

Prüfer: Prof. Dr. Kurt Rothermel

Betreuerin: Dr. Cora Burger

Begonnen am: 01.11.1999

Beendet am: 30.04.2000

CR-Nummer: *H.3.3, H.3.4, H.4.1, H.5.2, H.5.3, I.2.11*

Diplomarbeit Nr. *1818*

**Erweiterung von MoleOffice
um gemeinsame
Dokumentenbearbeitung**

Lukas Weberruß

INHALTSVERZEICHNIS

Kapitel 1	Einleitung	1
1.1	Motivation	1
1.2	Aufgabenstellung.....	3
1.3	Übersicht	5
Kapitel 2	Grundlagen	6
2.1	Gemeinsame Dokumentenerstellung.....	6
2.1.1	Beschreibung von Dokumenten.....	6
2.1.2	Das Team	8
2.2	Agententechnologie.....	9
2.2.1	Definitionen	9
2.2.2	Mobile Agentensysteme	11
2.2.3	Eigenschaften.....	12
2.2.4	Bewertung.....	14
2.3	Das Mole System	14
2.3.1	Die Struktur von Mole	14
2.3.2	Mole Lokationen und Agenten	15
2.3.3	Agentenkommunikation	16
2.4	Das MoleOffice System	17
2.4.1	Gesamtkonzept	17
2.4.2	Agenten.....	18
2.4.3	Benutzungsoberfläche.....	19
2.4.4	Kommunikation	20
2.4.5	Benutzerverwaltung und Sicherheit.....	20
2.5	Corba-DDE.....	21
2.5.1	Dokumentenverwaltung	21
2.5.2	Transparenz.....	22
2.5.3	Wrapper	22
2.5.4	Autorenumgebung	23
Kapitel 3	Entwurf des DDE-Systems.....	25
3.1	Randbedingungen.....	25
3.2	Überblick über die Struktur des DDE-Systems.....	26
3.3	Verwendete Dokumentenstruktur	27
3.3.1	Aufbau der Bücherregalkomponenten	27
3.3.2	Identifikation der Dokumente.....	28
3.3.3	Zugriff auf Dokumente	29
3.3.4	Datenhaltung.....	30
3.4	Die Agenten von MoleOffice	31
3.5	Informationsaustausch	34
3.5.1	Explizite Kommunikation.....	34
3.5.2	Implizite Kommunikation.....	34
3.5.3	Informationswege	35
3.6	Einsatz mobiler Agenten	37
3.7	Aktionen und Filter	38

3.8	Sicherheit und Benutzerverwaltung	40
3.8.1	Benutzerverwaltung	40
3.8.2	Authentifizierung	41
3.8.3	Verschlüsselung	41
3.9	Die Editor Umgebung	42
3.9.1	Bedeutung von Einbenutzereditoren in MoleDDE	42
3.9.2	Begriffsbestimmungen	43
3.9.3	Der Emacs in Mole-DDE	43
3.10	Wrapper	47
3.10.1	Definition	47
3.10.2	Überblick	48
3.10.3	Informationsumsetzung	49
3.10.4	Emacs Wrapper	49
3.11	Schnittstellen	50
3.11.1	Schnittstelle zwischen Mole-DDE und Wrapper	50
3.11.2	Schnittstelle zwischen Emacs und Wrapper	52
3.11.3	Kommunikationsprotokoll	52
3.12	Die Benutzungsoberfläche	55
3.12.1	Überblick	55
3.12.2	Das MoleOffice Applet	56
3.12.3	Das DDE Informations Fenster	58
3.12.4	Das DDE Setup Fenster	60
3.12.5	Der Startbildschirm	60
3.12.6	Das Loginfenster	61
3.12.7	Die Passwortverwaltung	61
Kapitel 4 Implementierung		63
4.1	Entwicklungswerkzeuge	63
4.1.1	Compiler	63
4.1.2	Das Java Development Environment	64
4.2	Implementierung	65
4.2.1	Das Bücherregal	65
4.2.2	Der Taktgeber der Agenten	66
4.2.3	Agentenkommunikation	68
4.2.4	Wrapper Kommunikation	69
4.2.5	Transport von Informationseinheiten	71
4.2.6	Verwaltung der Kapitel	72
4.2.7	Filterung von Aktionen	73
4.2.8	Anbindung der Benutzungsoberfläche	75
Kapitel 5 Testszenarien		80
5.1	Testumgebung	80
5.2	Test des Dokumentensystems	81
5.3	DDE System	82
5.4	Editor Umgebung	86
5.5	Vergleich mit CorbaDDE	88
Kapitel 6 Installation und Konfiguration		89
6.1	Installation von Mole-DDE	89

6.2	Konfiguration des Server.....	90
6.2.1	Mole.....	90
6.2.2	Laufzeitumgebung.....	90
6.2.3	Neuübersetzung.....	91
6.2.4	MoleOffice Konfigurationsprogramm.....	91
6.2.5	Starten und Beenden des Servers.....	93
6.2.6	Konfiguration des Klienten.....	93
6.2.7	Starten und Beenden des Klienten.....	94
Kapitel 7	Zusammenfassung.....	95
7.1	Bewertung.....	95
7.2	Ausblick.....	96
Kapitel 8	Anhang.....	98
8.1	Literaturverzeichnis.....	98
8.2	Code Beispiele.....	100
8.3	Klassenhierarchie.....	103
8.3.1	mole.dde.....	103
8.3.2	mole.dde.util.....	104
8.3.3	mole.dde.gui.....	105
8.3.4	mole.dde.wrapper.....	106
8.3.5	mole.moleschedule.....	107
8.4	GNU General Public License.....	109

ABBILDUNGS- UND TABELLENVERZEICHNIS

Abb. 1	:	Granularität der Struktur eines Dokumentes	7
Abb. 2	:	Umgebung von Agenten	10
Abb. 3	:	Darstellung von Objekten und Agenten	12
Abb. 4	:	Das Mole System	15
Abb. 5	:	Agenten des alten MoleOffice Systems	19
Abb. 6	:	Hierarchische Struktur von Dokumenten	21
Abb. 7	:	Definition einer IE in Corba-DDE	23
Abb. 8	:	Komponenten der Mole-DDE Plattform	26
Abb. 9	:	hierarchische Dokumentenstruktur des DDE-Systems	27
Abb. 10	:	Agenten-Struktur des DDE-Systems	32
Abb. 11	:	Informationsfluß der Editor Daten	36
Abb. 12	:	Datenfilterung	37
Abb. 13	:	Bücherregalmodus im XEmacs	44
Abb. 14	:	Büchermodus im XEmacs	45
Tab. 1	:	Bücherregal Kommandos	45
Abb. 15	:	Kapitelmodus im XEmacs	46
Tab. 2	:	Bücher Kommandos	46
Tab. 3	:	Kapitel Kommandos	47
Abb. 16	:	Der Wrapper als Schnittstelle	48
Abb. 17	:	Definition einer IE in BNF	51
Abb. 18	:	Kommunikationsprotokoll zwischen MoleDDE und Emacs	53
Abb. 19	:	Beispiel eines Sitzungsprotokolls	54
Abb. 20	:	Hauptfenster von MoleOffice	57
Abb. 21	:	Setup-Fenster von MoleOffice	58
Abb. 22	:	MoleDDE Informationsfenster	59
Abb. 23	:	MoleDDE Setup Fenster	60
Abb. 24	:	MoleDDE Start Fenster	61
Abb. 25	:	MoleDDE Loginfenster	61
Abb. 26	:	MoleDDE Passwort Konfigurationsfenster	62
Abb. 27	:	Beispiel zum Einsatz des HeartBeat-Mechanismus	67
Abb. 28	:	Erweiterte Agentenkommunikation	69
Abb. 29	:	Kommunikation zwischen DDE Agenten und Wrapper	71
Tab. 4	:	Zustandsinformation eines Kapitels	72
Abb. 30	:	Filterung von Aktionen	74
Tab. 5	:	Laufzeitumgebungen für die grafische Benutzungsoberfläche	78

Abb. 31	:	Analyse des Bücherregals	81
Abb. 32	:	Zusammenhang v. Zugriffen u. Dateigröße bei Bücherregalzugriffen	82
Abb. 33	:	Testumgebung zur Leistungsmessung des DDE Systems	83
Abb. 34	:	Auswirkung mehrerer Klienten	85
Abb. 35	:	Prozentuale Verteilung der Bearbeitungszeit	87
Tab. 6	:	Archive von Mole-DDE	90
Abb. 36	:	Konfigurationsdialog von MoleOffice	91
Abb. 37	:	Benutzerkonfiguration von MoleOffice	92
Abb. 38	:	Konfigurationsergebnisse	92
Abb. 39	:	Konfigurationsbeispiel eines MoleOffice Klienten	93
Abb. 40	:	Methode sendMessage	100
Abb. 41	:	Methode callRemote	100
Abb. 42	:	Methode receiveMessage (Teil 1)	101
Abb. 43	:	Methode receiveMessage (Teil 2)	102

KAPITEL 1

EINLEITUNG

1.1 Motivation

In den letzten Jahren haben sich die Möglichkeiten der Zusammenarbeit zwischen Menschen erheblich verändert. Eine der wichtigsten Veränderungen war bedingt durch die starke Verbreitung des Internets. Sie ermöglicht einen globalen Informationsaustausch und vor allem die Zusammenarbeit örtlich verteilter Computer.

Durch die erweiterten Möglichkeiten, die das Internet bietet, haben sich auch ganz neue Anforderungen an die Computer und die verwendete Software ergeben. Die meisten Projekte könnten heutzutage ohne die Hilfe eines Computers nicht mehr durchgeführt werden. Das Internet bietet die strukturellen Möglichkeiten gemeinsamer Projekte, die Computerhersteller überbieten sich mit ständig neuen Leistungsrekorden, doch fehlen oft noch Anwendungen dazu. Email und WWW haben sich zwar erfolgreich durchgesetzt, alleine mit diesen Kommunikationsmitteln lassen sich jedoch nicht alle Anforderungen einer ortsübergreifenden gemeinsamen Arbeit erfüllen.

Besonders im Bereich gemeinsam benutzter Dokumente entstehen häufig große Koordinationsprobleme, wenn mehrere Personen gemeinsam und eventuell gleichzeitig an einem Projekt arbeiten. Diese Probleme entstehen vor allem auch deswegen, weil es immer noch an geeigneten computerunterstützten Mitteln fehlt, die automatisch oder halbautomatisch für eine Synchronisation der Dokumente und Dokumentinformationen sorgen.

Das noch relativ junge Forschungsgebiet "Computer Supported Cooperative Work (CSCW)" befaßt sich mit den Möglichkeiten und Problemen, die die gemeinsame Arbeit weltweit verteilter Teams sowie ihre Unterstützung durch Rechner bietet. Ein Teilgebiet davon sind verteilte Dokumentensysteme, die beispielsweise das gemeinsame, gleichzeitige Editieren von Programmcode und Dokumentation unterstützen sollen.

Üblicherweise werden hier sogenannte Mehrbenutzereditoren eingesetzt. Das sind Werkzeuge, die unter einer Oberfläche die Fähigkeiten von normalen Einbenutzereditoren und die zusätzlichen Kommunikations-, Koordinations- und Kooperationsfähigkeiten zur Arbeit an gemeinsamen Dokumenten integrieren.

Die Mitglieder eines Teams können heute weltweit verteilt und mobil sein. Durch die sich oft dynamisch ändernde Struktur der Teams können sich auch die Rollen der Benutzer stetig verändern. Durch die Mobilität können einzelne Personen zeitweise nicht erreichbar sein. Auch durch kulturell bedingte Unterschiede zwischen verschiedenen Ländern können sich die Arbeitszeiten und Arbeitsweisen erheblich voneinander unterscheiden. Mit der Integration und der daraus entstehenden Anforderungen sind die Mehrbenutzereditoren oft überfordert, vor allem, was die unterschiedlichen Vorlieben und Gewohnheiten der Benutzer angeht. Neue Systeme werden oft nur dann akzeptiert, wenn sich die Benutzer möglichst wenig an die neue Situation anpassen müssen.

Eine Vielzahl unterschiedlicher Einbenutzereditoren versucht ebenfalls dieses Spektrum der Anforderungen zu erfüllen. Jedoch fehlt ihnen die zusätzlichen Fähigkeiten der Mehrbenutzereditoren zur gemeinsamen Dokumentenbearbeitung. Jedes dieser Programme zur Dokumentenerstellung hat dabei spezielle Vor- und Nachteile. Wichtige Vertreter dieser heterogenen Welt von Editoren sind derzeit *MICROSOFT WORD*TM oder *STAROFFICE*TM von Sun Microsystems¹ oder *FRAME MAKER*TM. In der *UNIX* Welt hat sich der unter der *GPL*² stehende *EMACS* stark verbreitet.

Diese Einbenutzereditoren können ausschließlich dazu verwendet werden quasi parallel zu arbeiten und dies auch nur dann, wenn zusätzliche Möglichkeiten zur Kommunikation und Kooperation bereitstehen. Selbst dann müssen sich alle daran beteiligten Personen untereinander absprechen und selbst für die nötige Konsistenz der Daten sorgen. Nur zu oft gelingt dies auf Grund der Komplexität der Struktur beziehungsweise der Synchronisationsanforderungen jedoch nicht.


Meistens ist es aber trotz aller Heterogenität und der Gefahr von Inkonsistenzen notwendig, daß mehrere Teammitglieder echt gleichzeitig an ein und demselben Doku-

1. ursprünglich von Stardivision entwickelt, 1999 von Sun Microsystems übernommen.

2. GNU General Public License

ment arbeiten. Viele Projekte können wegen ihrer Komplexität und der oft engen Zeitvorgaben nicht mehr als Einpersonenprojekte durchgeführt werden.

Eine Lösung stellt eine gemeinsame Arbeitsumgebung mit computerunterstützter Synchronisation dar, die es ermöglicht, trotz unterschiedlicher Editoren gemeinsam zu arbeiten. Es muß eine Umgebung geschaffen werden, die Unterstützung für die heterogene Editorwelt bietet und durch die Anbindung an ein Verteiltes Dokumentensystem dem Benutzer die Arbeit erleichtert.

 **Das Internet bietet zwar die technischen Möglichkeiten gemeinsamer verteilter Arbeit, es fehlt jedoch immer noch an geeigneten Anwendungen.**

1.2 Aufgabenstellung

Die Wurzeln dieser Arbeit liegen in zwei unterschiedlichen Projekten. Im Rahmen eines der Projekte wurde schon einmal ein verteiltes Dokumentensystem entwickelt. Dieses im weiteren Corba-DDE¹ genannte System ist eine verteilte Arbeitsumgebung, bei der die verschiedenen Editoren über eine Plattform mit definierten Schnittstellen miteinander kommunizieren können. Bei dieser Arbeit wurde auf das weitverbreitete System *CORBA*² zurückgegriffen, durch dessen Verwendung man auf der einen Seite Vorteile wie beispielsweise eine verkürzte Entwicklungszeit erreichte, andererseits ist die Implementierung in ihrer ganzen Struktur fest mit *CORBA* verknüpft und Teile davon sind kaum mehr für andere Projekte wiederzuverwenden, die nicht auf Basis dieser Plattform implementiert werden sollen.

In einigen schon abgeschlossenen Arbeiten wurde bereits ein System namens *MOLE-OFFICE* auf der Basis des mobilen Agentensystems *MOLE* entwickelt, welches über eine grafische Benutzeroberfläche den Teamteilnehmern die Möglichkeiten bietet, sich über Benutzer und Räumlichkeiten zu informieren. Es gibt Auskunft über aktive Tätigkeiten

-
1. DDE ist die englische Abkürzung für ein verteiltes Dokumentensystem (Distributet Document Environment)
 2. Common Object Request Broker Architecture

und die Bereitschaft zur Kommunikation. Es stellt also ein Benachrichtigungssystem zum Austausch von Informationen dar.

Ziel der vorliegenden Arbeit ist es, dieses zweite Projekt um Komponenten zur gemeinsamen Dokumentenerstellung, analog zum bestehenden Corba-DDE, zu erweitern. Es bietet damit insbesondere die Möglichkeit, über verschiedene Editoren hinweg gemeinsam zu arbeiten. Benutzer können sich an diesem System an- und abmelden, sowie mobil und von einem beliebigen Ort aus auf das System zugreifen. In diesem Projekt wird wie bei Corba-DDE exemplarisch ein Wrapper¹ für den Editor *EMACS* erstellt, der mit dem System über eine wohldefinierte Schnittstelle kommunizieren kann.

Des Weiteren wird *MOLEOFFICE* um eine Agentenstruktur erweitert, die es ermöglicht, den Zugriff auf Dokumente zu koordinieren. Hierzu werden sowohl Systemagenten zur Dokumentenbearbeitung als auch weitere (mobile) Agenten als Schnittstelle zwischen den Benutzern bzw. dem Emacs Wrapper und den Dokumentenagenten erstellt.

Durch die Rückmeldung vom System soll dem Benutzer der aktuelle Bearbeitungszustand grafisch visualisiert werden.

Dabei werden im Rahmen dieser Arbeit zum einen geeignete Konzepte vorgestellt, als auch eine prototypische Implementierung durchgeführt. Das System wird abschließend durch geeignete Testszenarien und Leistungsmessungen evaluiert.



Ziel dieser Arbeit ist die Erstellung eines prototypischen Verteilten Dokumentensystems auf der Basis mobiler Agenten.

1. Ein Wrapper stellt die Verbindung zwischen einem Editor und dem DDE-System da.

1.3 Übersicht

Nachdem im vorherigen Abschnitt ein Überblick über die Aufgabenstellung dieser Arbeit gegeben wurde, soll nun in diesem Abschnitt die Struktur der Arbeit erläutert werden.

Die vorliegende Arbeit teilt sich in 8 Kapitel. Nach der Einleitung folgt eine Einführung in wichtige Grundlagen in Kapitel 2. Behandelt werden sowohl allgemeine Informationen zu Agenten als auch eine Beschreibung der dieser Arbeit zugrunde liegenden Projekte.

Kapitel 3 erläutert den Entwurf. Zu Beginn erfolgt ein kurzer Überblick aller beteiligten Komponenten. Danach wird der Entwurf der Komponenten Bücherregal, Agentensystem, Editorumgebung, Wrapper und die grafische Oberfläche vorgestellt.

Die Beschreibung wichtiger Details der Implementierung erfolgt in Kapitel 4. Zusätzlich wird hier auch die verwendete Entwicklungsumgebung überblicksartig präsentiert.

Nach der Implementierung erfolgten ausgiebige Tests am DDE-System. Kapitel 5 stellt zum einen die Testprogramme vor, zum anderen werden die Ergebnisse kommentiert.

Kapitel 6 beschreibt den zum Einsatz von Mole-DDE notwendigen Installations- und Konfigurationsvorgang.

Schließlich wird in Kapitel 7 zusammenfassend dargestellt, welche Ergebnisse bei dem Projekt festzuhalten sind. Zudem erfolgt ein kurzer Ausblick auf zukünftige Erweiterungsmöglichkeiten.

In Kapitel 8 befinden sich sowohl das Literaturverzeichnis als auch kurze beispielhafte Codefragmente. Ergänzend liegt ein Ausschnitt aus der Klassenhierarchie von Mole-DDE vor.

KAPITEL 2

GRUNDLAGEN

Im folgenden werden die beiden dieser Arbeit zugrunde liegenden Projekte *CORBA-DDE* und *MOLEOFFICE* vorgestellt, da sie die Basis dieser Arbeit darstellen.

Zuvor wird eine kurze Einführung in die Themengebiete gemeinsame Dokumentenerstellung und Agententechnologie gegeben.

2.1 Gemeinsame Dokumentenerstellung

Diese Arbeit verwendet häufig den allgemeinen Begriff eines Dokuments. Um zu verstehen welche Bedeutung dieser Begriff hat, soll nun definiert werden, was unter einem Dokument im folgenden zu verstehen ist.

2.1.1 Beschreibung von Dokumenten

Beim Umgang mit Dokumenten ist es wichtig, eine Trennung in Daten, Layout und Struktur vorzunehmen. Die Komponenten haben dabei folgende Bedeutung:

- **Daten:** Diese bilden den aus Text, Abbildungen und Videosequenzen bestehenden Informationsgehalt des Dokumentes.
- **Layout:** Um den Informationsgehalt darstellen zu können, muß dieser in einem bestimmten Format präsentiert werden. Dem Leser eines Dokumentes muß die Informationsentnahme durch unterschiedliche Schriftgrößen oder Unterstreichungen erleichtert werden.
- **Struktur:** Die Gliederung des Dokuments in Kapitel, Abschnitte, Abbildungen oder Aufzählungen bezeichnet man als die Struktur eines Dokumentes.

Speziell für diese Arbeit ist es wichtig, daß auf die Daten effizient zugegriffen werden kann und bei Replikaten diese synchron gehalten beziehungsweise verarbeitet werden. Im Bereich der Datenhaltung kann zur Erhöhung der Performance ein geeigneter Caching Mechanismus den Aufwand der Zugriffe verringern, so muß beispielsweise bei mehrmaligem Lesen des gleichen Dokuments nicht jedesmal auf die Daten vom externen Speichermedium zugegriffen werden.

Entscheidend für die Qualität der Darstellung ist vor allem der Einsatz entsprechender Formate. Es ist Aufgabe eines guten Editors, die Formate nach den Wünschen der Benutzer darzustellen. Da die Darstellung der Daten Aufgabe des Editors ist und deshalb für ein Verteiltes Dokumentensystem keine entscheidende Rolle spielt, wird im folgenden nicht näher auf das Layout eines Dokuments eingegangen.

Von großer Bedeutung ist jedoch die Struktur eines Dokuments für diese Arbeit. Sie zergliedert das Dokument in Teilbereiche, die miteinander verknüpft sind. Dies kann in unterschiedlicher Granularität geschehen. Eine sehr feine Granularität wäre gegeben, würde man jedes einzelne Wort eines Absatzes als einzelnes Strukturelement betrachten.

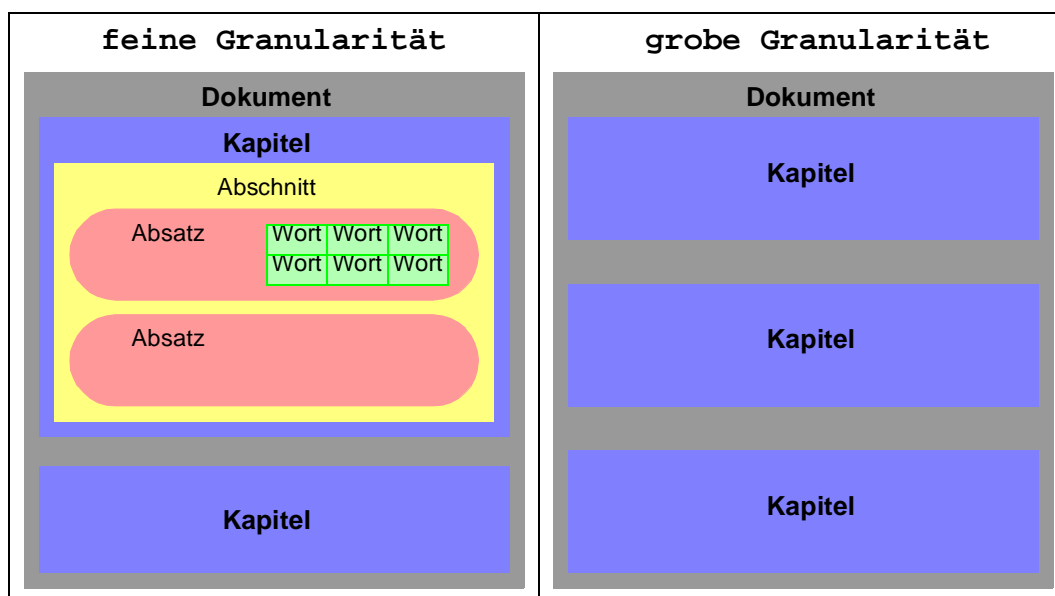


ABBILDUNG 1. Granularität der Struktur eines Dokumentes

Das Ziel einer solchen Gliederung ist es, Komponenten zu erhalten, die unabhängig voneinander behandelt werden können. Je feiner die Granularität ist, desto mehr Komponenten erhält man. Dadurch erhöht sich jedoch auch der Verwaltungsaufwand bzw. der Überhang an Informationen zur Verknüpfung dieser Komponenten. Deshalb ist eine sehr feine Gliederung häufig nur in der Theorie, jedoch nicht in der Praxis sinnvoll. Die Unabhängigkeit einzelner Komponenten und deren Granularität ist für ein Verteiltes Dokumentensystem von entscheidender Bedeutung, da Mechanismen wie Zugriffskontrollen oder Datenverwaltung diese direkt verwenden.

Der Zugriff auf Dokumente muß synchronisiert stattfinden. Ein Dokumentensystem muß stets gewährleisten können, daß es nicht zu ungewollten Inkonsistenzen kommt. Gibt es mehrere Replikate eines Dokuments, muß eine einfache eventuell automatische Möglichkeit vorhanden sein, diese wieder zusammen zu führen.

2.1.2 Das Team

Bei vielen Projekten sind Komplexität und Umfang der Arbeit für einen einzelnen Entwickler zu groß. Ein Team¹, bestehend aus mehreren, möglicherweise geographisch verteilten Teamteilnehmern muß diese Rolle übernehmen.

Durch ihren kooperativen, sich in den Fähigkeiten der einzelnen Teilnehmer ergänzenden Character, sind Teams eher dazu in der Lage, solche Projekte zu einem erfolgreichen Ende zu führen. Unterstützung bekommen sie hierbei durch geeignete Groupware-Systeme.

