In den Jahren 1991 - 1997	Jeweils einmonatige Ferialpraxis bei der Firma Universale-Bau AG, ZN-OÖ
23.08.1993 - 05.09.1993	Kinderbetreuung im Rahmen einer Ferienaktion in Obertraun
01.03.1998 - 01.07.1998	Tutoriumstätigkeit am Institut für Systemwissenschaften, Abteilung für angewandte Systemforschung und Statistik, “Übung Statistik 1”
01.05.1998 – 28.02.1999	Systementwurf, Systemprogrammierung und Softwareentwicklung bei Klockmann & Partner, Schörfling
Seit 01.03.1999	Universitätsassistentin am Institut für Informationsverarbeitung und Mikroprozessortechnik (FIM), Johannes Kepler Universität Linz
01.04.1999 – 30.09.1999	Nebenberuflich tätig als Schulungsleiter der Mitarbeiter der Firma Hoffmann-Elektrokohle GmbH, Gosau
September 2000	Lehrtätigkeit im Rahmen der Sommeruniversität der Universität Bremen
Dezember 2000	Lehrtätigkeit im Rahmen der Fort- und Weiterbildung für Lehrerinnen und Lehrer an Allgemeinbildenden Höheren Schulen und Bildungsanstalten für Kindergartenpädagogik, Pädagogisches Institut des Bundes in Oberösterreich

01.03.2000 – 01.02.2001	Im Rahmen der Dissertation tätig als Projektleiter bei der Firma Wellness-Software, Linz
In den Jahren 2000 - 2001	Projektmitarbeit bei einem Projekt zum Thema Intelligente Agenten (gefördert von der Österreichischen Nationalbank, Projekt Nr. 7742 und dem Land Oberösterreich, Wi/Ge 201.515/1-2000/Wwin)
2000/01	Mitglied des wissenschaftlichen Komitees von WOOBS ´01 (Workshop on Object-Oriented Business Solutions)
2001/02	Mitglied des wissenschaftlichen Komitees von WOOBS ´02 (Workshop on Object-Oriented Business Solutions)

Publikationen

Susanne Reisinger: Integration der Telearbeit im Unternehmen Einführung von Telearbeit in Unternehmen einschließlich Fallstudie im Architekturbüro Diplomarbeit. Linz: Juli 1998

Jörg R. Mühlbacher, Susanne Reisinger, Michael Sonntag: Intelligent Agents and XML - A method for accessing webportals in both B2C and B2B E-Commerce. Published in the Conference Proceedings of WOOBS'01 - Workshop on Object-Oriented Business Solutions, 2001

Michael Sonntag, Susanne Reisinger: Important Factors for E-Commerce. In: Hofer Christian, Chroust Gerhard (Ed.): IDIMT-2001. 9th Interdisciplinary Information Management Talks. Linz: Universitätsverlag Rudolf Trauner, 2001, 301-312

Michael Sonntag, Jörg R. Mühlbacher, Susanne Reisinger: Personalization of Web-Based Interfaces for Humans and Agents, Applied to E-Government Portals. In: Maria A. Wimmer (Ed.): Knowledge Management in e-Government. KMGov-2002. Linz: Universitätsverlag Rudolf Trauner 2002, 210-222

Roman Divotkey, Jörg R. Mühlbacher, Susanne Reisinger, Doris Remplbauer: The WeLearn Distance Teaching Framework. Granada: EDEN (European Distance Education Network), 2002 Eden Annual Conference, Open and Distance Learning in Europe and Beyond Rethinking International Co-operation, Conference Proceedings, 2002, 308-314

Jörg. R. Mühlbacher, Stephan C. Mühlbacher, Susanne Reisinger: Learning Arrangements and Settings for Distance Teaching/Coaching/Learning: Best practice report. In: Hofer Christian, Chroust Gerhard (Ed.): IDIMT-2002. 10th Interdisciplinary Information Management Talks. Linz: Universitätsverlag Rudolf Trauner 2002, 205-215

Vorträge

Susanne Reisinger: Telearbeit – Ist das etwas für mich?. Vortrag im Rahmen des KIM - Kommunikation im Mühlviertel, Kollerschlag, 2000