Im Bereich dieser Groupware-Systeme ist für die vorliegende Arbeit speziell die gemeinsame Dokumentenerstellung von großer Bedeutung. Die Teilnehmer eines Teams wollen gemeinsam und zum Teil gleichzeitig an bestimmten Teilen der Dokumente arbeiten und dabei unter anderem Informationen über den momentanen Bearbeitungszustand des Dokumentes erhalten. Je nachdem welche Rolle ein bestimmter Benutzer im Team einnimmt, hat er spezielle Rechte und Aufgaben beim Zugriff auf die Dokumente. So kann beispielsweise ein Kommentator das vorhandene Dokument nicht löschen, sondern nur in einem speziell dafür vorgesehenen Bereich Anmerkungen einbringen.

1. Definition des Begriffes Team kann in [Burger, 1997] nachgelesen werden.

Teamteilnehmer lassen sich auf die folgende Art und Weise charakterisieren:

- Die Teilnehmer eines Teams verfolgen ein gemeinsames Ziel.
- Jeder verfügt über besondere Kenntnisse in einem Teilbereich des Projekts, die er in das Team einbringt.
- Ein Teilnehmer verfügt über einen bestimmten aktuellen Arbeitsplatz und bestimmte Arbeitszeiten, die sich zum Teil durch die geographische Verteilung ergeben und sich ebenfalls dynamisch ändern können.
- Innerhalb eines Teams übernimmt jeder eine bestimmte Rolle und damit ein bestimmtes Aufgabengebiet. Diese Rollenvergabe kann auch dynamisch sein und führt zu einer (eventuell hierarchischen) Struktur innerhalb des Teams.

Um Rechte und Aufgaben des Teams auch elektronisch verwalten zu können, bedarf es eines administrativen Benutzers, der zum einen die Definitionen der Rollen vornimmt und zum anderen den Teilnehmern die Rollen zuweisen kann.

2.2 Agententechnologie

Im folgenden wird ein kurzer Überblick über die Agententechnologie und deren Besonderheiten vorgestellt.

2.2.1 Definitionen

Laut Duden handelt es sich bei einem Agenten um jemanden, “der im Auftrag für einen anderen eine Aufgabe erledigt”. Diese Definition beschreibt zwar, aus welchem Hintergrund heraus die Namensgebung für Agenten entstand, im Bereich der Informatik existieren allerdings weitergehende Definitionen von Agenten. Exemplarisch sollen an dieser Stelle zwei Definitionen näher beschrieben werden.

In [WoJe94] ist ein Agent “ein Programm, welches in der Lage ist, seine Entscheidung und sein Handeln basierend auf der Wahrnehmung seiner Umwelt, bei der Verfolgung eines oder mehrerer Ziele selbständig zu kontrollieren”.

Dabei muß es die folgenden Eigenschaften erfüllen:

- **Autonomie:** Die Agenten lösen eine ihnen gestellte Aufgabe selbständig, d.h. ohne weitere Benutzereingriffe und mit eigenständiger Kontrolle über ihr Handeln und ihren internen Status.
- **Kooperationsfähigkeit:** Besonders wichtig für ein Multiagenten System ist, daß Agenten in der Lage sind, mit anderen Agenten und Anwendern zusammenzuarbeiten.
- **Reaktionsfähigkeit:** Agenten beobachten ihre Umgebung und reagieren auf auftretende Veränderungen.
- **Unternehmungsgeist:** Im Gegensatz zu Objekten reagieren Agenten nicht nur auf Veränderungen, sie ergreifen sogar die Initiative bei der Verfolgung ihrer Ziele.

Diese Kriterien geben bei der Entwicklung von Agenten einen relativ engen Rahmen vor. Etwas allgemeiner gehalten ist die Definition von [WoMi1999]. Ein Agent ist demnach “ein Computer System, welches sich in einer speziellen Umgebung befindet und zum Erreichen seiner Entwurfziele die Fähigkeit besitzt, autonome Aktionen auszuführen.”

Er kann beispielsweise über Sensoren seine Umgebung überprüfen und diese über Aktionen verändern. Abbildung 2 zeigt dieses Szenario. Der Agent hat dabei allerdings nicht die totale Kontrolle über seine Umgebung. Es können auch andere Komponenten auf die Umgebung einwirken. Desweiteren kann der Agent aus Sicherheitsaspekten auch nur partiellen Zugriff auf die Umgebung haben.

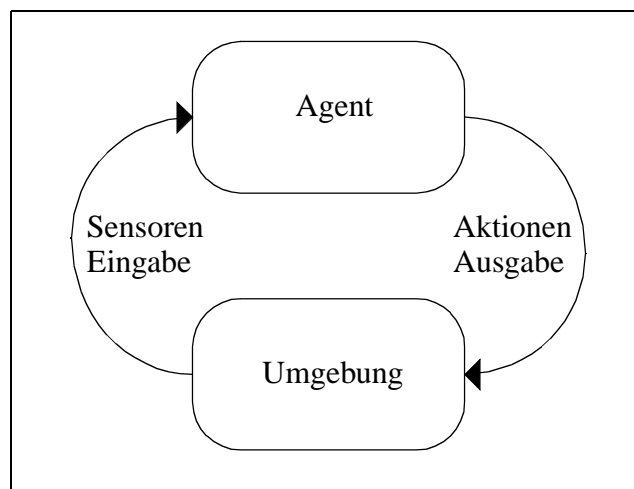


ABBILDUNG 2. Umgebung von Agenten

Eine Einteilung der Agentenumgebungen kann aufgrund der Art und Weise erfolgen, wie ein Agent seine Umgebung wahrnimmt. Es gibt dabei die folgenden Unterscheidungskriterien:

- **zugänglich vs. unzugänglich:**

Eine zugängliche Umgebung ist eine Umgebung, die vom Agenten vollständig erfaßt werden kann, bzw. von der er aktuelle und akkurate Informationen über dessen Zustand bekommen kann. Die meisten komplexeren Umgebungen, wie beispielsweise das Internet, sind eher unzugänglich.

- **deterministisch vs. nicht-deterministisch:**

Von einer deterministischen Umgebung spricht man, wenn jede Aktion einen einzigen garantierten Effekt hat und der Folgezustand der Umgebung klar definiert ist. Die reale Welt ist sicher nicht-deterministisch.

- **abgeschlossene Phasen vs. durchgehend:**

In einer Umgebung mit abgeschlossenen zeitlichen Phasen hängt die Leistungsfähigkeit des Agenten direkt von einer diskreten Anzahl von Phasen ab. Dabei besteht keine Abhängigkeit zu der Leistungsfähigkeit von Agenten in anderen Szenarien. Solche Umgebungen mit abgeschlossenen Phasen sind leichter zu handhaben, da die Agenten ihre Aktionen rein von der aktuellen Phase abhängig machen können.

- **statisch vs. dynamisch:**

Bei einer statischen Umgebung erfolgen die einzigen Veränderungen durch den Agenten selbst. Eine dynamische Umgebung hat weitere Prozesse, die auf ihr operieren, wodurch der Agent keine Kontrolle mehr über die Veränderungen hat. Die meisten Multiagentensysteme haben dynamische Umgebungen.

- **diskret vs. nicht-diskret:**

Man bezeichnet eine Umgebung als diskret, wenn es eine feste endgültige Anzahl von Aktionsmöglichkeiten gibt. Ein Schachspiel ist beispielsweise diskret, wogegen Taxifahren sicher nicht diskret ist.

Durch Verwendung dieser Kriterien läßt sich das DDE-System hinsichtlich seiner Agenten klassifizieren. Es soll in den folgenden Kapiteln daran gemessen werden.

2.2.2 Mobile Agentensysteme

Ein Spezialfall eines Agentensystems stellt ein mobiles Agentensystem dar. Hier ergeben sich durch die Mobilität der Agenten besondere Anforderungen.

Damit in Agentensystemen ein gewisses Maß an Sicherheit herrscht, muß man die lokalen Ressourcen vor unerlaubtem Zugriff schützen. In mobilen Agentensystemen, bei denen Agenten von einer Lokation zur nächsten wandern können, kann ihnen beispielsweise der Zugriff auf die Umgebung komplett untersagt werden. Um an Informationen zu kommen, müssen sie mit ortsfesten Agenten kooperieren.

Deshalb können im Bereich mobiler Agentensysteme die Agenten aufgrund ihrer Befugnisse und Möglichkeiten in zwei Klassen eingeteilt werden:

- **Systemagenten:**
Die Agenten sind ortsfest, können sich also nicht von einer zu einer anderen Stelle bewegen. Sie haben Zugriff auf Systemressourcen.
- **Mobile Agenten:**
Die Agenten können migrieren, d.h. von einer Lokation zu einer anderen wechseln. Es dreht sich dabei um eine Prozeßmigration, bei der die Information über den aktuellen Zustand beibehalten wird. Mobile Agenten dürfen aus Sicherheitsgründen nicht auf alle Systemressourcen zugreifen, sondern müssen sich über ortsfeste Systemagenten Zugang zu Ressourcen verschaffen.

Diese Aufteilung ist für die hier vorliegende Arbeit von besonderer Bedeutung, da auch das als Grundlage dienende *MOLE*¹ diese Unterscheidung macht.

2.2.3 Eigenschaften

Agenten und insbesondere Mobile Agenten sind ein relativ neues Programmiermodell. Während die traditionellen objektorientierten Ansätze ein Netz von Komponenten bilden, welche in einem nach außen offenen System arbeiten und dort Dienste anbieten, ist der Ansatz bei einem Mobilem Agentensystem ein anderer.

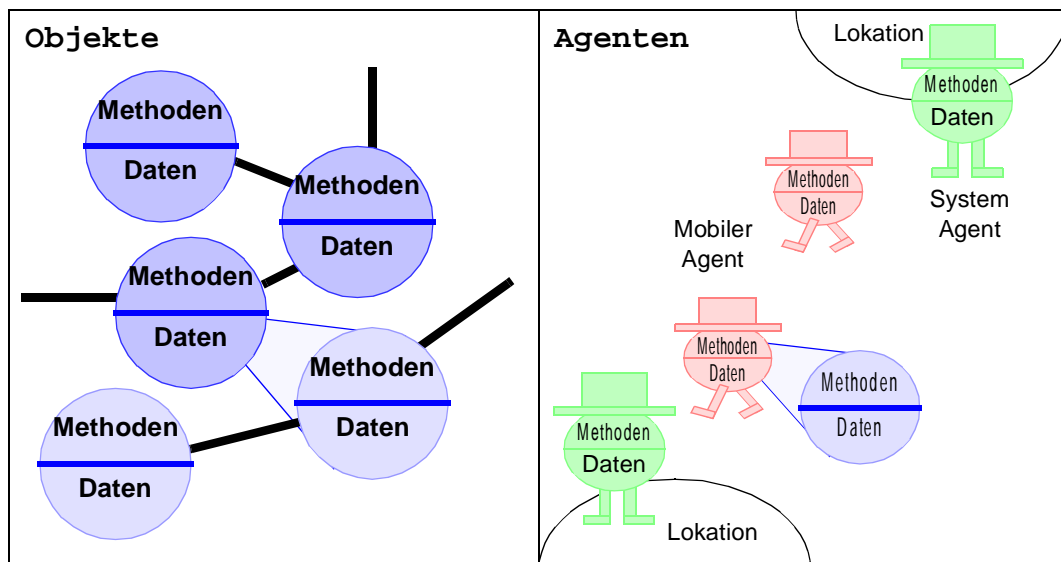


ABBILDUNG 3. Darstellung von Objekten und Agenten

1. siehe Abschnitt 2.3

Ein Agentensystem ist ein in sich geschlossenes System, in dem sich die Agenten frei bewegen können, Dienste anbieten oder vollbringen können und dabei ein gemeinsames kooperatives Ziel verfolgen. Die Agenten können diese Umgebung nicht verlassen. Prinzipiell sind zwar durchaus auch Umgebung mit konkurrierenden Agenten möglich; diese sollen aber in der vorliegenden Arbeit nicht näher betrachtet werden.

Die oben erläuterte Agentenumgebung ermöglicht überhaupt erst eine Ausführung der Agenten. Sie stellt sogenannte Lokationen zur Verfügung: Orte, an denen sich die Agenten aufhalten können. Die Agentenumgebung teilt sich in zwei Bereiche. Auf das Agentensystem selbst können alle Agenten zugreifen (auch die mobilen). So können alle Agenten miteinander kommunizieren. Sie befinden sich in einer Art Sandkasten, aus dem die mobilen Agenten nicht heraus können. Zusätzlich können Systemagenten auch die externe Umgebung beobachten und verändern.

Die Umgebung teilt sich also in einen Bereich, der für alle Agenten zugänglich ist und in einen, auf den nur die Systemagenten zugreifen können.

Agenten entscheiden selbst, ob eine ihrer Methoden aufgerufen wird oder nicht. Dies differenziert sie klar von Objekten, bei denen normalerweise das Objekt nicht darüber entscheidet, ob ein Aufruf seiner Methoden Sinn macht oder nicht.

2.2.4 Bewertung

Durch die Unterschiede zu *herkömmlichen* Programmierparadigmen ergeben sich sowohl Vor- als auch Nachteile. Diese bestehen in der Art der Programmierung und den Möglichkeiten ihrer Ausführung.

- **Vorteile:**

- Durch die Fähigkeit zur Migration können sich die Agenten zu den Daten begeben und nicht umgekehrt, was bei großen Datenmengen vorteilhaft ist
- Agenten sind autonom, d.h. sie entscheiden kontextabhängig selbst, ob sie Methoden ausführen oder nicht.
- Agenten sind dem Menschen nachgebildet, wodurch ihre Arbeitsweise leichter verständlich ist.

- **Nachteile:**

- Alle Agenten können nur innerhalb eines Agentensystems existieren.
- Durch den Unterschied zur objektorientierten Programmierung erfordert die Agentenprogrammierung ein Umdenken.
- Durch die Migration dürfen mobile Agenten nur eingeschränkt auf Systemressourcen zugreifen.
- Der Mehraufwand der Agenten-Kommunikation bzw. des Agentensystems sorgt für einen Leistungsverlust.



Durch den Einsatz von Agentensysteme ergeben sich neue Möglichkeiten und Probleme in der Realisierung komplexer Szenarien.

2.3 Das Mole System

Wie bereits erwähnt, setzt diese Arbeit das mit *MOLEOFFICE* begonnene Projekt, welches selbst auf der Basis des mobilern Agentensystems Mole realisiert wurde, fort. Deshalb will ich in diesem Kapitel etwas näher auf die Besonderheiten von *MOLE* eingehen.

2.3.1 Die Struktur von Mole

Abbildung 4 zeigt eine etwas vereinfachte Struktur eines auf der Basis von Mole laufenden Systems. Durch die Verwendung von Java als Programmiersprache kann Mole

auf verschiedenen Betriebssystemen laufen. Mole selbst stellt dabei pro Rechner beliebig viele Lokationen zur Verfügung und pro Rechner eine “*MOLE ENGINE*”: eine Art übergeordneter Prozeß, der die anderen Prozesse überwacht.

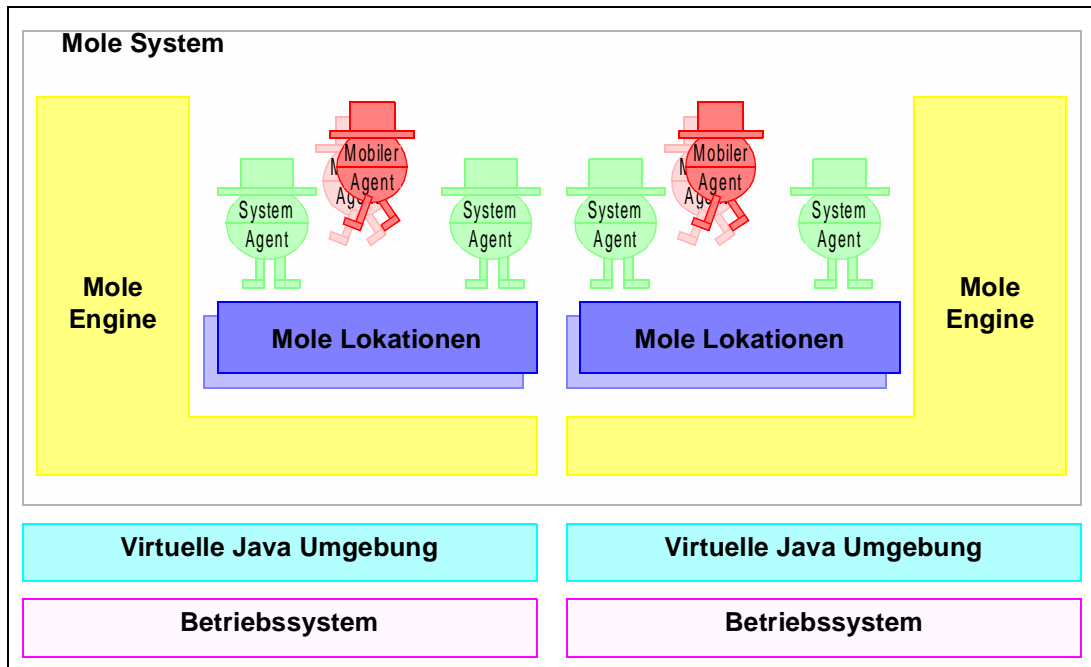


ABBILDUNG 4. Das Mole System

Mole stellt auch eine Java Klassenbibliothek zur Verwendung in eigenen Projekten dar. Agenten selbst gibt es in *MOLE* noch nicht, deren Entwicklung bleibt vollständig dem Benutzer von *MOLE* überlassen.

Die “Mole Engine” muß zwar auf jedem Rechner separat gestartet werden. Das System abstrahiert dabei allerdings von den unterschiedlichen Rechnern, es ist also vor dem Benutzer dahingehend transparent, daß der Benutzer nur die Lokationen seiner Agenten, nicht aber den eigentlichen Rechner kennt. Es können sogar prinzipiell mehrere abgeschlossene Agentensysteme innerhalb eines *MOLE* Systems laufen, ohne sich gegenseitig zu behindern.

2.3.2 Mole Lokationen und Agenten

Wie schon erwähnt gibt es in Mole sowohl Klassen für mobile Agenten, als auch für Systemagenten, wobei nur letztere direkt auf die Umgebung außerhalb des Mole Systems zugreifen können. Agenten haben einen eindeutigen Namen, der der Identifi-

kation innerhalb des Systems dient, wobei die “*Engine*” darauf achtet, daß die Namen eindeutig bleiben¹.

Die Lokationen sind Orte, an denen sich die Agenten treffen und lokal Informationen austauschen können. Sie sind vor dem Start des Systems statisch anzulegen. Jede Lokation hat einen eindeutigen Namen, ähnlich einem Rechner im Internet, beispielsweise “*moleoffice1.mole.informatik.uni-stuttgart.de*”. Jede Lokation kann vor dem Start des Systems mit Agenten “gefüllt” werden. Diese Agenten werden dann automatisch durch die entsprechenden Lokationen gestartet. Es muß also mindestens einen solchen Agenten im gesamten System geben, der statisch angelegt wurde und dann selbst beliebig andere Agenten erschaffen kann.

Jeder Agent hat eine bestimmte Lebenszeit, die damit beginnt, daß die entsprechende Lokation die “*start-Methode*” der Klasse aufruft und die endet, wenn die “*die-Methode*” aufgerufen wird.


Bei der Migration von Agenten werden diese kurzzeitig beendet, über den Serialisierungsmechanismus von Java in einer Objektdatei gespeichert, an die neue Lokation verschoben und dort schließlich erneut durch den Aufruf der “*start*”-Methode gestartet, ohne daß die Informationen der Daten verloren gehen². Da nach dem erneuten Start des mobilen Agenten sich die Informationen seiner Umgebung verändert haben können, muß darauf geachtet werden, daß globale Variablen, die speziell für die alte Lokation gültig waren, noch immer ihre Gültigkeit besitzen.

2.3.3 Agentenkommunikation

Die Kommunikation zwischen den Agenten kann auf zweierlei Weisen geschehen. Zum einen ist es möglich, eine Methode eines Agenten direkt aufzurufen. Dabei wird von der API unterschieden, ob die Agenten sich innerhalb der gleichen “*Engine*” befinden oder nicht. Agenten, die sich auf der gleichen “*Engine*” befinden, übergeben sich die Daten dabei als Referenz, im anderen Fall wird per RPC eine Kopie der Daten übergeben.

-
1. Leider hat sich im Laufe dieser Arbeit herausgestellt, daß dem nicht immer so ist, so kam es mehrfach vor, daß Agentennamen doppelt vorkamen, was natürlich zu einem völlig abnormalen Verhalten des Systems führt.
 2. Nähere Informationen zum Thema Serialisierung findet man in den meisten Java Büchern.

Die zweite Methode ist das Versenden von beliebigen Nachrichten, wobei der Empfänger selbst entscheiden kann, ob er die Nachrichten gleich verarbeiten will oder sie in einer Mailbox aufbewahrt, um sie zu einem späteren Zeitpunkt abzuarbeiten. Die Nachrichten sind ähnlich wie Emails aufgebaut; sie haben einen Absender, einen Empfänger und einen Inhalt.

 **Mole stellt eine einheitliche Umgebung dar, in der unterschiedliche Agentenklassen an verteilten Lokationen miteinander kommunizieren können.**

2.4 Das MoleOffice System

Wie im vorherigen Kapitel beschrieben, stellt Mole eine Plattform zur Erstellung eigener Projekte auf der Basis von Agenten dar. *MOLEOFFICE* wurde auf dieser Basis im Rahmen einer Diplomarbeit von Jörg Zimmer entwickelt¹. Es stellt einen Benachrichtigungsdienst in einem verteilten Team dar. *MOLEOFFICE* wurde mittels *JDK VERSION 1.0.X* entwickelt und basiert auf *MOLE 1.X*. Aus Zeitmangel wurde diese Version nicht mehr vollständig implementiert, weshalb Sven Tränkle *MOLEOFFICE* in einer Studienarbeit² zu einem lauffähigen System erweiterte. Er entschied sich, das System auf die damals aktuelle Version des *JDK 1.1.X* umzustellen. Auch von *MOLE* wurde die neue Version 3.0 verwendet. Dies brachte große Veränderungen mit sich, vor allem an der grafischen Benutzungsoberfläche.

2.4.1 Gesamtkonzept

Grundsätzlich läßt sich die bisherige Version von *MOLEOFFICE* in die Bereiche Benutzungsoberfläche, Agenten und von diesen beiden Bereichen benötigte Benutzer und Raumstrukturen gliedern. Bei *MOLEOFFICE* handelt es sich bisher um ein Benachrichtigungssystem, vergleichbar mit elektronischen Türzetteln. Allerdings wurde die Integration zusätzlicher Komponenten von Anfang an geplant. Angefangen beim integrierten

1. Benutzeragenten zur Unterstützung von Teamkoordination [ZiJö1996]

2. Realisierung eines CSCW-Benachrichtigungsdienstes mit MOLE-Agenten [TrSv1997]

Email-System, mit dem man per Mausklick anderen Benutzern Emails schreiben kann, bis hin zur vollständig integrierten Videokonferenz, stellt *MOLEOFFICE* somit ein komplettes Office-Paket dar. Auch die Verwendung externer Programme, wie beispielsweise der Sun Kalendermanager, wurde mit einbezogen. Eine verteilte Dokumentenbearbeitung war bisher allerdings nicht vorgesehen.

2.4.2 Agenten

Die Funktionalität des Gesamtsystems wird durch das Zusammenwirken aller beteiligten Komponenten erreicht. Da die Informationen des Systems nicht in einem zentralen Knoten zusammen laufen, kann man *MOLEOFFICE* deshalb als ein Verteiltes System bezeichnen. Es werden bisher die folgenden Agentenklassen verwendet:

- **Administratoragent:**
Im gesamten *MOLEOFFICE* muß genau ein zentraler Systemagent existieren, der Informationen über alle am System beteiligten Benutzer, Benutzeragenten und deren Aufenthaltsorte hat.
- **Benutzeragenten:**
Für jeden Benutzer existiert genau ein weiterer Systemagent, der ihn in *MOLEOFFICE* vertritt. Er verwaltet alle Zustandsinformationen und sendet Informationen an bei ihm angemeldete mobile Agenten anderer Benutzer. Er bildet gleichzeitig die Schnittstelle zur grafischen Oberfläche.
- **Mobiler Beobachtungsagent:**
Um Informationen von anderen Benutzeragenten zu erhalten, schickt dieser an alle Agenten, von denen er Informationen wünscht, einen solchen mobilen Agenten zur Beobachtung der Vorgänge. Dort empfängt er alle Ereignisse und filtert diese entsprechend den Benutzerwünschen, um sie schließlich an seinen Benutzeragenten weiterzureichen.

Durch den bisherigen statischen Charakter der Systemagenten war es nicht möglich, diese auf unterschiedliche Lokationen zu verteilen. Zukünftig wäre es natürlich wünschenswert, die Systemagenten so nah wie möglich beim Benutzer zu halten¹.

1. Alexander Kramer verwendet in seiner Arbeit schon ein Replikat des Systemagenten, welcher sich lokal auf dem mobilen Endgerät befindet.

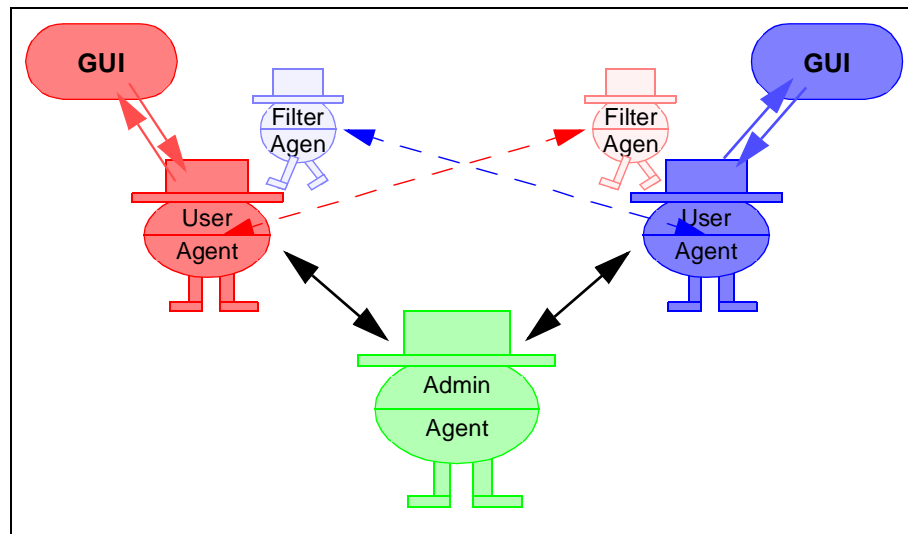


ABBILDUNG 5. Agenten des alten MoleOffice Systems

Die Abbildung verdeutlicht die Struktur des Systems und das Zusammenwirken der einzelnen Komponenten. Der Benutzer sieht von seiner Sicht aus eine grafische Oberfläche, über die er Einstellungen seines virtuellen Abbilds im System vornehmen kann und über die er über Aktivitäten anderer Teilnehmer informiert wird.

2.4.3 Benutzungsoberfläche

Die Qualität der Benutzungsoberfläche entscheidet oft darüber, ob ein System in der Praxis eingesetzt wird oder nicht.

Um die Akzeptanz des Systems zu erhöhen, entschied man sich für die Verwendung eines Applets, das sowohl in Suns Appletviewer als auch mit Einschränkungen im Netscape-Communicator lauffähig ist.

Die Oberfläche präsentiert sich nach dem Start mit grafischen Bildern, die die Kommunikationsbereitschaft der Teamkollegen verdeutlichen soll. Man kann sich durch einfache Mausklicks Informationen über seine Teamkollegen, die vorhandenen Räume und die persönlichen Einstellungen holen. Eintretende Ereignisse werden automatisch angezeigt und falls gewünscht akustisch signalisiert.

Im Konfigurationsbereich kann man umfangreiche Einstellungen bezüglich seiner Person und seiner Informationsfilter vornehmen und das elektronische Abbild der Räume den örtlichen Gegebenheiten anpassen.

2.4.4 Kommunikation

Die Kommunikation zwischen der grafischen Benutzungsoberfläche und dem Systemagenten erfolgt über den Mechanismus “*entfernter Methodenaufrufe*”¹. Dazu stellt dieser eine Kommunikationsschnittstelle zur Verfügung und ruft selbst Methoden der Kommunikationsschnittstelle des Systemagenten auf. Allerdings gestatten *Applets* eine Netzwerk-Kommunikation normalerweise nicht. Es müssen dazu spezielle Anpassungen für die verschiedenen Appletviewer gemacht werden. Die Kommunikation zwischen den Agenten des Systems erfolgt ausschließlich über Nachrichten, wobei der Empfänger durch den Inhalt der Nachricht entscheidet, wie mit der Nachricht zu verfahren ist. Der Absender wird nicht zwangsläufig überprüft, woraus man schließen kann, daß das Agentensystem sich in einer Kooperativen Umgebung befindet und nicht von “*fremden*” Agenten im System ausgegangen wird.

2.4.5 Benutzerverwaltung und Sicherheit

In einem verteilten System, an dem mehrere Benutzer teilhaben, Informationen einholen, diese zur Verfügung stellen oder Modifikationen an Dokumenten vornehmen können, muß sichergestellt sein, daß geeignete Verwaltungsmechanismen vorhanden sind, um eine eindeutige Authentifizierung zu ermöglichen.

In der mir vorliegenden und hier diskutierten Version von MoleOffice existieren bisher keinerlei Mechanismen zur eindeutigen Authentifizierung. Es kann deshalb nicht davon ausgegangen werden, daß Informationen über die Teamkollegen auch wirklich von ihnen stammen. Die Identifikation beim Start des Klienten erfolgt ausschließlich über den Agentennamen, der in einem HTML-Tag angegeben werden muß. Die Anmeldung erfolgt dann beim Systemagenten, der diese akzeptiert, sofern der Agentenname stimmt. Durch die Anmeldung beim Systemagenten muß dieser schon vor der eigentlichen Anmeldung im System vorhanden sein. Um eine dynamische Umgebung zu schaffen, könnte die Anmeldung zukünftig auch über den Administratoragent verlaufen, welcher daraufhin einen neuen Benutzeragent startet.

1. siehe Java und RMI (Remote Method Invocation)

Die Benutzerverwaltung selbst ist bisher ebenfalls statisch. Um einen Benutzer hinzuzufügen oder zu entfernen müssen die Veränderungen direkt im Java Quellcode vorgenommen, schließlich das System neu übersetzt und gestartet werden.

☞ **Aufbauend auf Mole stellt MoleOffice ein elektronisches Benachrichtigungsdienst zur Unterstützung von Teams dar.**

2.5 Corba-DDE

In einer Arbeit von Oliver Schramm¹ wurde bereits ein DDE-System entwickelt. Dabei wurde eine Erweiterung des Editors Emacs (bzw. XEmacs) vorgenommen, sowie ein geeigneter Wrapper zur Umsetzung des Informationsflusses zwischen Editor und DDE-System.

2.5.1 Dokumentenverwaltung

Laut Schramm handelt es sich bei Dokumenten um *aufgezeichnete Informationen*. Bei allgemeiner Information, wie beispielsweise einem Gespräch, werden zwar ebenfalls Informationen ausgetauscht, diese aber nicht zwangsläufig aufgezeichnet.

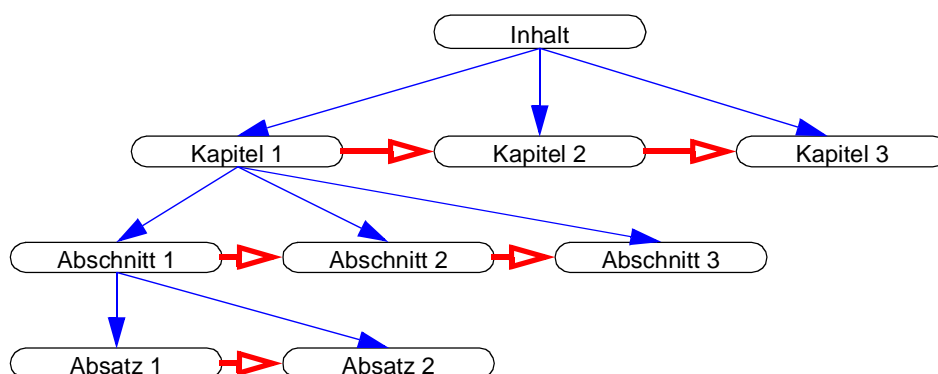


ABBILDUNG 6. Hierarchische Struktur von Dokumenten

1. Entwurf eines verteilten corbabasierten Dokumentensystems [ScOl1999]

Nach Abbildung 6 werden Dokumente nach ihrer *rekursiven Struktur* ausgehend von dem Strukturelement Überschrift aufgebaut und zusätzlich sequentiell miteinander verknüpft.

2.5.2 Transparenz

Ziel eines verteilten Dokumentensystems ist es, die zur Verfügung stehenden Ressourcen transparent zu verwalten. Eine völlige Nebenläufigkeitstransparenz ist allerdings zu restriktiv, da die Teamteilnehmer einer Gruppe sich durch diese Transparenz ihrer gegenseitigen Existenz nicht bewußt sind, dieses Bewußtsein aber Grundlage für ein kooperatives Handeln ist. Aus diesem Grund unterscheidet Schramm die drei folgenden Aspekte in Abhängigkeit des Arbeitsvorgangs:

- **Berücksichtigung von persönlicher Arbeitspraxis**
Bei vorgegebener Arbeitsstruktur ist die Transparenz anderer Teilnehmer wünschenswert.
- **Möglichkeit der Fokussierung**
Wichtige Nachrichten müssen durch eine Fokussierung aus dem Informationsfluß hervor gehoben werden.
- **Wahrung der Privatsphäre**
Es muß möglich sein, Aktionen an einem Dokument durchzuführen, ohne daß andere davon in Kenntnis gesetzt werden.

Es ist also für ein Dokumentensystem wichtig, den richtigen Grad an Transparenz zu unterstützen. Vorteilhaft zeichnet sich demnach die Möglichkeit aus, eine *individuell konfigurierbare* Transparenz zur Verfügung zu stellen.

2.5.3 Wrapper

Zur Kommunikation zwischen der eingesetzten Autorenumgebung und der Dokumentenplattform entwirft Schramm einen Wrapper, der exemplarisch am Beispiel zwischen dem Editor Emacs und der Corba-DDE vorgestellt wird.

```

<IE> ::= "<" <ID> ">" <IETYPE> ">" <IECONTENT> ">" <NEWLINE>

<IETYPE> ::= <EMACS2WRAPPER> | <WRAPPER2EMACS>

<EMACS2WRAPPER> ::= "msg" | "reset" |
                    "init" | "close" | "bookShelfToc" |
                    "bookToc" | "openBook" | "closeBook" |
                    "createBook" | "destroyBook" |
                    "openSection" | "closeSection" |
                    "readSection" | "writeSection" |
                    "createSection" | "destroySection"

<WRAPPER2EMACS> ::= "bookshelf" | "book" | "section"

<ID> ::= ASCII Zeichen 65-90 und 97-122

<IECONTENT> ::= ASCII Zeichen 32-127

<NEWLINE> ::= ASCII Zeichen 10

```

ABBILDUNG 7. Definition einer IE in Corba-DDE

Die Schnittstelle zwischen Wrapper und Dokumentensystem ist durch die Corba-Orientierung des Projekts in Corba-IDL spezifiziert und soll hier nicht näher erläutert werden. Abbildung 7 zeigt die Syntax der Schnittstelle zwischen Wrapper und Emacs in einer BNF-ähnlichen Notation.

2.5.4 Autorenumgebung

Die Autorenumgebung stellt in Corba-DDE die einzige Schnittstelle zwischen Benutzer und DDE-System dar. Als exemplarisches System wurde dazu der Editor Emacs ausgewählt. Dieser stellt durch die Möglichkeit zur Erweiterung über in Emacs-Lisp geschriebenen Modulen eine geeignete Plattform für eigene Projekte dar. Um die für den Einsatz in einem DDE-System nötige Funktionalität zu bekommen, wurden hierzu Module für die folgenden Modi des Editors geschrieben:

- **Bücherregalmodus :**

Hier werden alle zur Auswahl stehenden Bücher als eine Art Inhaltsverzeichnis aufgeführt.

- **Büchermodus :**

Durch die Auswahl eines Buches gelangt man in den Büchermodus, in dem analog zum Bücherregalmodus das Inhaltsverzeichnis des aktuellen Buches gezeigt wird.

- **Kapitelmodus :**

Der Kapitelmodus stellt den eigentlichen Arbeitsmodus dar. Er vereint die Funktionalität eines normalen *LATEX*-Modus mit der Funktionalität, die speziell beim Einsatz eines Verteilten Dokumentensystems nötig ist.

In Corba-DDE bekommt man durch einen zusätzlichen Fensterbereich des Emacs Informationen darüber, wer gerade ein Kapitel geöffnet hat oder aus welchem Grund eine Aktion nicht durchgeführt werden konnte. Der Editor reagiert allerdings auf diese Meldungen nur im Erfolgsfall. Im Fehlerfall muß der Benutzer selbst die Systemmeldungen interpretieren und die nötige Aktion veranlassen.

KAPITEL 3

ENTWURF DES DDE-SYSTEMS

In den folgenden Abschnitten soll der prinzipielle Aufbau von Mole-DDE vorgestellt werden. Es wird diskutiert, welche Veränderungen an *MOLEOFFICE* vorgenommen wurden und welche Vor- und Nachteile verschiedene Varianten haben. Ein Schwerpunkt liegt auf der Erarbeitung einer Möglichkeit zur Integration von Mole-DDE an die gegebenen Randbedingungen.

3.1 Randbedingungen

Ziel dieser Arbeit ist es nicht, ein komplettes DDE-System von Grund auf neu zu entwerfen. In Kapitel 2.4 und Kapitel 2.5 wurden bereits die beiden dieser Arbeit zugrunde liegenden Projekte vorgestellt.

Die in Projekt Corba-DDE gemachte Arbeit an Editor und Wrapper sollte soweit wie möglich in dieses Projekt übernommen werden. Ebenso sinnvoll erschien es, die im Kapitel "Dokumentenverwaltung" auf Seite 21 dargestellten Überlegungen in dieses Projekt einfließen zu lassen.

Auf der anderen Seite existiert schon das auf Mole aufbauende *MOLEOFFICE* System. Das hier vorgestellte System wird vollständig in *MOLEOFFICE* integriert und stellt somit eine natürliche Erweiterung dar.

3.2 Überblick über die Struktur des DDE-Systems

Mole-DDE läßt sich prinzipiell in fünf Teilkomponenten gliedern:

- **Editor**
Auf oberster Ebene befinden sich die Editoren, die für die eigentliche Arbeit mit dem Dokument und dessen Layout zuständig ist
- **grafische Benutzungsoberfläche**
Zusätzlich zum Editor befindet sich auf der obersten Ebene auch eine eigene grafische Benutzungsoberfläche.
- **Agentensystem**
Das Agentensystem dient zur eigentlichen Verarbeitung der Informationen und stellt die Hauptkomponente des Verteilten Dokumentensystems dar.
- **Wrapper**
Die Editoren werden über einen Datenumsetzer, dem Wrapper, an das darunterliegende Agentensystem angeschlossen.
- **Bücherregalkomponenten**
Auf der unterster Ebene befinden sich die Komponenten des Bücherregals. Diese greifen direkt auf das Dateisystem zu.

Zur Kommunikation zwischen den verschiedenen Ebenen kommen unterschiedliche Protokolle zur Einsatz. So kommuniziert beispielsweise die GUI mit den Agenten über entfernte Methodenaufrufe, die Wrapper direkt über Sockets.

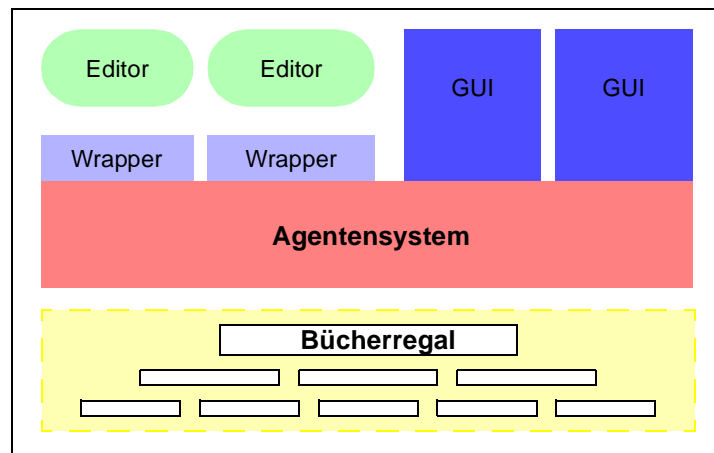


ABBILDUNG 8. Komponenten der Mole-DDE Plattform

Die Abbildung 8 zeigt, daß man sich das System als Plattform vorstellen kann, dessen zentrale Rolle das Agentensystem einnimmt.

Das gesamte MoleOffice teilt sich zusätzlich in die Bereiche Klient und Server. Dabei existiert für jeden angemeldeten Benutzer genau ein Klient und für das gesamte System genau ein Server:

- **Server :**
Zum Server gehört das Agentensystem mit dem Wrapper Server und das Bücherregal
- **Klient :**
Zum Klienten zählt die grafische Benutzungsoberfläche, der Wrapper und der Editor

3.3 Verwendete Dokumentenstruktur

Wie in Abschnitt 2.1.1 beschrieben, bedarf es einer geeigneten Struktur der Dokumente beziehungsweise deren Verwaltung. Ziel ist es, die Dokumentenstruktur sowohl flexibel, erweiterbar als auch effizient zu gestalten. Um die Integration der schon vorhandenen Komponenten des Corba-DDE zu ermöglichen, wurde die prinzipielle Struktur der Komponenten stark an die vorhandene Struktur angelehnt.

3.3.1 Aufbau der Bücherregalkomponenten

Mole-DDE unterscheidet verschiedene Bücherregale. Diese enthalten einzelne Bücher, welche wiederum in einzelne Kapitel unterteilt werden. Diese Anordnung ergibt, wie Abbildung 9 verdeutlicht, eine hierarchische Struktur.

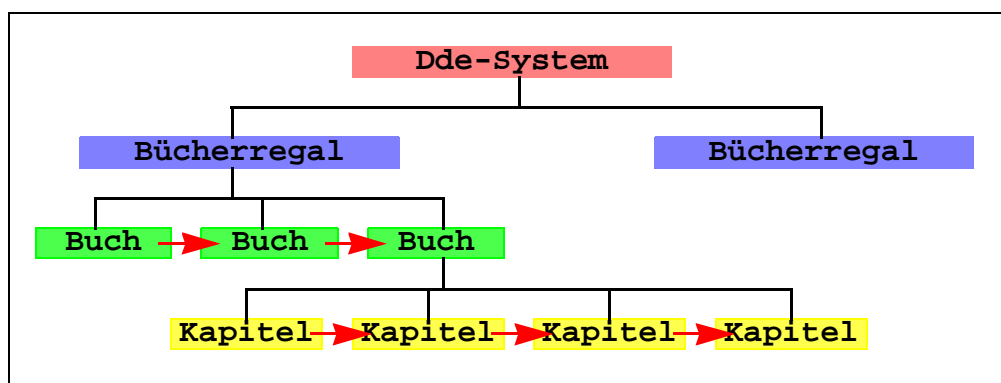


ABBILDUNG 9. hierarchische Dokumentenstruktur des DDE-Systems

Neben dieser hierarchischen Struktur existiert auch eine sequentielle Reihenfolge der Kapitel innerhalb eines Buches bzw. der Bücher innerhalb eines Bücherregales.

- **Bücherregal**
Die Unterscheidung in verschiedene Bücherregale dient lediglich zur Trennung des Systems in verschiedene Projekte, was grundsätzlich nichts Neues bringt und hier nicht weiter betrachtet wird. Es wird im folgenden immer nur von *einem* Bücherregal ausgegangen.
- **Buch**
Unter einem Buch versteht man ein zusammenhängendes, aus mehreren Teilen bestehendes Dokument, wie beispielsweise eine Diplomarbeit.
- **Kapitel**
Ein Kapitel ist eine beliebige abgeschlossene Teilkomponente eines Buches. Einleitung oder Literaturverzeichnis wären beispielsweise solche Kapitel.

Der Entwurf ist stark an das existierende DDE-System angelehnt, in dem ebenfalls die Unterscheidung zwischen Bücherregal, Büchern und Kapiteln vorgenommen wurde. Dies geschah vor allem aus Kompatibilitätsgründen. Es sollten sowohl der Wrapper als auch die Erweiterungen am Editor *EMACS* übernommen werden. Auch können somit zukünftige weiterführende Projekte, wie die Integration weiterer Editoren, leichter von einem Projekt zum anderen portiert werden.

Die Gliederung der Bücherregalkomponenten stellt zwar nur eine relativ grobe Struktur dar, dennoch können bei einer Synchronisation auch einzelne Wörter abgeglichen werden. Die vorhandene Objektstruktur erlaubt eine weitere Untergliederung der Kapitel in Abschnitte beziehungsweise Wörter. Sie ist aber bisher noch nicht implementiert.

Im Bereich des Feinentwurfs und der Implementierung konnte durch die *CORBA* Orientierung im vorhandene *CORBA-DDE* vom Programmcode, einschließlich Wrapper, trotz objektorientierter Programmierung nichts übernommen werden.

3.3.2 Identifikation der Dokumente

Die Ordnung der Bücher muß über einen zusätzlichen Identifikationsmechanismus gewährleistet werden, der bei fehlender Angabe bei der Erstellung entweder alphabetisch oder zeitlich sequentiell ergänzt werden muß. Die Identifikation der Kapitel sollte die natürliche Position innerhalb des Buches sein. Auch hier kann man beim Erzeugen angeben, an welcher Position ein neues Kapitel eingefügt werden soll. Beide Ordnungen sollen während des Betriebs vom Benutzer frei verändert werden können.

Zur eindeutigen Identifikation der Bücher und Kapitel innerhalb von MoleOffice wurden zwei Alternativen betrachtet:

- Identifikation über eine *eindeutige Nummer*.
- Identifikation über den *Namen* (Kapitel-, Buch-, Bücherregal-).

Eine Nummer bietet den Vorteil, daß sie bei automatischer Generierung immer eindeutig ist; sie hat allerdings auch den Nachteil, nicht sehr aussagekräftig zu sein. Bei der Identifikation über den Namen muß dagegen darauf geachtet werden, daß kein Bücher- oder Kapitelname doppelt vergeben werden.

Deshalb habe ich mich für eine Kombination beider Verfahren entschieden. Mole-DDE selbst kann sowohl mit Namen zur Identifikation umgehen als auch eindeutige Nummern verwenden. Bei Kapiteln geschieht die Identifikation über Nummern einfach durch deren Position im Buch¹. Die hier exemplarisch verwendete Editorumgebung (Emacs und Emacs-Wrapper) kann allerdings nur mit einer Kombination aus Buch- und Kapitelnamen zur eindeutigen Identifizierung umgehen, was eine Doppelvergabe eines Kapitelnamens innerhalb eines Buches zwar ausschließt, aber dem Benutzer nicht die Möglichkeit gibt, die Bücher und Kapitel neu anzuordnen. In Mole-DDE selbst ist dies jedoch möglich. Auch die Umbenennung eines Buches oder Kapitels ist in Mole-DDE möglich, wird aber auf der obersten Ebene im Editor nicht angeboten.

3.3.3 Zugriff auf Dokumente

Alle Komponenten dieser Dokumentenhierarchie besitzen eigene Protokollierungsfunktionen. Sämtliche Modifikationen werden festgehalten, so daß sowohl dem System als auch den Teamteilnehmern ersichtlich ist, welche Historie das Dokument aufweist. Es ist beispielsweise ersichtlich, wer das Dokument angelegt, wer es seit seiner Erstellung verändert oder auch wer es gerade bearbeitet hat.

Um den Zugriff auf die Dokumente zu kontrollieren ist es nötig, Zugriffsbestimmungen einzuführen². So haben beispielsweise die Kapitel spezifische Zugriffsrechte, die bestimmte Aktionen entweder gestatten oder untersagen. Diese Zugriffslisten können dabei entweder pro Benutzer, pro Dokument oder aber pro Rolle vergeben werden.

1. Diese ist per Definition immer eindeutig.

2. man spricht im Englischen auch von Access Control Lists (ACLs).

Eine Rechtevergabe pro Rolle erfordert natürlich eine spezielle Rollenverteilung¹, eine pro Benutzer eine erweiterte Benutzerverwaltung bzw. ein administratives Benutzerkonto.

Für die Einführung eines administrativen Benutzerkontos gibt es die folgenden Alternativen zur Festlegung der Gültigkeitsbereiche:

- **Systemweite Gültigkeit**
Im gesamten System existiert genau ein administratives Konto zur übergeordneten Verwaltung aller Rechte.
- **Projektspezifische Administration**
Pro Bücherregal (oder Buch) existiert eine spezielle Rolle zur Administration aller projektbezogenen Aspekte.
- **Dokumentspezifische Administration**
Pro Dokument (Kapitel) existiert eine Konto, welches die Zugriffsmöglichkeiten festlegt. Diese Aufgabe kann entweder eine bestimmte Rolle übernehmen oder aber beispielsweise der Besitzer (Erzeuger) eines Dokuments.

Setzt man alle diese Varianten gleichzeitig ein, bekommt man eine hierarchisch gegliederte Administration des Systems, in dem beispielsweise ein übergeordneter Administrator Projekte erzeugen und weitere untergeordnete projektbezogene Administratoren einsetzen kann. Diese können wiederum pro Dokument speziellen Personen die Aufgabe der Administration zuteilen.

3.3.4 Datenhaltung

Um die Datenhaltung der Bücherregalkomponenten effizient zu gestalten, wurde sowohl ein Lese- als auch ein Schreibcache innerhalb der Dokumentenstruktur integriert. Dokumente werden erst bei einem Lesezugriff in den Speicher geholt und verharren dort, bis auf sie eine gewisse Zeit lang nicht mehr zugegriffen wurde. Ein Schreibzugriff wird unmittelbar nur im Speicher vorgenommen.

1. siehe Kapitel 2.1.2 auf Seite 8.

Erst eine automatisch ablaufende Autospeicherung schreibt veränderte Daten wieder zurück auf das Speichermedium, damit sie auch persistent und bei einem erneuten Start des Systems dem Bücherregal vorliegen.



Zur strukturierten Datenhaltung wurde ein Bücherregal entworfen, welches Informationen über den genauen Zustand der Dokumente bereithält.

3.4 Die Agenten von MoleOffice

In diesem Abschnitt sollen die Agenten des Systems und ihr Beitrag zur Erledigung der Gesamtaufgabe dargestellt werden.

Beim Entwurf des Systems stellte sich heraus, daß nicht nur neue Agenten hinzu kommen, sondern auch die schon vorhandenen erweitert werden mußten. Die Erweiterung der schon vorhandenen Agenten sollte allerdings so gering wie möglich bleiben und die Funktionalität in neue Agenten kommen.

Die folgende Abbildung verdeutlicht die zentrale Rolle des Agentensystems innerhalb von Mole-DDE. Die Farben veranschaulichen die Zugehörigkeit der Komponenten zu den unterschiedlichen Benutzern.