Susanne Reisinger: Deshalb Informatik!. Vortrag im Rahmen des TEC - Tomorrow's Experts in Computing, J.K. Universität Linz, 2001

Susanne Reisinger: Intelligent Agents and XML. Vortrag im Rahmen der 15th European Conference on Object Oriented Programming, Workshop on Object Oriented Business Solutions, Budapest, Juni 2001

Susanne Reisinger.: WeLearn. Gastvortrag im Rahmen von 20 Jahre Studienzentrum Bregenz zum Thema „Neues Lernen im Gespräch", 2001

Susanne Reisinger: WeLearn - a Free Available E-Learning Platform. Gastvortrag im Rahmen der International ERASMUS/SOKRATES Partner Conference, Universität Växjö, Schweden, 2002

Susanne Reisinger: The WeLearn Distance Teaching Framework. Vortrag im Rahmen der Annual Eden (European Distance Education Network) Conference 2002, Granada, Juni, 2002

Susanne Reisinger: Learning Arrangements and Settings for Distance Teaching/Coaching/Learning: Best practice report. Vortrag im Rahmen IDIMT-2002. 10th Interdisciplinary Information Management Talks, Zadov, September, 2002

Susanne Reisinger: Didaktische Modelle für Distance Education. Workshop im Rahmen des Tag des selbstgesteuerten Lernens, Johannes Kepler Universität, Linz, September, 2002

Susanne Reisinger: The New WeLearn Distance Teaching Framework. Vortrag im Rahmen des Tag des selbstgesteuerten Lernens, Johannes Kepler Universität, Linz, September, 2002

Lehrtätigkeit

Lehrtätigkeit an der Johannes Kepler Universität Linz, Institut für Informationsverarbeitung und Mikroprozessortechnik (FIM), SS 1999 – WS 2002/03:

SS 1999:	PR Programmierpraktikum 3
WS 1999/2000:	Gemeinsam mit Prof. J.R. Mühlbacher KV Propädeutikum aus Informatik PR Programmierpraktikum 3 (C++)
SS 2000:	PR Programmierpraktikum 3 (C++) SE Spezielle Kapitel aus Informatik in der Medizin: Sport-Informatik
WS 2000/01:	PR Praktikum Programmiersprache C++ KV Spezielle Kapitel aus Telekooperation: Software für Mobile Computing
SS 2001:	PR Praktikum Programmiersprache C++ SE Sportinformatik KV Spezielle Kapitel aus Computernetzwerke: Sicherheitsfragen

WS 2001/02: PR Praktikum Programmiersprache C++

PR Projektpraktikum Telekooperation

KV Spezielle Kapitel aus Telekooperation: Palm – Programming

Mitarbeit an der KV Propädeutikum

SS 2002: PR Praktikum: Programmiersprache C++

PR Praktikum: Teleteaching/Telelearning

Mitarbeit an der Vorlesung Betriebssysteme

WS 2002/03: PR Praktikum: Programmiersprache C++

KV Spezielle Kapitel aus Telekooperation: Pocket-PC – Programming

PR Projektpraktikum Telekooperation

Mitarbeit an der KV Propädeutikum

Lehrtätigkeit an der Universität Bremen, Fachbereich 3 Mathematik Informatik, Sommeruniversität 1999:

Praktikum: Programmierkurs C++

Praktikum: Programmierkurs LISP

Seminar zum Thema Telearbeit

Lehrtätigkeit gemeinsam mit Prof. J.R. Mühlbacher im Rahmen der Fort- und Weiterbildung für Lehrerinnen und Lehrer an Allgemeinbildenden Höheren Schulen und Bildungsanstalten für Kindergartenpädagogik, Pädagogisches Institut des Bundes in Oberösterreich, 2000

Propädeutikum aus Informatik

Besondere Kenntnisse

Programmiererfahrung in den Sprachen C#, Java, C++, Lisp, Oberon2 (Pascal-Familie)

Berufserfahrung auf dem Gebiet der Softwareentwicklung, des Software-Engineering, Security, Mobile Computing, Teleworking und Distance Education.

Derzeit verstärkt im Bereich des Distance Teaching/Coaching/Learning tätig.

Fremdsprachenkenntnisse in Englisch, Französisch und Spanisch