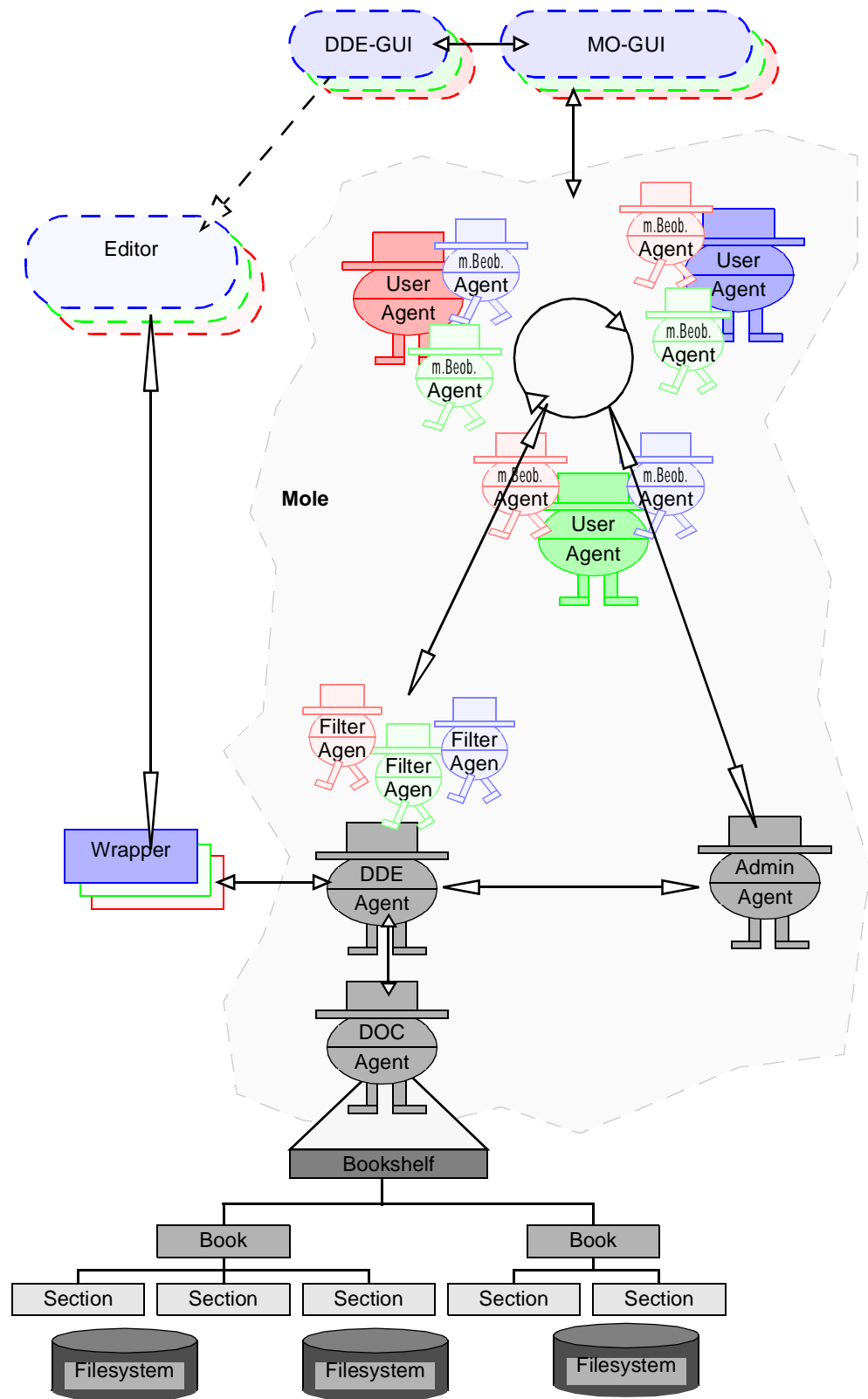


ABBILDUNG 10. Agenten-Struktur des DDE-Systems

Die folgenden Agentenklassen sind in Mole-DDE neu hinzugekommen:

- **DDE Agent :**

Der DDE Agent bildet die Schnittstelle zur externen Editorumgebung. Bei ihm melden sich die Editoren bzw. die Wrapper an. Er leitet die Zugriffe auf das Bücherregal koordiniert an den Dokumenten Agenten weiter und bekommt von diesem eine Rückmeldung über den Erfolg. Vom Administrator Agenten bekommt er die Benutzerinformationen, um die externen Informationen der Wrapper mit diesen abzugleichen. Er nimmt daraufhin eine Zuordnung der Wrapper zu den vorhandenen Benutzern vor und überprüft, ob diese angemeldet sind. Gibt der Dokumenten Agent eine positive Rückmeldung, generiert der DDE-Agent ein DDE-Aktions Objekt, welches alle Informationen über diese Aktion enthält. Diese Aktion schickt er sowohl dem entsprechenden Wrapper als auch dem zugehörigen mobilen Filteragenten.

- **Dokumenten Agent :**

Der Dokumenten Agent verwaltet alle Dokumente, koordiniert die Zugriffe auf sie und löst auftretende Inkonsistenzen verschiedener Replikate auf. Als zentrales Objekt enthält er das Bücherregal. Er selbst benötigt keine Informationen über die Editoren oder Wrapper. Auch die Darstellung der Informationen ist aus seiner Sichtweise unwichtig. Bei Zugriffen muß er die Dokumente sperren und freigeben können und dafür sorgen, daß die vorgenommenen Modifikationen korrekt gespeichert werden. Er stößt auch die periodische automatische Speicherung an.

- **mobiler DDE Filteragent :**

Ausgehend von jedem Benutzer Agenten wird nun ein zusätzlicher mobiler Agent zum DDE Agenten geschickt, um dort von auftretenden Ereignisse unterrichtet zu werden. Es liegt in seiner Verantwortung, die Informationen entsprechend seinen Filtern an andere Filteragenten und seinen Benutzer Agenten weiter zu leiten.

An den schon vorhandenen Agentenklassen¹ mußten lediglich kleinere Anpassungen vorgenommen werden:

- **Administrator Agent :**

Die zentrale Rolle spielt weiterhin der "Administrator Agent". Da auch der DDE Agent Veränderungen an den Benutzerinformationen vornehmen kann, besitzt dieser nun die Möglichkeit zur Synchronisation der Benutzerdaten mit dem DDE Agenten.

- **Benutzer Agent :**

Der Agent muß dahingehend erweitert werden, mit den neuen Agentenklassen zu kooperieren. Er nimmt die DDE-Aktionen von den DDE-Filteragenten entgegen und wertet sie aus beziehungsweise leitet die darin enthaltenen Informationen an die grafische Oberfläche weiter.

- **mobiler Beobachtungsagent :**

Diese Agentenklasse wird ohne Änderungen übernommen.

1. Zur genauen Beschreibung der Agenten siehe Kapitel 2.4.2 auf Seite 18

Jeder Agent hat demnach seine ganz persönliche Aufgabe und kein Agent darf auf das komplette System zugreifen. So kennt beispielsweise der Dokumenten Agent keine Benutzer, sondern nur Objekte, die seine Bücher und Kapitel anlegen, verändern oder löschen.



Innerhalb des gesamten DDE-Systems bilden die Agenten von Mole-DDE die zentrale Rolle.

3.5 Informationsaustausch

Für ein produktives Arbeiten ist es wichtig, ein effektives Kommunikationsmittel nutzen zu können. Um das gemeinsame Ziel zu erreichen, müssen sowohl synchron als auch asynchron Informationen ausgetauscht werden. Die Kommunikation kann hierbei zum einen implizit vom System vorgenommen werden. Man nimmt dabei im voraus eventuell Einstellungen vor. Zum anderen können aber auch Informationen explizit vom Benutzer generiert werden und beispielsweise per Email versandt werden.

3.5.1 Explizite Kommunikation

In *MOLEOFFICE* ist ein Konferenz-System vorgesehen, welches es ermöglicht Informationen mit mehreren Personen gleichzeitig über Audio und eventuell Video auszutauschen. Geplant wurde zudem ein Emailsysteem zu integrieren, bei welchem man nur die gewünschte Person innerhalb seines Kontextes auszuwählen braucht, um diesem per Email eine Nachricht zu hinterlassen. Beide Komponenten befinden sich bisher allerdings erst in der Planung.

3.5.2 Implizite Kommunikation

Wichtiger für diese Arbeit ist die implizite Kommunikation. Innerhalb von Mole-DDE wird man über die Aktivitäten der anderen Teilnehmer, bezüglich ihrer Arbeit an den Dokumenten informiert, so beispielsweise auch welche Kapitel gerade geöffnet sind.

In *MOLEOFFICE* sieht man, welche Aktivität ein Teilnehmer prinzipiell gerade vornimmt und dessen Kommunikationsbereitschaft. Beide Informationssysteme können

über die Konfiguration von Filtern, beispielsweise bezüglich ihres Informationsgehaltes, gesteuert werden.

3.5.3 Informationswege

Informationen entstehen in Mole-DDE an unterschiedlichen Quellen, ganz im Gegensatz zum ursprünglichen *MOLEOFFICE*, wo die Informationen ausschließlich an der grafischen Benutzeroberfläche entstanden.

Ändert man den Zustand seiner Person, oder die Filterregeln an der grafischen Benutzeroberfläche, werden diese Informationen über den entsprechenden Benutzeragent an die mobilen Beobachtungsagenten und an den Administratoragent weitergereicht.

Abbildung 11 verdeutlicht am Beispiel ausgehend vom Editor den Fluß der Information:

In *MOLE-DDE* entstehen Informationen vor allem auch im Editor. Dieser leitet sie an seinen Wrapper weiter, der die Informationen dann aufbereitet an den DDE Agenten reicht.

Der DDE Agent unternimmt hier schon eine erste Filterung und Aufspaltung der Informationen. Manche Daten des Wrappers, wie beispielsweise das Lesen des Bücherregals, müssen lediglich an den Dokumentenagent gereicht werden. Andere Informationen, wie die Anmeldeinformationen, braucht nur der DDE Agent selbst. Der Großteil der Informationen reicht der DDE Agent zur Bearbeitung an den Dokumentenagenten weiter.

Der Dokumenten Agent muß die Daten geeignet verarbeiten und entsprechende Daten zurück an den DDE Agenten senden, woraufhin der DDE Agent zum einen eine DDE-Aktion¹ für den zur Nachricht gehörenden Filter Agenten als auch eine Antwort direkt an den Wrapper generiert.

1. siehe Abschnitt 3.7

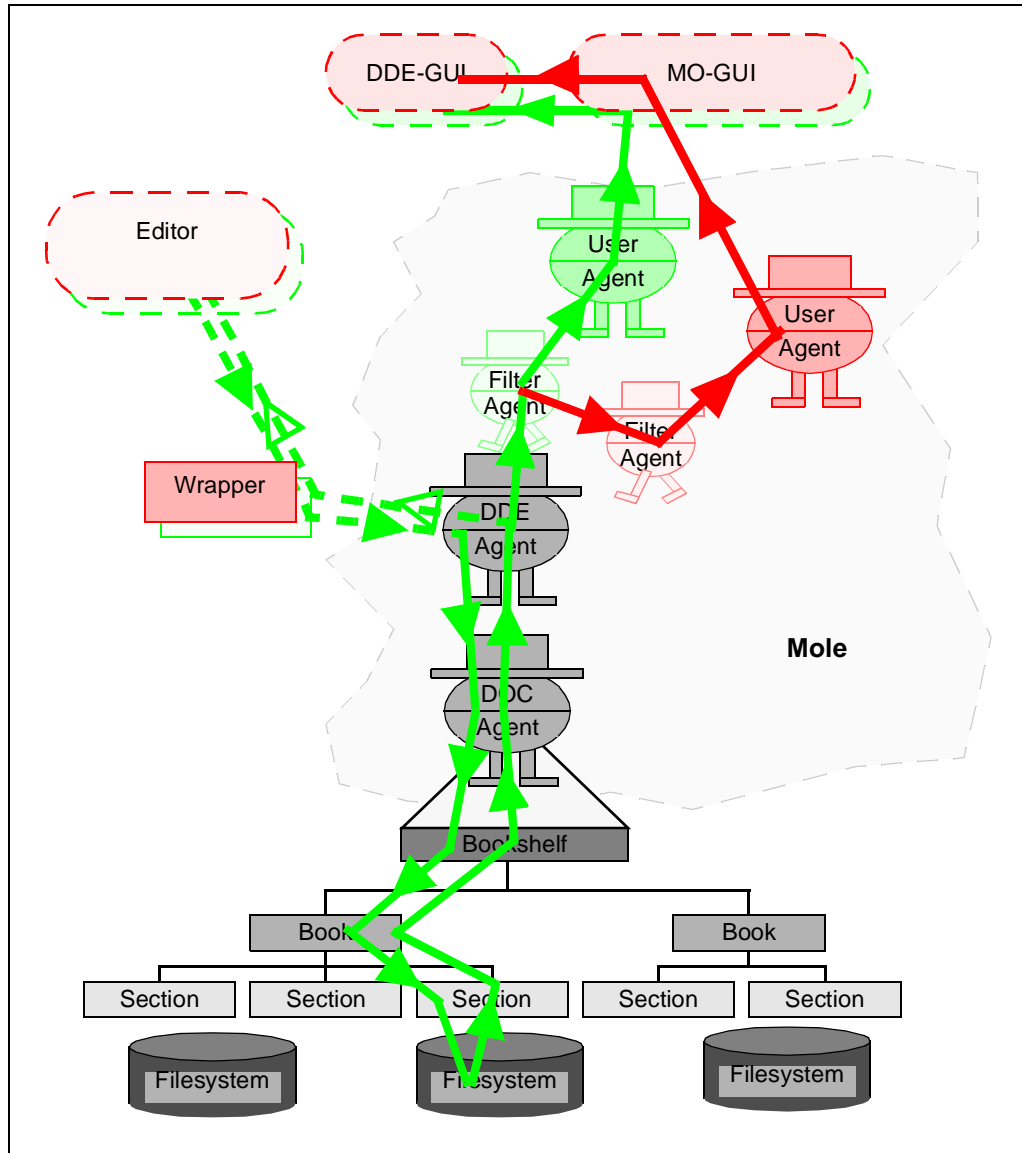


ABBILDUNG 11. Informationsfluß der Editor Daten

Die Filter Agenten leiten diese Informationen je nach Einstellung der Filter an ihre Benutzeragenten beziehungsweise an die anderen Filter Agenten weiter, wo sie schließlich zurück an die grafische Oberfläche gelangen.

Aber auch die erweiterte grafische Oberfläche ist nicht nur Datensinke, sondern auch Datenquelle. So werden Informationen über die Rollen, Zugriffsrechte und Filter von hier aus zum Benutzeragenten gereicht und gelangen von dort aus zum einen zu den DDE Filteragenten, zum anderen an den Dokumenten Agenten und den DDE Agenten.

3.6 Einsatz mobiler Agenten

Wie in Abschnitt 2.3 schon erläutert wurde, handelt es sich bei Mole um ein mobiles Agentensystem, was nicht heißt, daß alle vorhandenen Agenten unbedingt auch mobil sein müssen.

Mobile Agenten sind unter den folgenden Voraussetzungen besonders gut geeignet:

1. Die Daten werden vom mobilen Agenten selbst verarbeitet.
2. Die Daten werden gefiltert, beziehungsweise die Datenmenge wird durch den mobilen Agenten reduziert.
3. Die zu verarbeitende Datenmenge ist im Schnitt größer als das serialisierte Agentenobjekt.
4. Durch die Positionierung mehrerer Agenten an einer Lokation müssen Daten überhaupt nicht transportiert werden.
5. Es muß kein Zugriff auf Systemressourcen erfolgen.

Die Punkte 1 bis 5 hängen zum Teil eng miteinander zusammen, schließen sich jedoch auch teilweise gegenseitig aus. Insbesondere zur Datenverarbeitung braucht man häufig Zugriff auf Systemressourcen, beispielsweise wenn man Daten speichern will. Will ein mobiler Agent die empfangenen Daten nicht selbst verarbeiten sondern einem Systemagenten zu Bearbeitung weiterreichen, muß bedacht werden, daß durch die gehaltenen Daten auch das serialisierte Agentenobjekt um die Größe der gehaltenen Daten wächst.

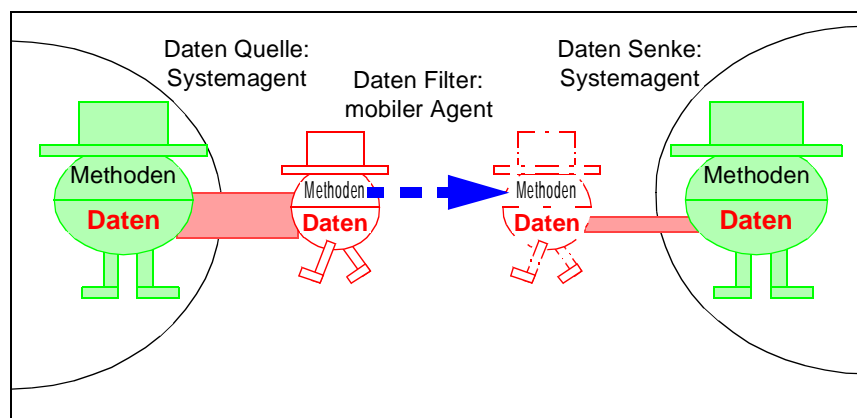


ABBILDUNG 12. Datenfilterung

Optimal scheint deshalb der Einsatz mobiler Agenten als Filtermechanismus, da hier die Datenmenge vor dem Weiterreichen normalerweise reduziert wird. In Mole-DDE existieren zwei verschiedene mobile Agentenklassen. Sowohl die Beobachtungsagen-

ten als auch die DDE Filteragenten reduzieren durch einen variablen Filtermechanismus die durch sie verlaufenden Datenströme.

3.7 Aktionen und Filter

Mole-DDE verwendet zum Austausch dokumentenspezifischer Informationen ein spezielles Objekt, in dem die folgenden für die Kommunikationspartner wichtigen Informationen über die Aktion beinhaltet sind:

- **Das betroffene Objekt der Aktion:**
Dies kann zum Beispiel ein Kapitel oder ein Buch sein.
- **Der Erzeuger der Aktion:**
In dem hier verwendeten Szenario ist das der Name seines Benutzeragenten.
- **Die erzeugte Aktion:**
Dies ist eine eindeutige Beschreibung der Aktion, wie zum Beispiel "Kapitel lesen"

Bisher existieren die folgenden Aktionen, die über ein DDE-Aktionsobjekt transportiert werden:

- **Buch erzeugen**
- **Buch löschen**
- **Kapitel erzeugen**
- **Kapitel löschen**
- **Kapitel öffnen**
- **Kapitel schließen**
- **Kapitel lesen**
- **Kapitel schreiben**

Das Bücherregal wird, falls es noch nicht existiert, automatisch vom Dokumenten Agenten erzeugt. Beim Öffnen und Schließen von Kapiteln handelt es sich um einen einmaligen Vorgang, wenn man ein Dokument bearbeiten will. Auch mehrmaliges Lesen und Schreiben öffnet und schließt das Dokument zwischenzeitlich nicht. Durch das Öffnen und Schließen wird der Arbeitsbereich des entsprechenden Dokuments also betreten oder verlassen, d.h. es wird festgestellt, ob bei gegebener Situation ein Zugriff

erlaubt ist oder nicht. Dabei können verschiedene Aspekte bei der Entscheidungsfindung eine Rolle spielen:

- **Besitzer des Kapitels**
- **Zustand des Kapitels** (geöffnet, geschlossen)
- **Rolle des Zugreifers** (Verfasser, Kommentator, Teamleiter, ...)

Folgende weitere Aktionen sind noch möglich, die allerdings nur zwischen Wrapper, DDE Agent und Dokumentenagent gültig sind und von denen niemand sonst unterrichtet werden muß:

- **Inhaltsverzeichnis des Bücherregals lesen**
- **Inhaltsverzeichnis eines Buches lesen**
- **Anmeldung an das System**
- **Abmeldung vom System**

Indem ein Benutzer den Inhalt eines Inhaltsverzeichnisses im Editor sieht, erkennt er automatisch den Erfolg seiner Aktion. Anfragen nach dem Inhaltsverzeichnis werden auch stets erfolgreich ausgeführt¹.

Für zukünftige Erweiterungen sind die folgenden Aktionen in Mole-DDE geplant:


- **Kapitel dem Bücherregal entnehmen und lokal ablegen,**
- **lokales Kapitel in das Bücheregal einfügen,**
- **temporäre, zeitlich bedingte Abmeldung und**
- **eine erneute Anmeldung**

In Corba-DDE ist es beispielsweise möglich, ein Kapitel aus dem Bücherregal zu entnehmen und es später selbst wieder in das Projekt einzufügen.

Beim Transport der Aktionen kommt der in Abschnitt 3.5 beschriebene Filtermechanismus zum Tragen. Es werden also nicht alle Aktionen an alle beteiligten Benutzeragenten weitergereicht. Da der DDE Agent nur dem zur Aktion zugehörigen DDE Filteragent die Aktion weiterreicht, liegt die Verantwortung über den Fortgang der

1. Lediglich im Falle einer mißlungenen Anmeldung kann es nicht gelesen werden.

Aktion bei dessen Filtern. Auch steht bei starker Belastung des DDE Agenten dieser schneller wieder für neue Aktionen zur Verfügung.

 **Der Transport dokumentenspezifischer Informationen geschieht über ein spezielles, filterbares Aktionsobjekt.**

3.8 Sicherheit und Benutzerverwaltung

Um die Sicherheit der Daten in Mole-DDE zu gewährleisten, ist ein Authentifizierungsmechanismus unabdingbar. Dieser benötigt natürlich seinerseits eine Benutzerverwaltung. Im bisherigen MoleOffice existierte schon ein Feld für das Passwort im Benutzerobjekt, allerdings wurde es seither nicht eingesetzt. Auch eine Benutzerverwaltung fehlte völlig, so daß die Informationen direkt in einem Java-Quellprogramm verändert wurden.

In einer zum Teil parallel verlaufenden Arbeit¹ verwendet Alexander Kramer ebenfalls das Passwortfeld zur Authentifizierung mobiler Benutzer über das IRDA-Protokoll. Die Passwortüberprüfung geschieht hier zwischen dem Benutzeragenten und seinem Äquivalent im mobilen Endgerät. Bisher fehlt allerdings die Möglichkeit, das Passwort zu setzen oder zu verändern.

3.8.1 Benutzerverwaltung

Innerhalb der Konfigurationswerkzeuge für Mole-DDE entstand eine Benutzerverwaltung, ähnlich dem unter Unix bekannten Verfahren einer Passwort-Datei. Die Passwörter werden allerdings nicht in der Passwort-Datei selbst oder einer extra "Shadow-Datei" abgelegt, sondern liegen innerhalb des Benutzerobjekts beim Systemagenten.

Die Passwortdatenbank speichert neben dem Namen des Benutzers auch Namen und Ort seines Benutzeragenten. Es können hier auch mehrere Einträge für einzelne Benutzer mit unterschiedlichen Konfigurationen existieren.

1. Benachrichtigung von Teamkollegen über Erreichbarkeit mittels mobiler Geräte

Änderungen an der Datei werden allerdings erst nach einem Neustart des MoleOffice Servers aktiviert. Der Benutzeragent ist als Systemagent nicht mobil und kann deshalb bei einer Änderung seiner Lokation den Ort nicht wechseln. Änderungen am Anmeldenamen sind nicht sinnvoll, da der Name im System als Referenz dient und die persönlichen Einstellungen, wie der angezeigte Name, unabhängig von diesem frei gewählt werden können.

Das Passwort kann hingegen während des Betriebs über eine grafische Oberfläche beliebig verändert werden und ist sofort aktiv. Durch die Verwendung des gleichen Passwortfeldes in beiden Projekten wurde eine Zusammenführung der Projekte ermöglicht.

3.8.2 Authentifizierung

Die Authentifizierung erfolgt beim Start der grafischen Oberfläche. Der Benutzer wird über ein Fenster nach seinem Passwort gefragt, welches innerhalb des *MOLEOFFICE*-Klienten direkt überprüft wird. Falls keine Übereinstimmung vorliegt, wird die Oberfläche sofort wieder beendet¹ und es kann auch keine Anmeldung des Editors vorgenommen werden. Damit auch der DDE Agent weiß, ob ein Benutzer angemeldet ist oder nicht, wird ein Zugangsfeld innerhalb der Benutzerkomponente bei erfolgreicher Anmeldung gesetzt und beim Schließen der Benutzungsoberfläche wieder gelöscht.

3.8.3 Verschlüsselung

Normalerweise sollten alle Passwörter sowohl verschlüsselt gespeichert als auch verschlüsselt übertragen werden. Da die Kommunikation der Agenten bisher keine verschlüsselte Übertragung ermöglicht, bleibt nur eine verschlüsselte Speicherung.

1. Bei einer verschlüsselten Datenübertragung wäre eine dreimalige Eingabemöglichkeit des Passwortes vorzuziehen.

Um kompatibel zum Projekt mobiler Endgeräte zu bleiben wurde darauf allerdings verzichtet, so daß bisher nur eine geringe Sicherheitsstufe vorliegt.

 **Durch den Einsatz von Benutzerverwaltung und Authentifizierungsmechanismen wurde ein grundlegender Zugriffsschutz entworfen.**

3.9 Die Editor Umgebung

Bei den klassischen Mehrbenutzereditoren steht normalerweise eine integrierte Umgebung zur Verfügung, bestehend aus der eigentlichen Komponente zum Schreiben, Feldern für Zustandsinformationen, Einstellungsmöglichkeiten für Rollen, Rechte, Filter und persönlichen Einstellungen etc. Bei der verteilten Dokumentenbearbeitung mit erweiterten Einbenutzereditoren ist dies jedoch nicht zwingend notwendig. Je nachdem wie gut sich die Editoren erweitern lassen beziehungsweise welche Funktionalität schon im voraus in eine separate Benutzungsoberfläche aufgenommen wurde, sind viele Funktionen gar nicht im eigentlichen Editor vorhanden.

3.9.1 Bedeutung von Einbenutzereditoren in MoleDDE

Die Motivation zum Einsatz von Einbenutzereditoren auf der Basis einer zentralen (oder verteilten) Kommunikations- und Verwaltungsarchitektur ergibt sich aus der Individualität der verschiedenen Editoren. So wird jedem Benutzer ermöglicht, mit dem Editor seiner Wahl weiterzuarbeiten. Eine Erweiterung der Funktionalität der Editoren ist dennoch notwendig: Von sich aus ist keiner der verfügbaren Editoren in der Lage, direkt mit dem MoleDDE zu kooperieren.

Bei der Erweiterung ergeben sich zwei verschiedene Ansatzmöglichkeiten.

1. Zum einen kann man versuchen, in jeden Editor - ähnlich den Mehrbenutzereditoren - die volle Funktionalität zu bringen.
2. Als zweite Möglichkeit bietet sich eine zusätzliche Komponente an, die möglichst viele Funktionen vereint und zum Ziel hat, eine Erweiterung der Editoren möglichst gering zu halten.

Im ersten Fall unterscheidet sich der Einbenutzereditor kaum von einem Mehrbenutzereditor. Dieser Ansatz bietet jedoch die Möglichkeit, alle Arbeiten unter einer Oberfläche vollbringen zu können. Die zweite Variante erlaubt hingegen eine wesentlich einfachere Integration zusätzlicher Editoren, da hier nur einmal der Entwicklungsaufwand für die DDE-Funktionalität anfällt.

3.9.2 Begriffsbestimmungen

Im Corba-DDE wurde der Editor Emacs dahingehend erweitert, über einen Wrapper mit dem eigentlichen DDE-Projekt zu interagieren.

Aus Gründen der Lesbarkeit werden für dieses Kapitel folgende begrifflichen Konventionen getroffen:

- **GNU-Emacs :**
Dieser unter der GNU General Public License entwickelter Editor ist frei verfügbar und beliebig erweiterbar, sofern die Veränderungen stets unter der GPL bleiben.
- **XEmacs :**
Ausgehend vom “GNU Emacs Version 19” wurde ein weiterer Editor mit dem Namen “Lucid Emacs” von Lucid Inc., Sun Microsystems und der Universität von Illinois entwickelt. Dieser wurde später auch XEmacs genannt, da er über eine grafische Unterstützung durch Maus Knöpfe und Menüleisten verfügt. Er wurde in der aktuellen Version 21.1 verwendet. Der XEmacs ist ebenfalls frei erweiterbar.
- **Emacs :**
Dies bezeichnet einen allgemeinen Begriff, der stellvertretend für alle Emacs Versionen steht. Er wird immer dann verwendet, wenn entweder alle kompatiblen Versionen gemeint sind, oder es keine Rolle spielt, welcher gemeint ist.

3.9.3 Der Emacs in Mole-DDE

Im Gegensatz zu Corba-DDE existiert in Mole-DDE sehr wohl eine grafische Benutzungsoberfläche außerhalb des Emacs, die große Teile der Funktionalität übernehmen kann. Es erfolgt also eine Implementierung der Variante zwei aus Abschnitt 3.9.1 . Lediglich die klassischen Funktionen eines Editors sind unbedingt notwendig.

Innerhalb des Editorbereichs wird man nur über seine eigenen Aktionen unterrichtet. Da die Erweiterungen am Emacs keine informationsgesteuerte Kommunikation mit Rückmeldungen erlaubt, muß der Benutzer über den Erfolg oder Mißerfolg direkt unterrichtet werden und selbst agieren. Falls beispielsweise das Öffnen eines Kapitels

fehlschlägt, wechselt der Emacs trotzdem in den Kapitelmodus. Der Benutzer bekommt lediglich eine Mitteilung, daß das Kapitel nicht geöffnet werden kann und der Inhalt des Kapitels besteht aus einer Warnung, es nicht zu benutzen. Der Benutzer muß nun selbst dafür sorgen, wieder zurück in den Büchermodus zu wechseln. Alle darüber hinaus gehenden Informationen wurden aus dem Editor herausgenommen und in die neue grafische Oberfläche integriert.

Die Steuerung des Emacs wurde prinzipiell von Corba-DDE übernommen. Es fand lediglich eine Anpassung an neu hinzugekommene und noch nicht vorhandene Funktionen in den verschiedenen Modi statt. Die folgenden Modi bietet der Emacs unter Mole-DDE an:

- **Bücherregalmodus :**

Es wird das aktuelle Inhaltsverzeichnis des Bücherregals angezeigt, d.h. alle vorhandenen Bücher werden aufgelistet

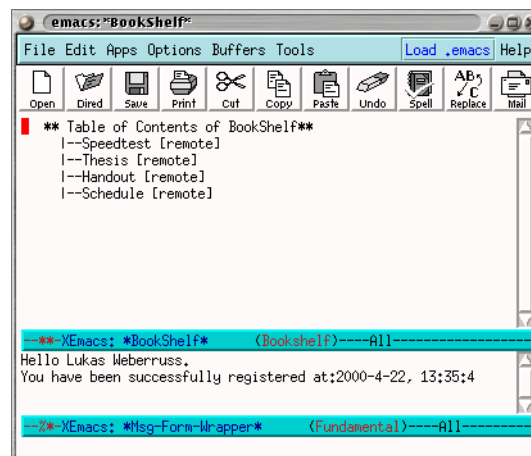


ABBILDUNG 13. Bücherregalmodus im XEmacs

Durch Auswahl eines Buches gelangt man in den Büchermodus. Die Tabelle beschreibt, welche Kommandos in diesem Modus existieren.

Kommando	Befehl
<CTRL>+<x>-<CTRL>+<r> <m> oder <M>	Bücherregal aktualisieren Buch markieren
<c> oder <C> <d> oder <D>	neues Buch erstellen Buch löschen
<u> oder <U> <DOWN>	Buch demarkieren nächste Zeile
<UP> <ENTER>	vorherige Zeile Buch öffnen
<CTRL>+<x>-<CTRL>+<c> oder <CTRL>+<x>-<CTRL>+<k>	Abmelden und Emacs beenden

Tabelle 1. Bücherregal Kommandos

Alle anderen unter dem Emacs üblichen Kommandos sind bis auf wenige Ausnahmen im Bücherregalmodus nicht erlaubt, da dieser Modus nicht zur Eingabe von Daten dient, sondern nur das Inhaltsverzeichnis zeigt und die Möglichkeit bietet in den Büchermodus des entsprechenden Buches zu wechseln. Eine direkte Auswahl der Bücher per Maus ist nicht möglich.

- **Büchermodus :**

Die einzelnen Kapitel des entsprechenden Buches werden der Reihenfolge nach aufgelistet. Es kann entweder zurück in den Bücherregalmodus gewechselt werden oder durch die Auswahl eines Kapitels in den Kapitelmodus.

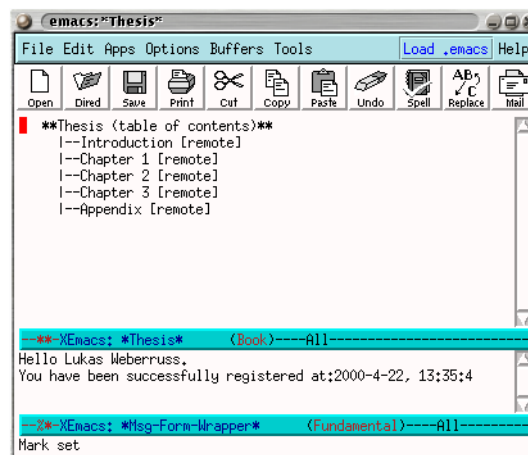


ABBILDUNG 14. Büchermodus im XEmacs

Auch hier gibt es speziell für den Büchermodus angepaßte Tastaturkommandos, die an die unter dem Emacs sonst üblichen Kommandos angelehnt wurden:

Kommando	Befehl
<CTRL>+<x>-<CTRL>+<r>	Buch aktualisieren
<m> oder <M>	Kapitel markieren
<c> oder <C>	neues Kapitel erstellen
<d> oder <D>	Kapitel löschen
<u> oder <U>	Kapitel demarkieren
<DOWN>	nächste Zeile
<UP>	vorherige Zeile
<ENTER>	Kapitel öffnen
<CTRL>+<x>-<CTRL>+<k>	Kapitel schließen
<CTRL>+<x>-<CTRL>+<c>	Abmelden und Emacs beenden

Tabelle 2. Bücher Kommandos

Es werden hier ebenfalls kaum zusätzliche für den Emacs typische Kommandos unterstützt, da analog zum Bücherregal keine direkte Manipulation der Daten möglich ist.

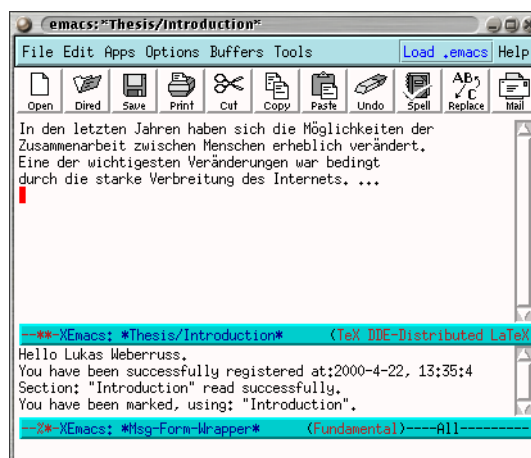


ABBILDUNG 15. Kapitelmodus im XEmacs

- **Kapitelmodus:**

Zusätzlich zum Kapitelmodus wird bisher ein Latexmodus gestartet, so daß wie gewohnt die Textbearbeitung mit den Latexerweiterungen des Emacs vorgenommen werden kann. Durch Verlassen des Kapitelmodus wechselt man automatisch wieder in das übergeordnete Buch.

Im Gegensatz zu den beiden anderen Modi wird hier nicht ein Inhaltsverzeichnis zur Auswahl eines Dokuments gezeigt, sondern der Inhalt selbst zur Bearbeitung geöffnet. Die folgenden Kommandos sind speziell zum Arbeiten mit dem DDE-System entworfen und ersetzen die zum Teil äquivalenten Gegenstücke des normalen Emacs:

Kommando	Befehl
<CTRL>+<x>-<CTRL>+<r>	Kapitel aktualisieren
<CTRL>+<x>-<CTRL>+<f>	Kapitel öffnen
<CTRL>+<x>-<CTRL>+<k>	Kapitel schließen und Büchermodus öffnen
<CTRL>+<x>-<CTRL>+<s>	Kapitel speichern
<CTRL>+<x> <CTRL>+<c>	Kapitel schließen, Abmelden und Emacs beenden

Tabelle 3. Kapitel Kommandos

Hinzu kommen die gewohnten, nicht überschriebenen Emacs Kommandos, einschließlich der für den *LATEX*-Modus üblichen.



Der Emacs stellt durch seine freie Lizenzpolitik und der Möglichkeit zur Programmierung eigener Module einen hervorragenden Editor zur Integration in ein DDE System dar.

3.10 Wrapper

Um einen Editor an das DDE System anzuschließen, bedarf es *eventuell* einer weiteren Komponente, die gegebenenfalls eine Umsetzung der Schnittstellen bzw. des Informationsflusses vornimmt.

3.10.1 Definition

Es gibt unterschiedliche Möglichkeiten der Definition eines Wrappers. Als gemeinsame Grundlage gilt jedoch: Als Wrapper bezeichnet man eine Komponente, die als Schnittstelle zwischen einem Editor und einem Verteilten Dokumentensystem dient.

Der Umfang eines Wrappers kann sehr stark variieren, so daß eine der beiden folgenden Definitionen die grundlegende Definition ergänzt:

1. Als Wrapper bezeichnet man nur die zur Schnittstelle gehörenden, zusätzlichen Komponenten, nicht aber die Anpassungen am Editor.
2. Als Wrapper bezeichnet man alle Erweiterungen - sowohl am Editor, als auch an zusätzlichen Komponenten, die zur Integration des Editors nötig sind.

Die erste Variante erscheint zuerst geeigneter, da hier der Wrapper eine separate Komponente darstellt. Jedoch ist es extrem schwer zu entscheiden, was alles (vom Editor aus gesehen) *extern* ist.

In dieser Arbeit wird deshalb anstatt dessen die folgende Erweiterung verwendet:

3. Als Wrapper bezeichnet man alle in einem separaten Thread laufenden Komponenten, die im normalen unmodifizierten Editor nicht vorhanden sind.

Nach dieser Definition gehören alle Module, die den Editor erweitern und in dessen Umgebung laufen, nicht zum Wrapper.

3.10.2 Überblick

Nicht alle Editoren eignen sich gleich gut für Erweiterungen hinsichtlich eines DDE-Systems. Der im vorherigen Abschnitt vorgestellte Emacs kann beispielsweise extrem gut programmiert werden. Normalerweise möchte man die für die DDE-Funktionalität nötigen Erweiterungen am Editor möglichst modular, mit verschiedenen DDE-Systemen, einsetzen können. Aus diesem Grund führt man als Zwischenebene den Wrapper ein, der zur Aufgabe hat, die verschiedenen Schnittstellen von Editor und DDE-System zu überbrücken.

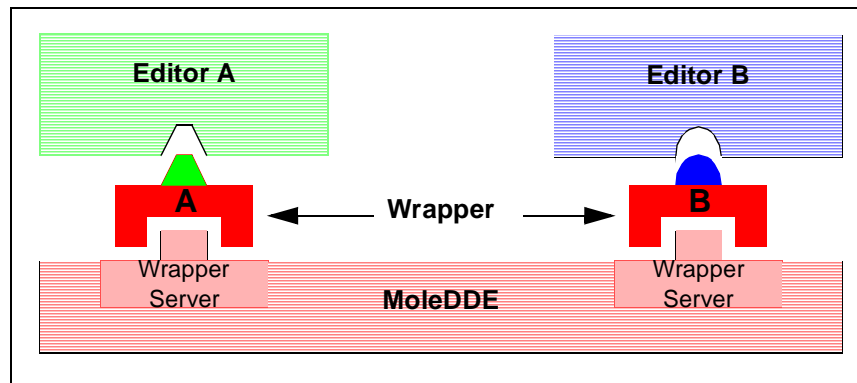


ABBILDUNG 16. Der Wrapper als Schnittstelle

Abbildung 16 verdeutlicht, wie das Szenario bei Mole-DDE aussieht. Über einen Wrapper Server können sich speziell für den Editor angepasste Wrapper anmelden. Falls der Editor eine geeignete Schnittstelle besitzt, ist es sogar möglich, daß sich diese direkt, ohne den Einsatz eines Wrappers, beim Wrapper Server anmelden.

3.10.3 Informationsumsetzung

Da nicht alle Editoren die vom DDE System angebotene Funktionalität vollständig unterstützen oder umgekehrt die Funktionalität der Editoren mächtiger ist als die des DDE Systems, ist es notwendig, beim Informationsfluß durch den Wrapper Funktionen zu manipulieren. Zusätzlich müssen noch die Informationen von der einen zur anderen Schnittstelle umgesetzt werden.

Die Manipulation der Funktionen kann auf folgende Weise erfolgen:

- Entfernen von nicht vorhandener Funktionalität.
- Hinzufügen von Funktionalität durch Simulation oder Verwendung von Standardergebnissen.

Die Umsetzung ist sowohl für die Richtung Editor nach DDE als auch in umgekehrter Richtung gültig.

Als Beispiel in der Richtung Editor nach DDE könnte man sich vorstellen, daß der Editor vor dem Lesen eines Kapitels nicht versucht, dieses zu öffnen, beziehungsweise beim Verlassen des Kapitels dieses nicht mehr schließt. Der Wrapper kann dann vor jeder Leseanfrage des Editors selbst versuchen, das entsprechende Kapitel zu öffnen beziehungsweise nach jedem Schreibauftrag das Kapitel automatisch schließen.

In umgekehrter Richtung könnte man sich vorstellen, daß der Editor mehrere Bücherregale unterstützt. Der Wrapper würde in diesem Fall die unterschiedlichen Bücherregale auf *das* Bücherregal in Mole-DDE abbilden.

3.10.4 Emacs Wrapper

Da von Anfang an davon ausgegangen wurde, daß als Testobjekt für Mole-DDE der Emacs in ähnlicher Form weiterverwendet wird, empfahl es sich, eine ähnliche Schnittstelle zwischen Emacs und Wrapper bzw. zwischen Mole-DDE und Wrapper zu entwerfen. Durch diese Entscheidung konnte die Funktionalität des Wrappers stark herabgesetzt werden. Der Emacs enthält in seinen Modulen einen Großteil der Funktionalität selbst. Lediglich die von Mole-DDE angebotenen Funktionen zum Sortieren der Bücher und Kapitel¹ existieren nicht.

1. siehe Kapitel 3.3 auf Seite 27

Es ist also theoretisch möglich, den Emacs direkt und ohne den Einsatz eines Wrappers an das DDE System zu koppeln. Beim “harten” Beenden des Emacs kann es dann allerdings zu Komplikationen kommen. Empfehlenswert ist diese Vorgehensweise nicht, da im Wrapper eventuell Optimierungen vorgenommen werden. So wäre es beispielsweise denkbar, daß der Wrapper als “Proxy¹” mit integriertem Cache arbeitet. Er kann Informationen, die früher schon abgefragt wurden, direkt an den Editor weitergeben. Damit die Daten trotzdem konsistent bleiben, muß der Wrapper eine Anfrage auf Modifikation beim DDE System stellen und nur dann eine Übertragung veranlassen, falls sich Informationen geändert haben.

Durch diese Maßnahme kann zum einen der Datendurchsatz erhöht werden und zum anderen Mole-DDE entlastet werden.



Ein Wrapper ermöglicht durch spezielle Funktionen und Schnittstellen den Einsatz unterschiedlicher Editoren.

3.11 Schnittstellen

Wie in Abschnitt 3.10 geschildert, existieren sowohl zwischen Emacs und Wrapper Schnittstellen, als auch zwischen Wrapper und Mole-DDE.

Die Schnittstelle zwischen Emacs und Wrapper stellt nur ein beliebiges Beispiel zwischen einem Editor und dem Wrapper dar, wohingegen die Schnittstelle zwischen Mole-DDE und Wrapper einen Standard zur Kommunikation zwischen einem Wrapper und Mole-DDE definiert, den man bei der Integration weiterer Editoren einhalten muß.

3.11.1 Schnittstelle zwischen Mole-DDE und Wrapper

Wie schon in Abschnitt 3.7 erwähnt, arbeitet Mole-DDE intern mit sogenannten DDE-Aktionen zur Kommunikation zwischen den beteiligten Komponenten. Beim Versenden dieser Informationen müssen diese vom DDE Agenten in Informationseinheiten

1. Der Begriff bedeutet eigentlich “Bevollmächtigter” und wird vor allem im World Wide Web verwendet.

(IE) transformiert werden. Als hereinkommende Nachrichten werden ebenfalls Informationseinheiten erwartet, die wiederum in DDE-Aktionen umgewandelt werden.

Abbildung 17 zeigt das für die Informationseinheiten verwendete Format. Momentan sind die zur Trennung einzelner Blöcke notwendigen Sonderzeichen "<" und ">" noch nicht als normale Zeichen zugelassen.

```

<IE> ::= "<" <ID> "><" <CMD> "><"
        <CONTENT> ">" <NEWLINE>

<ID> ::= <DDE2WRAPPER> | <WRAPPER2DDE>

<DDE2WRAPPER> ::= "BookShelf" | "Minibuffer" | <OBJECT>

<WRAPPER2DDE> ::= "0" | "BookShelf" | <OBJECT>

<OBJECT> ::= <BOOKNAME> |
            <BOOKNAME> "/" <SECTIONNAME>

<BOOKNAME> ::= <BOOKNAME> <ZEICHEN> | <ZEICHEN>

<SECTIONNAME> ::= <SECTIONNAME> <ZEICHEN> | <ZEICHEN>

<CMD> ::= msg |
        init | close | bookShelfToc |
        bookToc | openBook | closeBook |
        createBook | destroyBook |
        openSection | closeSection |
        readSection | writeSection |
        createSection | destroySection

<CONTENT> ::= <CONTENT> | <CONTENT> <ZEICHEN>

<NEWLINE> ::= ASCII Zeichen 10

<ZEICHEN> ::= ASCII Zeichen 32-127 außer 60,62

```

Anmerkung:

Die genaue Definition der einzelnen Kommandos von <CMD> können der Datei *dde/wrapper/IE.java* entnommen werden .

ABBILDUNG 17. Definition einer IE in BNF

Um den Inhalt des Kapitels "Einleitung" vom Buch "Diplomarbeit" zu schreiben, kann beispielsweise die folgende, durch ein Neue-Zeile-Zeichen abgeschlossene IE verwendet werden:

<Diplomarbeit/Einleitung><E2WWRITE><Dies ist die Einleitung>

3.11.2 Schnittstelle zwischen Emacs und Wrapper

Um den Wrapper nicht unnötig mit einer weiteren Transformation des Informationsflusses zu belasten, haben die Informationseinheiten dieselbe Syntax wie die im vorherigen Abschnitt dargestellte.

3.11.3 Kommunikationsprotokoll

Bei der Kommunikation zwischen Mole-DDE, Wrapper und Emacs ist neben der Syntax der Informationseinheiten auch ein ganz spezielles Protokoll einzuhalten. Es beschreibt, in welcher Reihenfolge und unter welchen Bedingungen die Informationseinheiten eingesetzt werden dürfen.

Die Kommunikation zwischen Emacs und Mole-DDE ist dabei nach dem Klient / Server Modell konzipiert. Mole-DDE stellt den Server dar. Von seiner Sicht aus existieren beliebig viele Wrapper als Klienten. Für den Emacs als Klient arbeitet der Wrapper hingegen transparent. Er kommuniziert quasi direkt mit dem Server. Somit ist der Wrapper in Wirklichkeit Klient und Server.

Die Initiative geht üblicherweise vom Emacs aus. Dieser meldet sich beim Start über den Wrapper bei Mole-DDE an und kann im folgenden mit diesem fast beliebig kommunizieren. Beim Beenden meldet er sich schließlich auf dem gleichen Weg bei Mole-DDE wieder ab. Geschieht dies nicht, sorgt ein spezieller Mechanismus dafür, daß nach einer gewissen Zeit die Verbindung automatisch unterbrochen wird.

Abbildung 18 beschreibt die Kommunikationsmöglichkeiten aus der Sicht vom Emacs. Die gestrichelten Linien sollen verdeutlichen, daß es zwar nicht möglich ist, ein nicht geöffnetes Buch zu lesen oder zu beschreiben, aber zwischenzeitlich andere Kommunikationen, wie das Öffnen eines weiteren Kapitels, durchaus durchführbar sind. Beim Abmelden werden automatisch alle noch geöffneten Kapitel geschlossen.

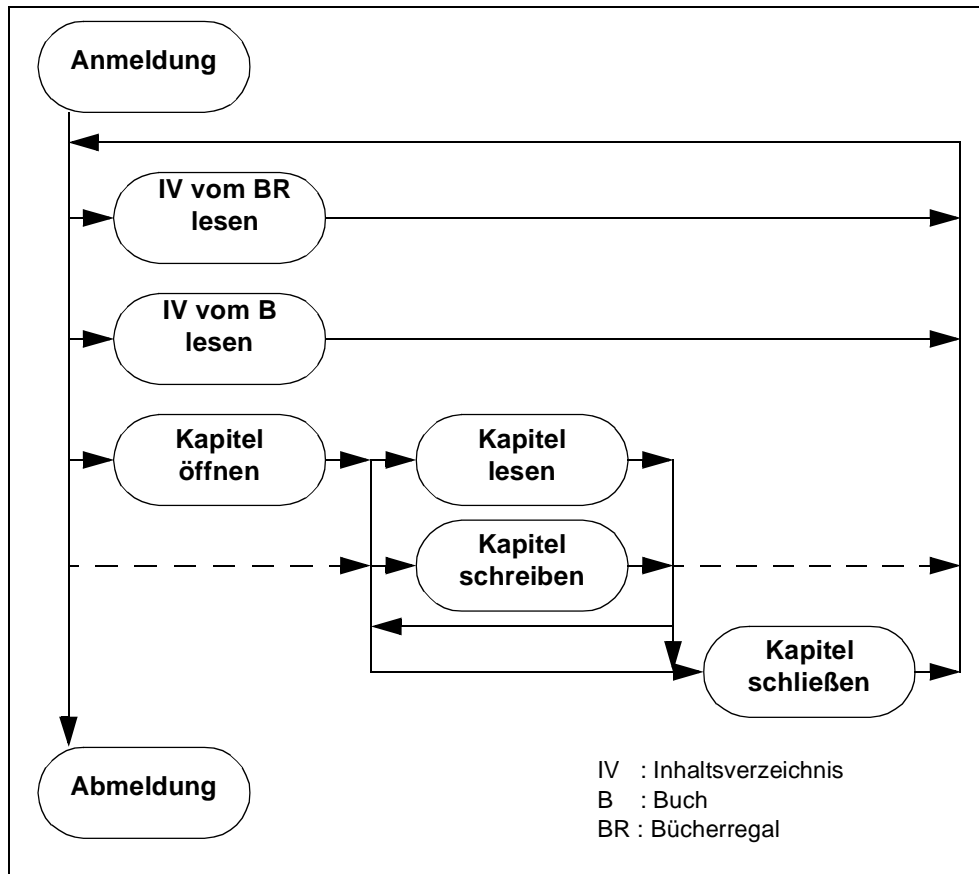


ABBILDUNG 18. Kommunikationsprotokoll zwischen MoleDDE und Emacs

Die Kommunikation zwischen Emacs, Wrapper und Mole-DDE kann zum Teil parallel erfolgen. Der Emacs fordert beispielsweise direkt nach dem Versenden der Anmeldeinformationen den Inhalt des Bücherregals an, noch bevor er eine positive Rückmeldung von MoleDDE bekommt. Abbildung 19 beschreibt einen Teil einer solchen Sitzung an einem Beispiel.

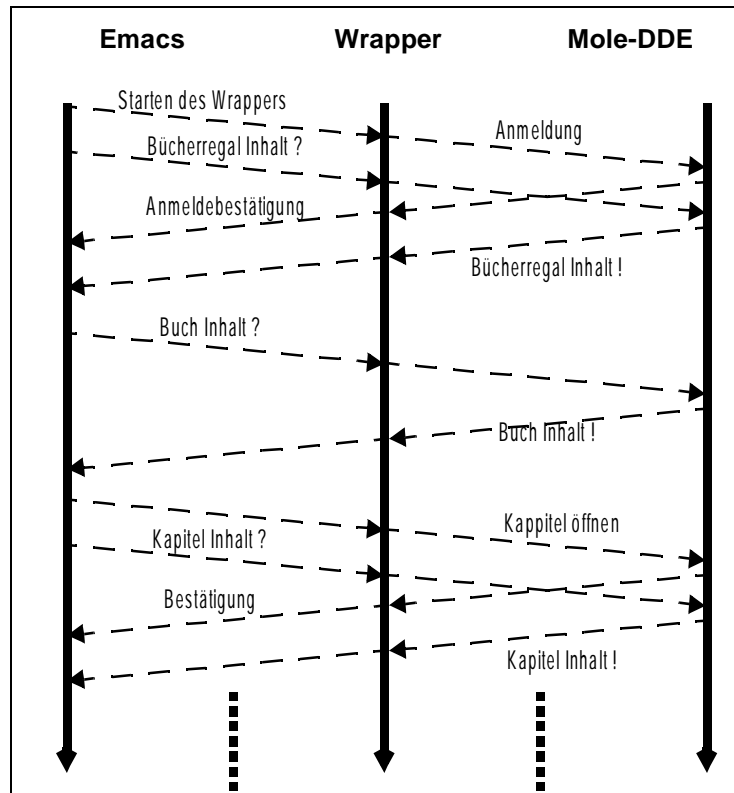


ABBILDUNG 19. Beispiel eines Sitzungsprotokolls

Der Emacs startet den Wrapper und schickt direkt danach eine Anfrage nach dem Inhalt des Bücherregals ab. Mole-DDE speichert die ankommenden Anfragen in einer Warteschlange und arbeitet sie sequentiell ab. Er meldet also den Wrapper zuerst bei sich an, schickt eine Anfrage nach dem Inhalt des Bücherregals an den Dokumentenagenten und schickt die Antwort schließlich zurück an den Wrapper.

Beim Editor kommen hintereinander zwei Nachrichten an. Die erste enthält die Anmeldebestätigung und die zweite den Inhalt des Bücherregals. Nachdem der Benutzer ein Buch ausgewählt hat, schickt der Editor auch hier eine Anfrage nach dem Inhaltsverzeichnis an Mole-DDE, welcher analog zur Abfrage nach dem Inhaltsverzeichnis des Bücherregals verfährt.

Nach der Auswahl eines Kapitels sendet der Editor zwei Anfragen los. Die erste ist die Aufforderung, das Kapitel zu öffnen; die zweite, es zu lesen. Auch hier legt Mole-DDE die Anfragen in einem Puffer ab, um sie sequentiell bearbeiten zu können.

Durch den zur Kommunikation zwischen Editor, Wrapper und Mole-DDE gewählten Mechanismus kann es nicht vorkommen, daß sich Nachrichten überholen. Es kann

allerdings vorkommen, daß durch Netzwerkfehler Nachrichten verlorengehen oder nur noch fehlerhaft am anderen Ende ankommen.

 **Das Kommunikationsprotokoll beschreibt welche Informationen in welcher Reihenfolge zwischen Editor, Wrapper und Mole-DDE ausgetauscht werden können.**

3.12 Die Benutzungsoberfläche

Zur Bedienung der verschiedenen Funktionen eines Verteilten Dokumentensystems eignet sich eine grafische Benutzungsoberfläche besonders gut. Auch unter MoleOffice konnten schon alle Einstellungen mit Hilfe grafischer Elemente wie Formulare und Knöpfe vorgenommen werden.

Ziel einer grafischen Benutzungsoberfläche muß stets die Möglichkeit einer einfachen, intuitiven Benutzung sein. Nur so erlangt ein Verteiltes Dokumentensystem auch die zum breiten Einsatz nötige Akzeptanz.

3.12.1 Überblick

Die in dieser Arbeit neu hinzugefügte Funktionalität eines Verteilten Dokumentensystems wurde soweit wie möglich in separate Komponenten ausgegliedert. Die von MoleOffice gewohnte Oberfläche wurde deshalb an möglichst wenigen Stellen verändert.

Das Applet von MoleOffice soll weiterhin die Funktion des Benachrichtigungsdienstes haben, d.h. es soll alle hierfür benötigten Komponenten beinhalten, darüberhinaus aber nicht durch DDE-Funktionen zu sehr an Komplexität und Umfang gewinnen. Zudem wurde es komplett durch die grafischen Elemente des AWT (Abstract Window Toolkit) entworfen, die neuen Komponenten sollten hingegen aus Elementen von Swing bestehen.

Dem Appletfenster wurde die Aufgabe der zentralen Hauptkomponente zugeordnet, so daß sich von hier aus alle anderen Funktionen, die nicht dem Benachrichtigungsdienst zuzuordnen sind, starten lassen.

3.12.2 Das MoleOffice Applet

Das Applet-Fenster von MoleOffice besteht aus drei horizontal getrennten Bereichen. Oben befinden sich Knöpfe, um zur Konfiguration oder zu Mole-DDE zu gelangen beziehungsweise einen Informationsabgleich vorzunehmen. Zudem existieren eine Auswahlliste für den aktuellen Raum und Radioknöpfe, um die Kommunikationsbereitschaft und die aktiven Tätigkeiten festzulegen.

Der mittlere Bereich dient dazu, den Zustand der Teamkollegen anzuzeigen. Dabei wird die Kommunikationsbereitschaft durch farbige Gesichter verdeutlicht. Durch ein Anklicken der Namen über den Gesichtssymbolen erhält man genauere Informationen über die entsprechende Person.

Im unteren Bereich befindet sich schließlich ein Textfeld, in dem beliebige Informationen eingeblendet werden. Der Hauptzweck dieser Komponente besteht in der Mitteilung von nach der Filterung der DDE Filteragenten beim Benutzeragenten eintreffenden DDE-Aktionen.

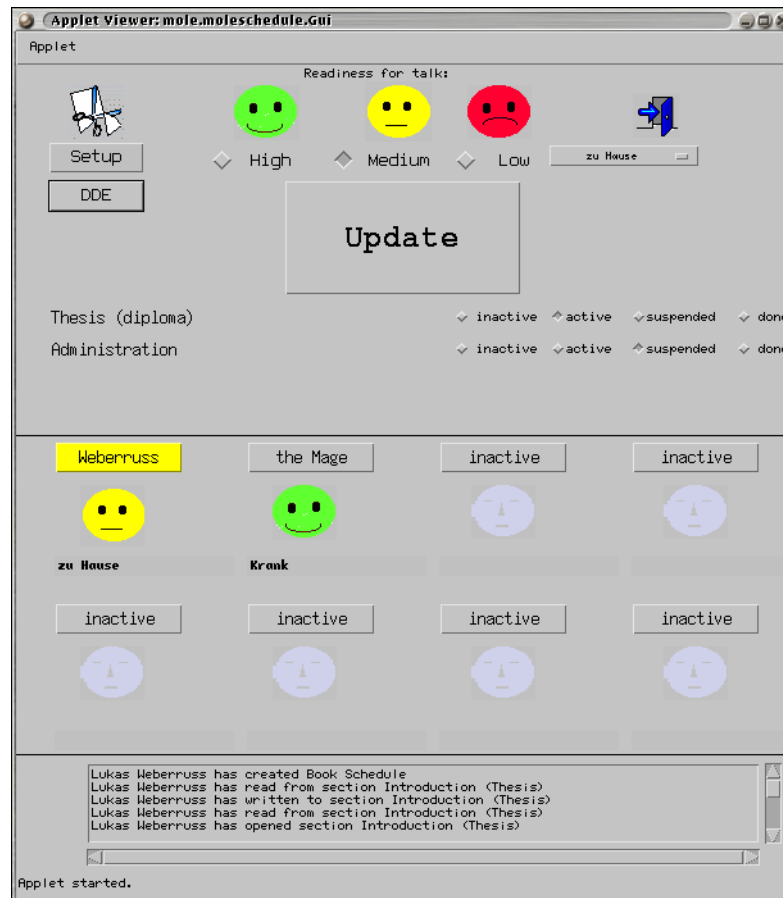


ABBILDUNG 20. Hauptfenster von MoleOffice

Im Setup Fenster kann man alle persönlichen Einstellungen vornehmen. Diese können durch Betätigung eines Knopfes dauerhaft beim persönlichen Benutzeragent gespeichert werden. Neu ist vor allem die Möglichkeit, sein Passwort zu verändern (Knopf links unten).

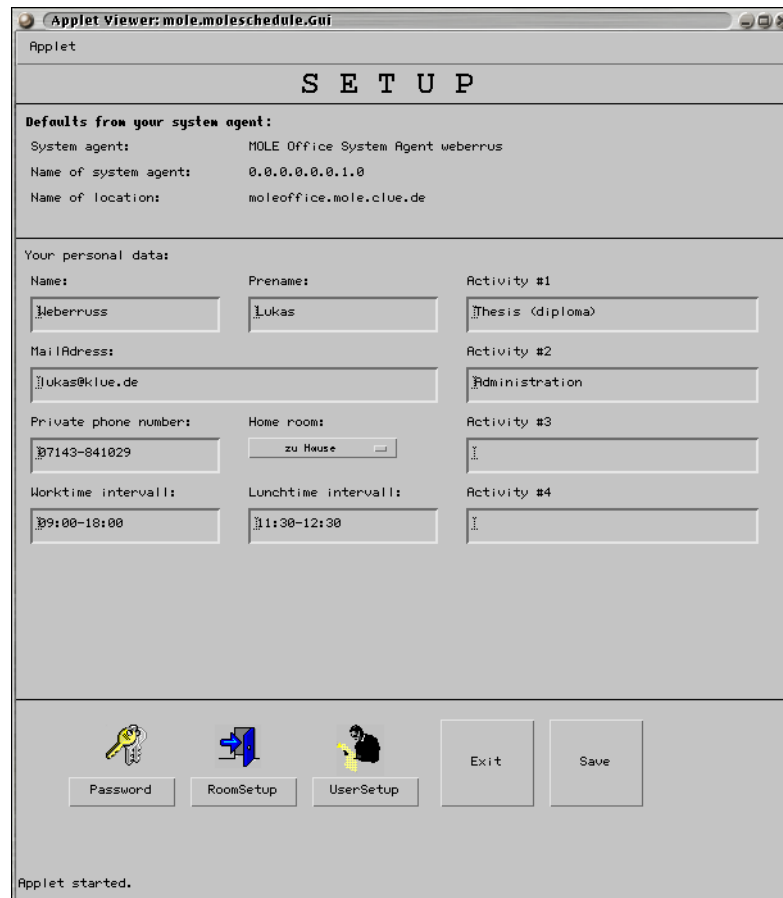


ABBILDUNG 21. Setup-Fenster von MoleOffice

Wie schon erwähnt läuft der komplette MoleOffice Klient als Applet innerhalb der virtuellen Laufzeitumgebung des Appletviewers, so daß bei jedem Benutzer eine solche vorhanden sein und gestartet werden muß.

3.12.3 Das DDE Informations Fenster

Durch die Aufteilung der für die Nutzung eines Verteilten Dokumentensystems notwendigen Information wurde eine relativ übersichtliche, einfach zu bedienende grafische Benutzungsoberfläche entworfen. Dabei werden durch einfache Mausklicks sämtliche Informationen über die Kapitel angezeigt.

Das Fenster teilt sich in drei Hauptbereiche:

- Rechts oben kann man durch Knöpfe in das Konfigurationsfenster von MoleDDE wechseln, den eigentlichen Editor starten, ein manuelles Update auslösen oder das Fenster temporär schließen.
- Links unten befindet sich eine Liste aller existierenden Kapitel mit der Angabe des entsprechenden Buches. Durch Anklicken eines Kapitels in dieser Liste werden sämtliche Informationen über dieses Kapitel angezeigt.
- Diese Informationen werden rechts unten in drei weiteren Bereichen dargestellt. Ein Bereich umfaßt dabei eine Liste der Personen, die das Kapitel gerade geöffnet haben. Ein weiterer Bereich zeigt den Besitzer des Kapitels. Schließlich ist eine Liste vorhanden, die alle Personen aufzählt, welche Modifikationen am Kapitel vorgenommen haben.

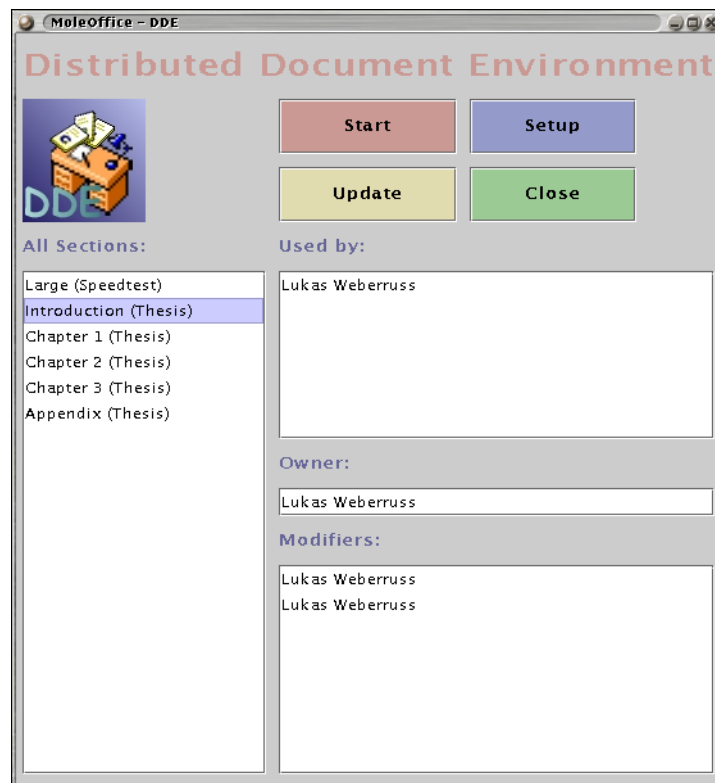


ABBILDUNG 22. MoleDDE Informationsfenster

Im Gegensatz zum Nachrichtbereich des Hauptfensters von MoleOffice, bei dem Ereignisse, also DDE Aktionen, angezeigt werden, sieht man hier eine statische Übersicht des aktuellen Zustands. Arbeitet man selbst gerade nicht an einem Dokument innerhalb von Mole-DDE, kann man dieses Fenster geschlossen halten und wird durch eine entsprechende Einstellung der Filter trotzdem über wichtige Ereignisse informiert.

Je nach Einstellung der eigenen Eingangs-Filter beziehungsweise der fremden Ausgangs-Filter kommt es vor, daß gewisse Informationen erst nach einem manuellen Update dargestellt werden.

3.12.4 Das DDE Setup Fenster

Durch das Betätigen des Setup Knopfes des Informationsfensters öffnet sich ein Fenster zur Konfiguration aller DDE-spezifischen Parameter. Hier können der gewünschte Editor ausgesucht, die notwendigen Einstellungen für den gewählten Editor vorgenommen und im unteren Bereich die Regeln für die Eingangs- und Ausgangsfilter eingestellt werden.

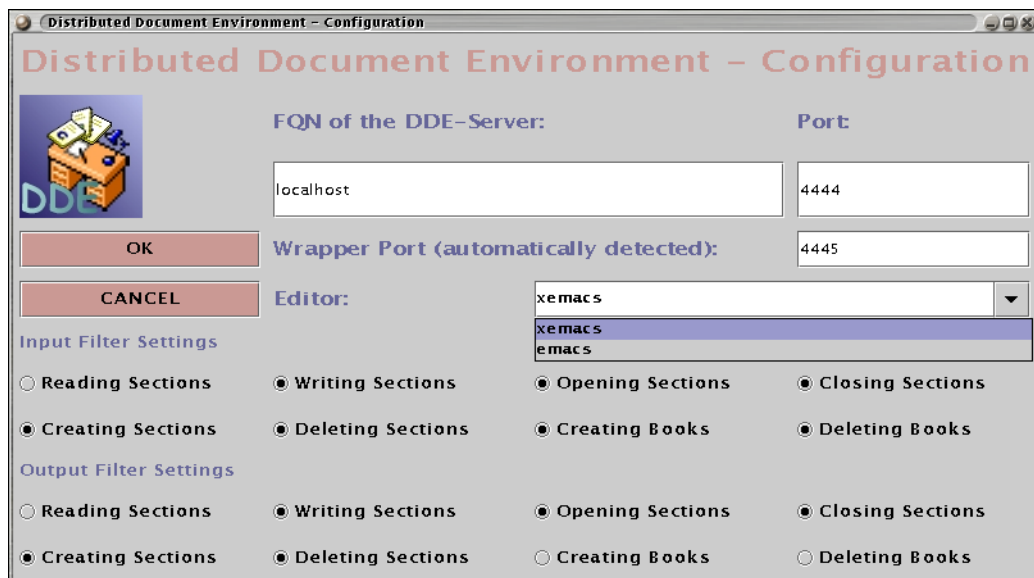


ABBILDUNG 23. MoleDDE Setup Fenster

3.12.5 Der Startbildschirm

Je nach Ausstattung des für den Klienten verwendeten Rechners und der Qualität des Netzwerks kann der Start von MoleOffice relativ lange dauern. Bisher wurden auf der Konsole zum einen ein Countdown von zehn bis null heruntergezählt und zum anderen Debugging-Informationen angezeigt.

Im Falle eines erfolgreichen Starts von MoleOffice sind die Debugging-Informationen jedoch relativ unwichtig und zudem nicht sichtbar, falls MoleOffice nicht von einer Konsole aus gestartet wurde.

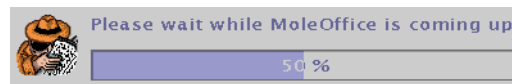


ABBILDUNG 24. MoleDDE Start Fenster

Deshalb werden die Startmeldungen nun in einer Datei mitprotokolliert und ein kleines Fenster, wie in Abbildung 24 dargestellt, zeigt den Verlauf des Startvorgangs an.

3.12.6 Das Loginfenster

Bevor ein Benutzer mit MoleOffice arbeiten kann, muß er sich nun über ein Passwort gegenüber dem MoleOffice Server authentifizieren. Bei der ersten Anmeldung ist dieses allerdings noch nicht gesetzt.

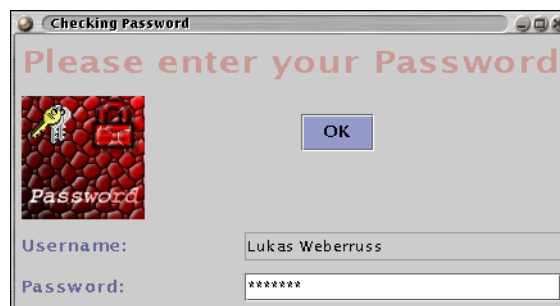


ABBILDUNG 25. MoleDDE Loginfenster

Der Benutzername ist schon im Voraus durch die Konfiguration der HTML-Datei festgelegt. Das Feld mit dem Benutzernamen dient lediglich der eigenen Kontrolle und Information.

3.12.7 Die Passwortverwaltung

Ausgehend vom Konfigurationsdialog von MoleOffice kann auch das Passwort verändert werden. Auch hier dient der Benutzername nur der eigenen Kontrolle.



ABBILDUNG 26. MoleDDE Passwort Konfigurationsfenster

Ein Administrator kann an dieser Stelle aber auch einen fremden Benutzernamen eintragen und dadurch beispielsweise dessen Zugang freischalten.



Eine relativ einfach gehaltene grafische Oberfläche erlaubt eine komfortable Benutzung von Mole-DDE

KAPITEL 4

IMPLEMENTIERUNG

Auch bei der Implementierung des Entwurfs wurden einige zusätzliche Entscheidungen getroffen. Diese sollen in diesem Kapitel vorgestellt und anhand von Beispielen erläutert werden. Zudem sollen die zur Entwicklung des Systems eingesetzten Werkzeuge vorgestellt werden.

4.1 Entwicklungswerkzeuge

Gegenwärtig werden größere Projekte nur noch sehr selten ohne entsprechende mächtige Entwicklungswerkzeuge durchgeführt. Oft helfen grafische Editoren bei der Gestaltung der Benutzungsoberflächen oder es werden Programme zur Versionskontrolle eingesetzt, um den Überblick parallel laufender Arbeiten zu kontrollieren.

4.1.1 Compiler

Mole selbst ist komplett in Java implementiert worden, wodurch eine umfangreiche Klassenbibliothek zur Verwendung der eigenen Agenten vorhanden ist. Ursprünglich entworfen wurde Mole unter Verwendung der Version 1.0x des “*Java Development Kids*¹” von Sun MicrosystemsTM; es wurde aber später auf die nicht ganz kompatible Version 1.1x des *JDK* portiert. Auch die aktuellere Version 1.2.x ist in vielen Punkten nicht mehr kompatibel zur Vorgängerversion. Vor allem die Thread Umgebung wurde stark verändert. Threads sollen nicht mehr durch die schon vorhandenen “*stop*” und “*pause*” Aufrufe unterbrochen werden, sondern es soll durch eigene Konzepte deren Semantik implementiert werden. In Mole ist ein mächtiger Thread Mechanismus mit eigenem Scheduler schon integriert. Aufgrund dieser Inkompatibilitäten war es nicht möglich, das aktuelle *JDK* in der Version 1.2.2 zu verwenden.

Bei der Implementierung der grafischen Benutzungsoberfläche war von großer Bedeutung, daß auch das *AWT* des *JDK* stark überarbeitet und *SWING* fest in den Compiler integriert wurde.

1. im weiteren nur noch *JDK* genannt.

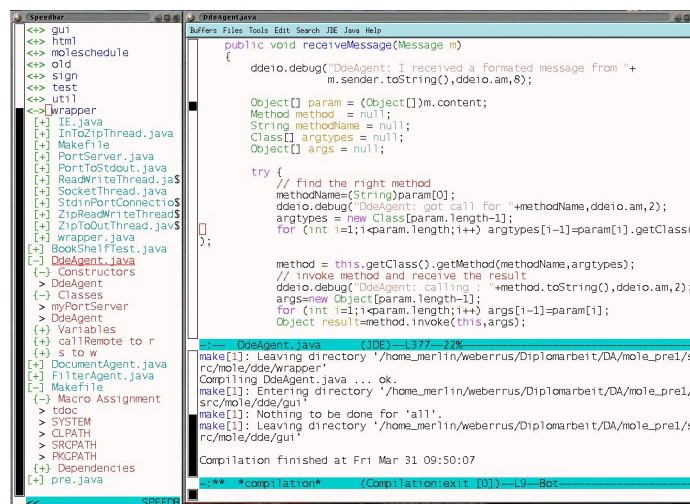
Die von mir neu entworfenen Klassen wurden so gestaltet, daß sie sowohl mit dem *JDK 1.1.x* als auch mit dem *JDK 1.2.x* zusammenarbeiten. Zur Gestaltung der grafischen Oberfläche wurde auf die *SWING* Klassenbibliothek zurückgegriffen, da diese wesentlich mehr Möglichkeiten zur Gestaltung bieten als die des AWT. Die komplette Klientenumgebung ist daher auch unter *JDK 1.2.2* lauffähig.

4.1.2 Das Java Development Environment

Da die unter Unix frei zugängliche Software im Bereich der Entwicklung von auf Java basierenden Projekten sich noch im Anfangsstadium ihrer Entwicklung befindet, wurde auf den Großteil dieser Werkzeuge verzichtet.

Eingesetzt wurde allerdings ein spezielles, unter der GPL vertriebenes Modul für den Editor Emacs, der neben der Verwendung im Projekt selbst auch als Entwicklungswerkzeug eingesetzt wurde. Das Java Development Environment in der Version 2.1.5 bietet dem Programmierer die folgenden Hilfsmittel an:

- Farbliche Hervorhebungen im Quelltext zur besseren Übersicht.
- Direktes Compilieren innerhalb des Emacs zum zügigeren Erkennen und Beheben von Fehlern.
- Eine “*Speedbar*”, um auf die vorhandenen Klassen zuzugreifen und so die entsprechenden Methoden oder globalen Variablen im Quelltext einfacher zu finden.



Die Umgebung bietet darüber hinaus noch einige weitere von mir nicht genutzte Funktionen.

4.2 Implementierung

In diesem Kapitel soll gezeigt werden, welche Konzepte bei der Programmierung des Systems eingesetzt werden und welche Vor- beziehungsweise Nachteile sie haben. Über dieses Kapitel hinausgehende Informationen findet man in der von Java mittels “*JAVADOC*” generierten HTML-Struktur.

4.2.1 Das Bücherregal

Das gesamte Bücherregal wird so entworfen, daß es eins zu eins auf Klassen abgebildet werden kann. Desweiteren kann es durch seinen modularen Aufbau auch jederzeit in beliebigen anderen Projekten weiterverwendet werden. Die unter Corba-DDE vorhandenen Klassen fanden durch ihre Corba-Orientierung in diesem Projekt keinen Einsatz.

Zentrales Objekt innerhalb des Bücherregals ist das Kapitel. Es besitzt die folgenden *wichtigen* Variablen:

- Die Identifikationsnummer (*id*)
- Einen beliebigen String als Namen (*name*)
- Der Name des übergeordneten Buches (*bookname*)
- Einen Verweis auf den Besitzer des Kapitels (*owner*)
- Einen Vektor auf alle aktuellen Benutzer des Kapitels (*users*)
- Eine Liste aller Benutzer, die dieses Kapitel verändert haben (*modifiers*)
- Den aktuellen Inhalt des Kapitels (*content*)

Da die Klasse serialisierbar ist, werden alle Parameter automatisch mit gespeichert. Der Inhalt des Kapitels wird allerdings von dieser Komponente selbst verwaltet und nur bei Bedarf geladen oder gespeichert.

Zur externen Datenhaltung des Kapitelinhalts können zwei verschiedene Alternativen herangezogen werden:

1. Speichern und Laden des Inhalts geschieht über die Serialisierung in eine Datei.
2. Der Inhalt des Kapitels wird eins zu eins mit den üblichen Mitteln in eine Datei geschrieben.

Punkt 1 hat den Vorteil, daß ein einziger Befehl den Inhalt laden bzw. speichern kann zudem ist stets überprüfbar, ob der Inhalt wirklich erfolgreich geladen werden kann oder ob es Modifikationen gab. Gerade aber das letzte Argument erlaubt es nicht, auch

außerhalb von Mole-DDE Modifikationen am Inhalt vorzunehmen, weshalb die Entscheidung zugunsten der zweiten Variante ausfiel. Durch diese Entscheidung ist es beispielsweise möglich, relativ einfach schon vorhandene Dokumente nach Mole-DDE zu portieren.

Die Buchklasse ähnelt der Kapitelklasse mit dem Unterschied daß die Semantik der verschiedenen Felder eine andere ist. Inhalt eines Buches sind die einzelnen Kapitel, übergeordnetes Objekt ist das Bücherregal. Es gibt ebenfalls Listen, die die Nutzung des Buches beschreiben. Auch hier wird der Inhalt erst dann dynamisch geladen, wenn darauf zugegriffen wird. Bei der Abfrage des Inhaltsverzeichnisses eines Bücherregals ist es nicht von Bedeutung, welche Kapitel die Bücher haben beziehungsweise welchen Inhalt das einzelne Kapitel hat. Zusätzlich zur Funktionalität der Kapitel kommt in der Buchklasse noch hinzu, daß man Kapitel anlegen, verschieben oder löschen kann.

Auch das Bücherregal ist im Prinzip ähnlich aufgebaut. Da aber hier Modifikationen seltener sind als in den darunterliegenden Klassen, wird es beim Initialisieren schon vollständig hergestellt und muß nicht erst beim ersten Zugriff geladen werden.

4.2.2 Der Taktgeber der Agenten

Da die Funktionalität der in Mole integrierten Herzschlagfunktion nicht ausreichend ist, wird eine eigene, variablere Implementierung herangezogen. Sie unterstützt zusätzlich folgende Eigenschaften

- Eine Einstellung der minimalen Wiederholungsrate eines Herzschlages¹.
- Informationen über die Anzahl bisher gemachter Herzschläge.
- Läuft auch während der “*stop*”-Methode eines Agenten.
- Kann (und muß) vom Agenten selbst beendet werden.

Damit ein Agent den neuen Herzschlag verwenden kann, muß er die Schnittstelle “*Pumper*” implementieren, welche analog zur Schnittstelle “*Runnable*” lediglich die Implementierung einer bestimmten Methode erzwingt. Jeder Agent, der die neue Funktionalität benutzen will, braucht eine eigene Instanz der Klasse “*HeartBeat*”.

1. Die maximale Wiederholungsrate des Herzschlags kann aufgrund der nicht vorhandenen Echtzeit Umgebung nicht sicher garantiert werden.

```
public class DdeAgent
extends SystemAgent
implements Pumper
{
    [...]
    private HeartBeat heartbeat;
    [...]
    public void start()
    {
        [...]
        heartbeat=new HeartBeat(this,speed);
        heartbeat.start();
        [...]
    }
    [...]
    public void pump(long pumps)
    {
        [...]
    }
    [...]
}
```

ABBILDUNG 27. Beispiel zum Einsatz des HeartBeat-Mechanismus

Dieser bietet dann einen separaten Thread, der regelmäßig die “*pump*”-Methode des Agenten aufruft.

Der neue Herzschlagmechanismus wird in den beiden Agentenklassen DDE Agent und Dokumenten Agent eingesetzt. Beim DDE Agenten erfolgt die Kommunikation mit dem Wrapper Server über diesen Mechanismus. Es wird periodisch geprüft, ob neue Informationen vorliegen. Ist dies der Fall, wird ein neuer Thread gestartet, der die Informationen abarbeitet. Da unter Umständen relativ oft Informationen zur Verarbeitung anstehen, reichte hier der von Mole unterstützte Rhythmus von einem Aufruf pro Sekunde nicht aus. Zudem können auch während des Herunterfahrens des DDE-Servers noch Informationen, wie beispielsweise eine Abmeldung des Wrappers entgegenommen werden.

Beim Dokumentenagenten regt der regelmäßige Herzschlag eine Funktion zum automatischen Speichern an. Auch hier ist ein Rhythmus von einem Aufruf pro Sekunde nicht geeignet.

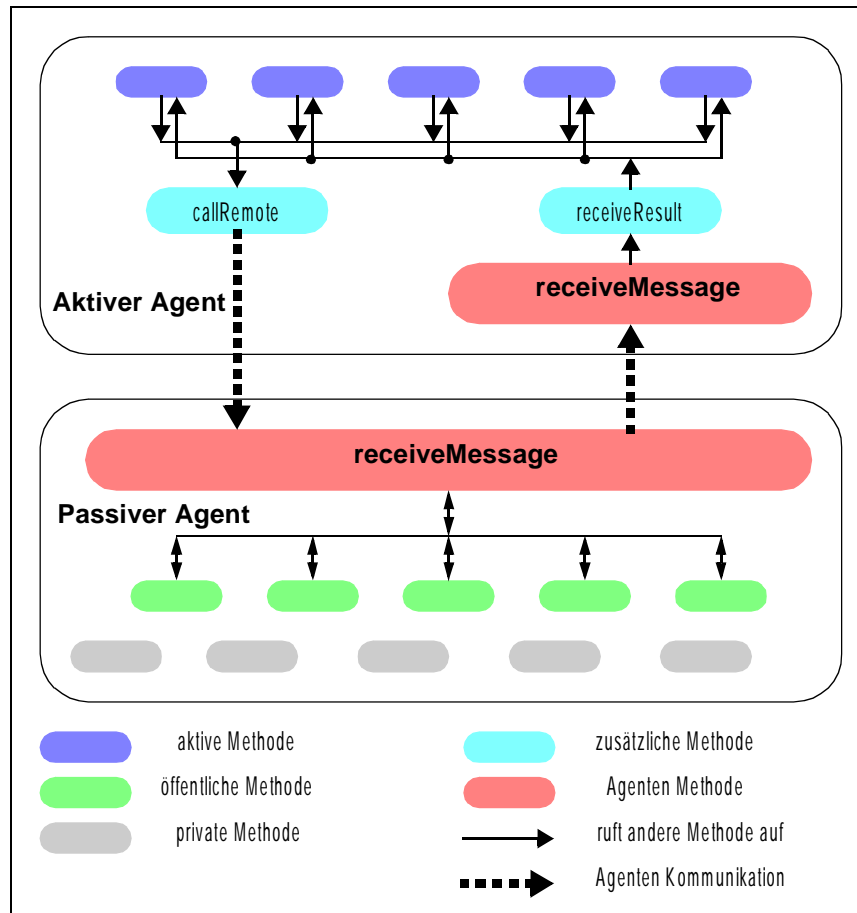
4.2.3 Agentenkommunikation

Wie in Abschnitt 2.3 beschrieben, enthält Mole schon einen geeigneten Mechanismus zur Kommunikation der Agenten. Dieser beruht darauf, daß in der Methode “receiveMessage” implementiert wird, was ein Agent beim Erhalt einer Nachricht durchführen soll. Bei den bisher in MoleOffice vorhandenen Agenten stellen die Implementierungen dieser Methode den Großteil der Klasse dar. Sie enthalten unzählige *IF-THEN-ELSE* Anweisungen, die den Inhalt der Nachrichten überprüfen und daraufhin eine geeignete eigene Methode aufrufen.

Um die Kommunikation der neu hinzugekommenen Agenten zu erleichtern und übersichtlicher zu gestalten, verwenden alle Agenten von Mole-DDE¹ eine ganz spezielle Implementierung der Methoden zum Empfangen und Versenden von Nachrichten. Durch den Einsatz geeigneter Nachrichten kann ein Agent jede als “*public*” deklarierte Methode eines anderen Agenten von Mole-DDE direkt adressieren, unabhängig davon, an welcher Lokation sich die Kommunikationspartner befinden.

Falls die aufgerufene Methode ein Ergebnis zurückliefert, wird dieses selbst in eine Nachricht verpackt und an den aufgerufenen Agenten zurückgeschickt. Hierdurch entfallen die in der Methode “receiveMessage” gebündelten *IF-THEN-ELSE* Anweisungen. Die Überprüfung der Nachrichten geschieht in der entsprechenden, indirekt aufgerufenen Methode selbst.

1. DDE Agent, Dokumenten Agent und Filter Agent


ABBILDUNG 28. Erweiterte Agentenkommunikation

Wie die Abbildung zeigt, dient die Methode “receiveMessage” dabei als Schnittstelle, die wie ein Multiplexer den Aufruf an das eigentliche Ziel weiterreicht. Zum Empfang von Ergebnissen wurde die öffentliche Methode “receiveResult” eingeführt. Sie entscheidet anhand eines von der aufrufenden Methode gesetzten Schalters, wie mit dem Ergebnis weiter zu verfahren ist.

4.2.4 Wrapper Kommunikation

Für die Kommunikation zwischen Wrapper und DDE Agenten wurden zwei verschiedene Mechanismen in Betracht gezogen:

1. Kommunikation über “Entfernte Methoden Aufrufe” (RMI)
2. Kommunikation über Datenströme auf Unix Ports

Variante 1 verspricht eine relativ einfache Implementierung, da beide Kommunikationspartner lediglich geeignete Methoden bereitstellen müssen. Um alle weiteren Details zur Datenübertragung kümmert sich der RMI-Mechanismus von Java.

Variante 2 ermöglicht hingegen eine individuelle, problemangepaßte Kommunikation. Sie verlangt allerdings eine eigene Implementierung der Kommunikationsdetails. Da eventuell große Datenmengen¹ zwischen dem DDE Agenten und den Wrappern ausgetauscht werden müssen, wird diese Variante bevorzugt.

Die verwendete Lösung beruht auf einer Klienten/Server Architektur, wobei der DDE Agent einen Server startet, der auf einem freien Unix-Port auf die Anmeldung eines Klienten wartet. Im Fall einer erfolgreichen Anmeldung wird nun ein Kommunikationskanal geöffnet und dessen Zustand periodisch über den Herzschlagmechanismus abgefragt. Da einzelne Nachrichten relativ groß sein können, benötigt man eine geeignete leistungsfähige Struktur an Puffern, die zum einen effizient ist, zum anderen aber nicht das gesamte System während der Bearbeitung blockieren kann.

Über das Konfigurationsprogramm können die meisten dieser Parameter individuell an ein bestimmtes System angepaßt werden.

Um das zu transportierende Datenvolumen zu reduzieren, ist es mit dieser Methode relativ einfach möglich, die Daten in komprimierter Form zu übertragen. Hierzu existieren in Java die Klassen "*java.util.zip.GZIPInputStream*" und "*java.util.zip.GZIPOutputStream*". Sie kümmern sich über die zur Datenkomprimierung nötigen Details. Um die Sicherheit der Kommunikation zu erhöhen, kann nach dem gleichen Prinzip zusätzlich auch ein Verschlüsselungsmechanismus Anwendung finden.

1. Es gibt beispielsweise keine Größenbeschränkung für Dokumente.

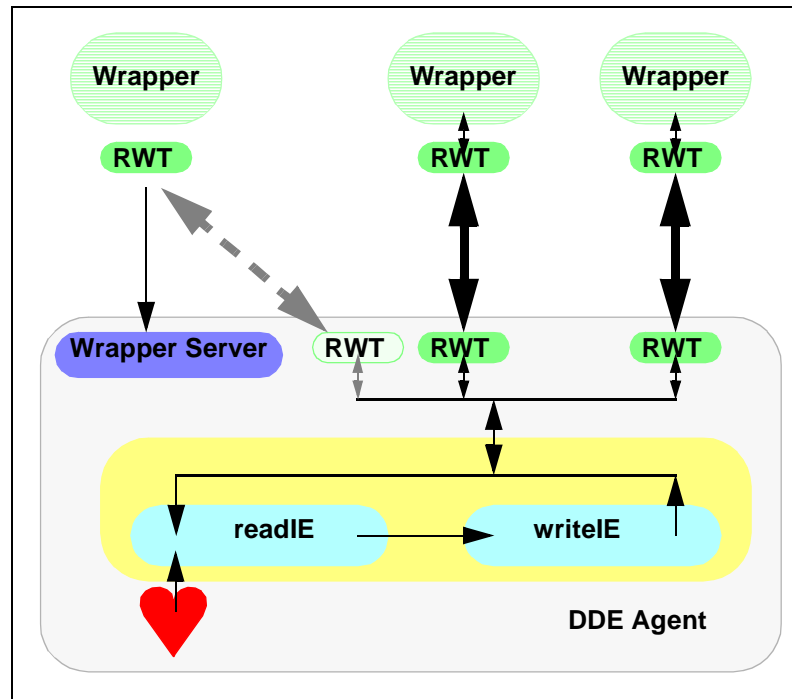


ABBILDUNG 29. Kommunikation zwischen DDE Agenten und Wrapper

Abbildung 29 zeigt eine beispielhafte Situation, in der sich gerade ein Wrapper beim DDE Agenten über dessen Wrapper Server anmeldet. Zwei weitere Wrapper sind schon mit Mole-DDE verbunden. Für die Verarbeitung der Informationen sind im wesentlichen zwei Methoden zuständig. Die *readIE*-Methode liest vom ReadWrite-Thread (RWT) eine Informationseinheit (IE) und reicht sie an die Methode *writeIE* weiter. Hier wird der Inhalt der IE geprüft und schließlich sowohl eine DDE Aktion als auch eine Antwort an den Wrapper generiert und verschickt.

4.2.5 Transport von Informationseinheiten

Der Informationsfluß zwischen DDE Agent und Editor verläuft über mehrere Stufen. Generiert werden die IEs sowohl im DDE Agenten als auch im Editor. Beim Transport vom DDE Agenten aus werden die IEs den entsprechenden RWTs übergeben. Dort werden sie über gepufferte Datenströme schließlich über einen Unix Port zur Gegenstelle übertragen. Die Funktionalität der Datenströme sorgt dafür, daß die zum Teil großen Datenmengen korrekt am anderen Ende wieder von einem RWT gelesen werden können. Der RWT des Wrappers liest die Informationen und übergibt sie dem eigentlichen Wrapper. Dieser baut die Informationen anhand der festen Struktur der IEs wieder zusammen.

Die Rekonstruktion gelingt durch die speziellen Trennzeichen der Informationseinheiten. Wie in Abschnitt 3.11.1 beschrieben, haben die Informationseinheiten folgende Struktur: “<...><...><...>\n”¹. Die einzelnen Nachrichten werden somit durch die Sequenz “>\n” am Ende jeder IE erkannt.

Der Wrapper überprüft nun den Zustand der Informationseinheiten, indem er die ersten beiden Felder überprüft (Identifikation und Kommando²). Das letzte Feld mit dem eigentlichen Inhalt wird nicht überprüft, da hier ein beliebiger Inhalt steht und die darunter liegende Protokollschicht des TCP/IP die Aufgabe übernimmt, die Daten ohne Fehler über das Netzwerk zu übertragen. Vom Wrapper bearbeitet werden die IEs wieder einem RWT übergeben, der die Daten nun seinerseits über einen weiteren Unix Port an den Editor reicht.

In der entgegengesetzten Richtung geschieht prinzipiell das gleiche, nur daß hier am Schluß der DDE Agent die Informationen wieder zusammensetzt.

4.2.6 Verwaltung der Kapitel

Der aktuelle Zustand der Kapitel wird durch verschiedene Komponenten beschrieben. Er läßt sich durch die in Tabelle 4 gezeigten Informationen stets bestimmen.

Attribut	statisch/dynamisch	zuständige Komponente
Name	statisch	ddeSection
Identifikationsnummer	dynamisch	ddeSection/ddeBook
Büchername	statisch	ddeSection/ddeBook
Eigentümer	statisch ^a	ddeSection
Benutzer	dynamisch	User/ddeSection
Modifizierer	dynamisch	ddeSection
Inhalt	dynamisch	ddeSection

Tabelle 4. Zustandsinformation eines Kapitels

a. Der Eigentümer ist bisher nicht veränderbar.

-
1. Das Zeichen für einen Zeilenumbruch wird hier als “\n” dargestellt, da diese Schreibweise in den meisten Programmiersprachen üblich ist (C, C++, Java, Shell, ...).
 2. siehe Abbildung 17: Definition einer IE in BNF

Es existieren sowohl statische, bei der Instantiierung festgelegte, als auch dynamische, zur Laufzeit veränderbare Attribute. Da bei einem Kapitel dessen Position im Buch zur Identifikation herangezogen wird, ist die Identifikationsnummer dynamisch. Der Name eines Kapitels ist bisher nicht direkt veränderbar¹.

Die Verknüpfung der Benutzer zu den Kapiteln geschieht über zwei unterschiedliche Listen. Zum einen “*weiß*” jedes Kapitel, wer es gerade geöffnet hat. Zum anderen besitzt jedes Benutzerobjekt eine Liste der geöffneten Kapitel, so daß zur Überprüfung der geöffneten Kapitel weder die Liste aller Benutzer noch die Liste aller Kapitel durchsucht werden muß. Die Verwaltung dieser Informationen teilen sich der DDE Agent und der Dokumenten Agent, da der Dokumenten Agent zwar Informationen über die Benutzer geöffneter Kapitel durch Überprüfung aller Kapitel bekommen könnte, er selbst aber aus Konsistenzgründen keine Veränderung an den Benutzerinformationen vornehmen darf. Der DDE Agent darf seinerseits keine Veränderungen am Bücherregal vornehmen.

4.2.7 Filterung von Aktionen

Wie schon in Kapitel 3.7 auf Seite 38 erwähnt wurde können die im DDE Agenten generierten Aktionen durch die persönlichen Einstellungen gefiltert werden. Zwei unterschiedliche Filter sind vorhanden: Zum einen existiert der Ausgangsfilter, der bestimmt, welche Informationen an welche anderen Agenten weitergereicht werden, zum anderen bestimmt ein Eingangsfilter, ob Informationen an den persönlichen Benutzeragenten übergeben werden.

1. Innherhalb des DDE-Systems kann diese Funktionalität bei Bedarf leicht ergänzt werden. Um sie nutzen zu können bedarf es allerdings auch einen geeigneten Editor.

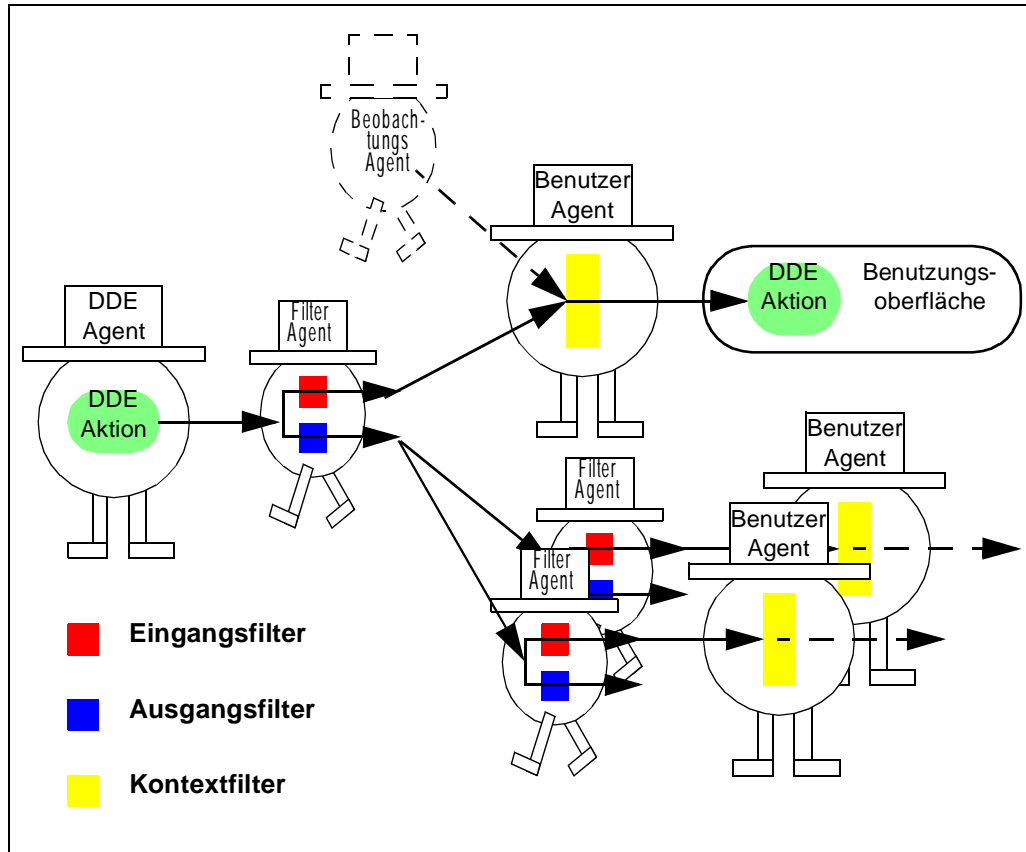


ABBILDUNG 30. Filterung von Aktionen

Durch diese Aufteilung der Filter können für jede Klasse von Aktionen die beiden folgenden Entscheidungen getroffen werden:

- Will ich, daß andere von dieser Aktion unterrichtet werden?
- Habe ich selbst Interesse an dieser Aktion?

Vom DDE Agenten wird in jedem Fall der Filter Agent informiert, dessen zugehöriger Benutzer die Aktion ausgelöst hat. Dieser trifft nun für beide Filter eine Entscheidung. Es ist also auf diese Weise nicht möglich, Informationen nur an bestimmte (autorisierte) Benutzer zu reichen. Dagegen kann beispielsweise eingestellt werden, daß man selbst nicht über das Lesen eines Kapitels informiert werden, anderen diese Information allerdings nicht vorenthalten will.

Generell bekommt man allerdings nur Informationen über Personen im eigenen Kontext, unabhängig davon welche Art von Informationen es sich handelt. Dieser Kontextfilter existierte schon in *MOLEOFFICE*.



Durch Kombination verschiedener Filter wird dem Benutzer die Möglichkeit gegeben, die Verbreitung von durch ihn generierten Informationen zu steuern.

4.2.8 Anbindung der Benutzungsoberfläche

Aus technischer Hinsicht ergeben sich folgende Anforderungen an die Benutzungsoberfläche:

1. Interaktion mit dem Benutzeragenten
2. Visualisierung von Zustandsinformationen
3. Modifikation der Daten
4. Starten, Beenden und Kontrollieren von Programmen
5. Lesen und Schreiben des Dateisystems

Vor allem zur Realisierung der Punkte 1, 4 und 5 bedarf es Zugriffsmöglichkeiten auf die Systemressourcen. Die Programmiersprache Java bietet hier mittlerweile für alle Punkte geeignete Konzepte an.

Allerdings wurde die bisherige grafische Benutzungsoberfläche als Applet und nicht als Applikation realisiert. Applets laufen im Prinzip in einer Art Sandkasten, was bedeutet, daß sie viele Dinge aus sicherheitstechnischen Aspekten nicht tun können, ganz im Gegensatz zu normalen Applikationen. Für das bisherige MoleOffice war lediglich ein Zugriff auf das Netzwerk, nicht aber auf externe Programme oder Daten nötig, weshalb die Funktionalität eines Applets dort noch völlig ausreichte.

Unter die Zugriffseinschränkungen eines Applets fallen vor allem die folgenden Punkte:

- 1. Zugriff auf das Netzwerk**
- 2. Zugriff auf das Dateisystem**
- 3. Zugriff auf die Prozeßumgebung anderer Programme**
- 4. Uneingeschränktes Erzeugen von Fenstern**

Wie man sieht, widersprechen diese Punkte teilweise den oben erwähnten Anforderungen. Da diese Einschränkungen oft zu restriktiv sind, gibt es Mechanismen, um den Sandkasten zu verlassen. Da dies nicht unkontrolliert und vor allem unbemerkt geschehen darf, muß es eine Kontrollinstanz geben, die der Programmierer eines Applets nicht alleine umgehen kann. Weil für die Sicherheit die Prozeßumgebung, in der das Applet läuft, zuständig ist, werden die verschiedenen Umgebungen auf die Möglichkeit ihrer Einsatztauglichkeit für dieses Projekt hin untersucht. Aus der Vielzahl der Java Prozeß Umgebungen wurden die folgenden ausgewählt:

- Netscape Communicator Version 4.x
- MS Internet Explorer Version 4.x
- Suns Appletviewer Version 1.1.x
- Suns Appletviewer Version 1.2.x

Bei der Untersuchung wurde festgestellt, daß sämtliche hier aufgeführten Umgebungen einen anderen Mechanismus bieten, die Zugriffsmöglichkeiten einzustellen. Im Internet können Informationen abgerufen werden, die bis aufs genaueste den jeweils relativ komplizierten Vorgang beschreiben, seine Applets unter den entsprechenden Umgebungen lauffähig zu machen.

Der Netscape Communicator verlangt eine öffentliche, kostenpflichtige Registrierung der eigenen Klassen, wofür man einen Schlüssel zum Signieren der Klassen bekommt. Zuvor muß man allerdings ein JAR-Archiv erstellen, denn nur diese lassen sich digital signieren. Mit der Klassenbibliothek "*netscape.security*" ist es schließlich möglich, den Anwender über sicherheitskritische Vorgänge zu informieren.

Eine ähnliche Situation besteht bei Microsofts Internet Explorer. Er benutzt jedoch komplett andere (Windows-)Programme und benötigt zusätzlich den Einsatz der Klassenbibliothek "*com.ms.security*" in den Methoden "*start*", "*stop*", "*init*" und "*destroy*".

Das Archivformat von Microsoft ist ebenfalls anders aufgebaut¹ und zum JAR-Archiv Format inkompatibel.

Beim Appletviewer in der Version 1.1.x kann der Benutzer selbst mit den im JDK enthaltenen Programmen einen Schlüssel generieren und seine Klassen damit signieren. Dazu muß ebenfalls zuerst ein JAR-Archiv erstellt werden. Der generierte öffentliche Schlüssel muß dann jedem Anwender vorliegen.

Beim Appletviewer in der Version 1.2.x sind zwei Alternativen möglich. Zum einen kann man ähnlich der Version 1.1.x ein Archiv erstellen und dieses mit dem selbst generierten Schlüssel signieren. Als zweite Möglichkeit kann jeder Anwender in einer Datei selbst festlegen, welche Zugriffe von welcher Klasse und welchem Entwickler erlaubt sind.

Alle hier genannten Varianten sind weitgehend inkompatibel zueinander. Daß der Weg von Netscape ein anderer ist als der des Internet Explorers hat vermutlich firmenpolitische Gründe. Daß aber sogar die beiden JDK Versionen zueinander inkompatible signierte Archive verwenden, erschwert die Arbeit des Programmierer unnötig.

Aufgrund dieser Problematik gibt es für die vorliegende Arbeit also folgende Alternativen:

- 1.Portierung des GUI Applets in eine Applikation.**
- 2.Durchführung der Signierungen und Codeanpassungen.**
- 3.Verzicht auf nicht erlaubte Zugriffe.**

Da die grafische Oberfläche von MoleOffice in der mir zugrundeliegenden Version schon relativ groß ist und vor allem gut funktioniert, ist es nicht ratsam, den kompletten Code umzuschreiben. Allerdings ist es für ein prototypisches System auch nicht ökonomisch, mehrere öffentliche und kostenpflichtige Schlüssel zu erwerben und eine für alle Umgebungen kompatible Lösung anzustreben.

Durch eine genaue Überprüfung der Zugriffe können manche eventuell durch Alternativen ersetzt werden. Durch den Einsatz eines zusätzlichen Programms ist der direkte Zugriff auf das Dateisystem über java-eigene Klassen nicht notwendig (nur beim

1. Die Archive haben die Endung .cab

appletviewer). Denkbar wäre auch, das Starten und Beenden von Programmen teilweise vom MoleOffice Server durchführen zu lassen. Um aber entfernte Programme zu starten oder auf einen entfernten Bildschirm zugreifen zu können, bedarf es weiterer sicherheitskritischer Mechanismen, die zum einen teilweise nicht unter allen Betriebssystemen möglich sind. Zum anderen müssen auf Administrator Ebene im System zusätzliche Dienste freigeschaltet werden. Im Unix Umfeld kann man beispielsweise über den Internetdämon (*inetd*), entfernte Programme starten. Auch X11 bietet prinzipiell die Möglichkeit, durch ausdrückliche Genehmigung eines Benutzers ein Fenster zu schicken. Diese Variante belastet allerdings das Netzwerk übermäßig stark.

Nach Abschluß aller Untersuchungen wurde die grafische Oberfläche von MoleOffice unter den folgenden Umgebungen erfolgreich eingesetzt:

Umgebung	Umgehung des Zugriffsschutzes	Bemerkung
Appletviewer V 1.1.x (Archiv)	<ul style="list-style-type: none"> • Generierung eines Schlüssel-paares • Generierung eines JAR Archives • Signierung des Archives • Einsatz eines zusätzlichen Programms zum Schreiben auf das Dateisystem 	Keine Möglichkeit zur Signierung eines für den Appletviewer V 1.2.x signiertes Archiv
Appletviewer V 1.2.x (Archiv)	<ul style="list-style-type: none"> • Generierung eines Schlüssel-paares • Generierung eines JAR Archives • Signierung des Archives • Einsatz eines zusätzlichen Programms zum Schreiben auf das Dateisystem 	Keine Möglichkeit zur Signierung eines für den Appletviewer V 1.1.x signiertes Archiv
Appletviewer V1.2.x	<ul style="list-style-type: none"> • Anpassung der Datei zur Zugriffskontrolle • Einsatz eines zusätzlichen Programms zum Schreiben auf das Dateisystem 	Durch fehlendes Archiv, extrem lange Ladezeit
Netscape Communicator mit jdk1.2.2 Plugin (Archiv)	<ul style="list-style-type: none"> • Installation eines etwa 30MB großen Plugins für alle Anwender • Anpassung der Datei zur Zugriffskontrolle 	Gelegentliche Abstürze da Betaversion

Tabelle 5. Laufzeitumgebungen für die grafische Benutzungsoberfläche

Als optimale Variante stellte sich der Einsatz eines signierten Archives unter dem JDK 1.1.x heraus. Auch der Einsatz des Appletviewers in der aktuelleren Version des JDK 1.2.x bietet eine stabile Arbeitsumgebung, nur daß hier die lange Ladezeit der einzelnen Klassen beim Start des Systems negativ auffällt.



Die Nutzung signierter Applets ermöglicht dem Mole-DDE Klienten, auf alle wichtigen Systemressourcen zuzugreifen.

KAPITEL 5

TESTSZENARIEN

Um zum einen die Korrektheit und die Leistungsfähigkeit von Mole-DDE zu überprüfen und zum anderen über einen Vergleich mit Corba-DDE zu verfügen, wurden mehrere Testszzenarien erstellt. Da sich Mole-DDE in verschiedene Komponenten zerlegen läßt, wurden diese einzeln getestet und bewertet.

5.1 Testumgebung

Die Messungen von Mole-DDE mitsamt aller Komponenten wurden auf einem SMP¹ Rechner mit zwei Intel Pentium II 350 bei 128 MB RAM durchgeführt. Als Betriebssystem diente Redhat Linux Version 6.1 mit *GLIBC 2.1* und *JDK 1.2.2*², bzw. *JDK 1.1.8*. Da das *JDK* den Betrieb mehrerer Prozessoren unterstützt und auch bei der Programmierung von Mole-DDE darauf geachtet wurde, mehrere Threads zu verwenden beziehungsweise Klient und Server auf dem gleichen Rechner liefen erschien es in diesem Fall durchaus als sinnvoll, die Tests auf einem Multiprozessor System durchzuführen.

Für die Vergleichsmessungen mit Corba-DDE stand eine Sparc Sun Ultra 80 Workstation zur Verfügung. Sie war mit 4 Prozessoren a 450 MHz und 1 GB RAM bestückt. Vor allem der extrem großzügig ausgelegte Speicher machte sich hier im Vergleich mit dem PC bemerkbar. Um nicht von schwankenden Netzwerkbelastungen betroffen zu sein, mußte beim Test des gesamten Systems ein XEmacs, der MoleOffice Server und der MoleOffice Klient gleichzeitig auf einem Rechner laufen. Der Server selbst benötigt zwar nur relativ wenig Speicher, aber sowohl Emacs als auch der Appletviewer mit dem Klienten benötigen viel Speicherplatz.

1. Symmetric Multi Processing (Intel Spezifikation zum Betrieb mehrerer Prozessoren in einem PC)
2. Da das *JDK* von SUN keine Unterstützung für "native Threads" bietet wurde das Blackdawn *JDK* eingesetzt.

5.2 Test des Dokumentensystems

Ursprünglich nur zur Überprüfung der Funktionsweise des Bücherregals geplant, entstand ein Programm, welches über eine eigene Kommandozeile steuerbar ist. Die Testkomponente ersetzt also quasi komplett das Agentensystem. Man vergleiche dazu die Abbildung 9 aus Kapitel 3.3 auf Seite 27 mit der Abbildung 31 .

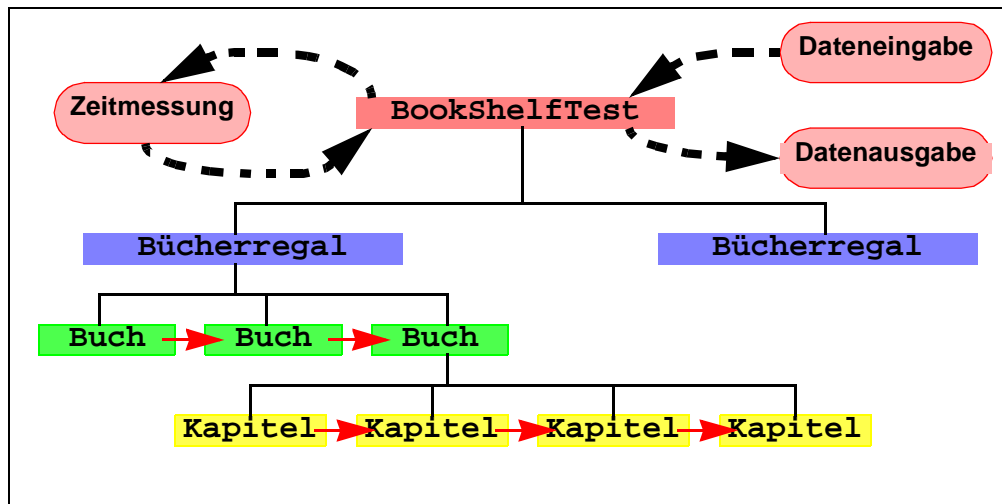


ABBILDUNG 31. Analyse des Bücherregals

Es lassen sich in der Testumgebung entweder von Hand oder per Datei beliebige sequentielle Beispielszenarien erstellen und ausführen. Durch die Möglichkeit der Verwendung einer Datei lassen sich auch repräsentative Zeitmessungen durchführen.

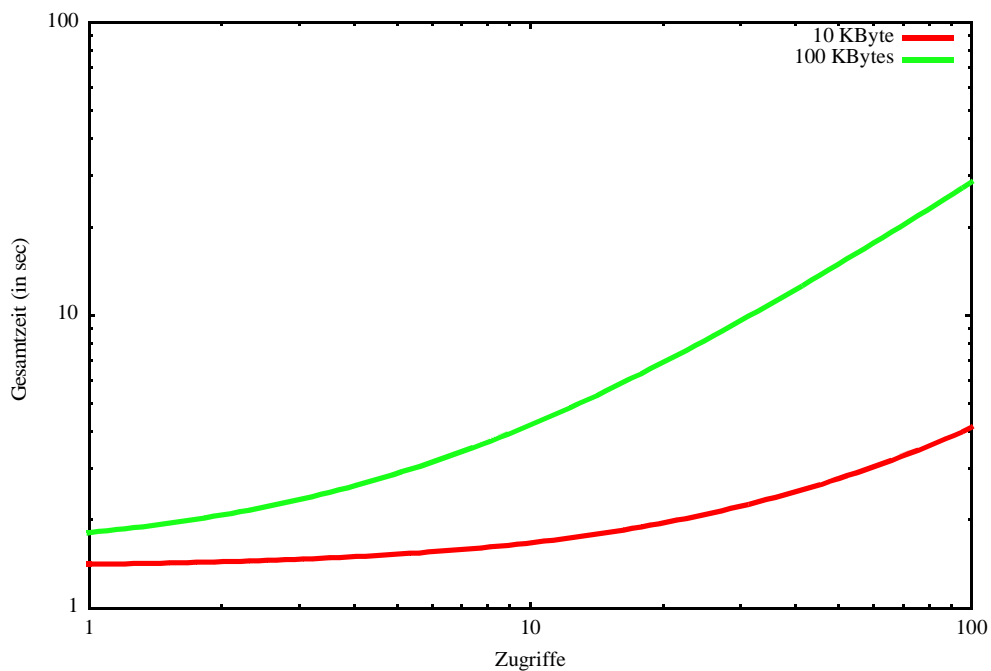


ABBILDUNG 32. Zusammenhang von Zugriffen und Dateigröße bei Bücherregalzugriffen

Die Abbildung zeigt in einem logarithmischen Maßstab, daß die Zugriffszeit nicht von Anfang an linear ansteigt. Es existiert eine sogenannte Mindestzeit, egal welche Dateigröße vorliegt. Diese ergibt sich vor allem aus der Aufrufzeit des JDK. Anmeldung und Initialisieren des Bücherregals benötigen zudem eine gewisse Zeit. Bei einer Dateigröße von 100 KByte ergibt sich so ab etwa 10 Zugriffen ein linearer Verlauf, bei einer Dateigröße von nur 10 KByte erst ab etwa 50 Zugriffen.

Als Ergebnis ist ein maximaler Datendurchsatz von etwa 370 KB pro Sekunde festzuhalten. Die für die virtuelle Laufzeitumgebung relativ positiven Ergebnisse dieser Messungen lassen nur eingeschränkt Rückschlüsse auf das Verhalten von Mole-DDE zu, da sie sich nur auf das Bücherregal, dessen Bücher und Kapitel beziehen. Weiterhin werden keinerlei Daten über ein Netzwerk transportiert oder gar von Agenten verarbeitet.

5.3 DDE System

Als weitaus komplexere Aufgabe als die Messungen des Bücherregals stellte sich eine Einschätzung des kompletten Agentensystem heraus. Um eine relativ realistische Umgebung zu schaffen, müßte ein ziemlich großes Szenario mit vielen Netzwerkknoten arrangiert werden. Da aber hier der Einfluß des Netzwerks selbst ziemlich sicher

eine Verfälschung der Ergebnisse zur Folge hätte, wurde auch hier ein eher einfaches Szenarium aufgebaut.

Die Meßumgebung besteht aus einem Shellprogramm, welches beliebig viele Kommunikationspartner parallel starten und durch Analyse der Datenausgabe und Laufzeit Aussagen über die Korrektheit der Daten und Leistung machen kann.

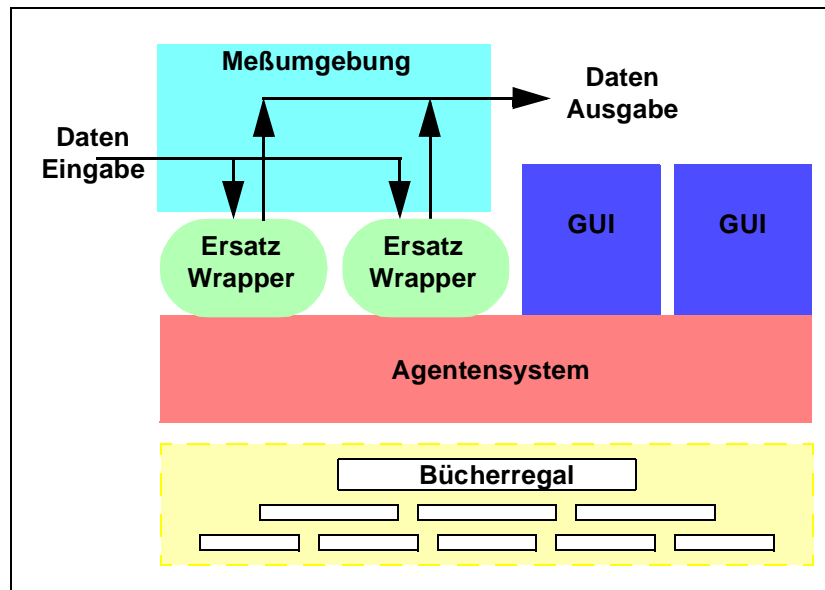


ABBILDUNG 33. Testumgebung zur Leistungsmessung des DDE Systems

Der Server wurde mit einer Konfiguration gestartet, in der alle Agenten physikalisch auf demselben Rechner liegen, so daß innerhalb des Agentensystems nicht auf das Netzwerk zugegriffen werden muß. Da der Zugriff auf Mole-DDE dennoch über einen Unix Port gemacht werden muß, wurden bei diesen Tests zumindest das "Loopback Device¹" und somit auch die Netzwerk Protokollschichten in gewisser Weise mit gemessen.

Es gab verschiedene Parameter, die während der Tests eingesetzt wurden, um eine optimale Konfiguration des Servers herauszufinden. Das Konfigurationsprogramm des

1. Jeder Rechner im Internet hat eine private nur für ihn gültige IP Adresse über die der gesamte lokale Netzwerkverkehr abgewickelt wird.

MoleOffice Servers wurde so angelegt, daß man dort unter anderem die folgenden Parameter einstellen kann:

- **Thread-Sleep-Cycle Time:**
Beschreibt die Zeit, wie lange ein einzelner Thread sich beim Lesen und Schreiben von Informationen “*schlafen legt*”, falls gerade keine Informationen vorliegen.
Ist er relativ groß, kommt es zu unnötig langen Wartezeiten, bis dieser Thread im Falle von ankommenden Daten reagieren kann. Ist er zu klein, ergibt sich eine erhöhte und ebenfalls unnötige Prozessorlast¹.
- **Heartbeat-Cycle Time:**
Beschreibt die Häufigkeit der Überprüfungen des DDE Agenten, ob ein Wrapper Informationen sendet. Auch hier hat eine zu große Wartezeit zur Folge, daß MoleDDE nur sehr träge auf ankommende Informationen reagieren kann. Ist dieser Parameter zu niedrig, überprüft der DDE Agent beispielsweise noch während der Verarbeitung der vorherigen Daten den gleichen Wrapper nochmals.
- **Buffer-Size:**
Zwischen Sender und Empfänger befinden sich einige Puffer, deren Größe entscheidend für die Höhe des datendurchsatzes ist. Ein Puffer, der nur zu wenigen Prozent gefüllt wird, verbraucht allerdings unnötig viel Speicherplatz.

Je nach Ausrichtung dieser Werte benötigt der Server mehr oder weniger Speicher oder Rechenzeit. Ab gewissen Grenzen war ein Optimieren allerdings nicht weiter sinnvoll, da je nach Situation eine unterschiedliche Parametrisierung besser dazu geeignet ist, ein optimales Testergebnis zu erzielen.

1. Normalerweise würde man die Thread durch blockierende Aufrufe implementieren. Hierbei kam es allerdings zu unkontrollierten Überläufen der internen Puffer der Socket Implementierung im JDK 1.1.x

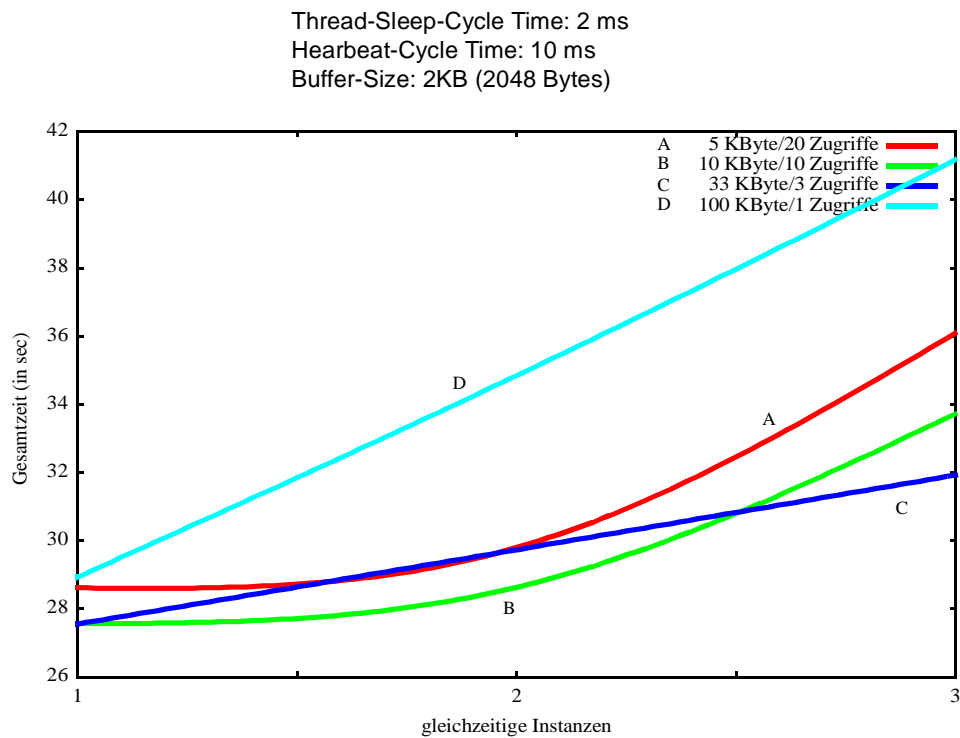


ABBILDUNG 34. Auswirkung mehrerer Klienten

Besonders eindrucksvoll, sind die Ergebnisse bei parallelen Zugriffen mehrerer Klienten. Hierzu wurden sowohl der Server als auch bis zu 3 Klienten-Testumgebungen gleichzeitig auf einem Rechner gestartet und die Zeit zwischen An- und Abmelden der Klienten gemessen. Abbildung 34 stellt das Zugriffsverhalten bei 4 verschiedenen Ausgangssituationen dar. Jeder Klient mußte die folgenden Aufgaben tätigen:

- am System anmelden
- Kapitel öffnen
- zwischen 1 und 20 mal den Inhalt eines Kapitels lesen
- Kapitel schließen
- vom System abmelden

Dabei wurde die Dateigröße des zu lesenden Dokuments so gewählt, daß die Werte miteinander vergleichbar sind. Beispielsweise wurde bei einer Dateigröße von 5 KB 20 mal zugegriffen, bei einer von 33 KB nur 3 mal. Vor allem die gekrümmten Kurven (A und B) bei häufigen, kleinen Zugriffen sind zu erwähnen, da man hier am besten die Auswirkungen einer starken Belastung des gesamten Agentensystems beschreiben kann. Für jede Anfrage an das DDE System werden eine große Anzahl weiterer

Kommunikationsschritte zwischen den Agenten angeregt. Da auch die hier ausgetauschten Informationen sowie der Zugriff auf die Agentenmethoden miteinander synchronisiert werden müssen, kommt es ab einem bestimmten Schwellenwert zu einer Überlast mit Verzögerungen, vor allem beim Einsatz mehrerer Klienten. Prinzipiell sind kleine Anfragen trotzdem besser parallelisierbar, wie der Vergleich der Steigungen der Kurven mit den Dateigrößen 33 KB und 100 KB zeigt (C und D).

Wenn man beachtet, daß die Y-Achse nicht bei 0 Sekunden beginnt, sieht man, daß die Kurven mit der Erhöhung der Anzahl von parallel zugreifenden Klienten nur extrem flach ansteigen. Der Gesamtdatendurchsatz, gemessen durch die Addition des Durchsatzes aller Klienten pro Zeiteinheit, ist also wesentlich höher als bei nur einem Klienten. Das bedeutet, daß die Belastungen durch gleichzeitige Zugriffe auf den Server eher gering sind. Dies gilt allerdings nur unter der Bedingung, daß nicht eine zu hohe Anzahl an parallelen Anfragen gleichzeitig anliegen.

Auf die Praxis übertragbar sind diese Messungen nicht uneingeschränkt, sie stellen jedoch einen Belastungstest dar. Die meiste Zeit verbringt ein angemeldeter Benutzer sicherlich damit, ein Dokument zu lesen oder es zu schreiben. Er wird es nicht andauernd öffnen und wieder schließen.

5.4 Editor Umgebung

Es ist nicht möglich, objektive Aussagen über das gesamte System inklusive des Editors vorzunehmen. Dazu würde ein Modul benötigt, das eine Kooperation des Emacs mit anderen Programmen ermöglicht. Java bietet hier die Möglichkeit, mit “*Beans*” zu arbeiten, der Emacs offeriert hingegen nichts Vergleichbares.

Da es also nicht möglich war, den Emacs durch andere Programme gezielt zu steuern, bestand nur die Möglichkeit, subjektive Einschätzungen und von Hand gemachte Messungen vorzunehmen.

Bei diesen zum Teil subjektiven Messungen stellt sich jedoch die Frage, welche Ergebnisse für ein solches System wichtig und vor allem aussagekräftig sind. Sicherlich sind relativ kurze Antwortzeiten beim “*browsen*” durch die Bücherregalstruktur extrem wichtig. Hier macht sich in besonderer Weise eine kurze Reaktionszeit innerhalb von Mole-DDE und dem Wrapper bemerkbar. Durch relativ niedrig angesetzte Zeiten für

die in Abschnitt 5.3 aufgeführten Parameter lassen sich hier Antwortzeiten realisieren, die nicht mehr mit den hier eingesetzten Mitteln meßbar waren. Auffallend sind die Wartezeiten hingegen beim Laden und Speichern von großen Dokumenten jenseits von etwa 100 KB. Die folgende Messungen wurden durchgeführt:

- Kapitel öffnen
- den Inhalt eines Kapitels lesen
- zwischen 1 und 10 mal:
 - den Inhalt des Kapitels schreiben.
 - den Inhalt des Kapitels lesen.
- Kapitel schließen
- Inhaltsverzeichnis des zugehörigen Buchs lesen

Die Messung des zehnfachen Lese-Schreibzykluses bei einem Datenaufkommen von über 2 MB ergab auf dem Pentium II eine Zeit von etwa 36 Minuten und auf der Sparc Workstation etwa 24 Minuten.

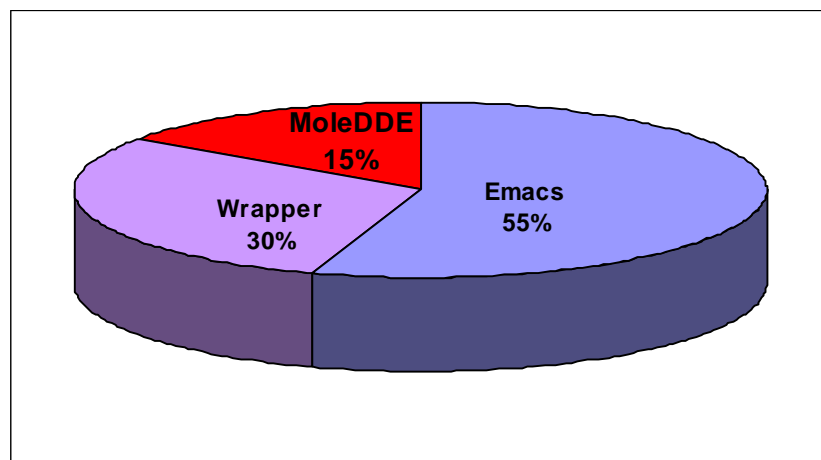


ABBILDUNG 35. Prozentuale Verteilung der Bearbeitungszeit

Durch den direkten Vergleich der Ergebnisse mit und ohne Editor bzw. Wrapper erhält man als Resultat, daß der Emacs selbst relativ lange braucht, die erhaltenen Ergebnisse zu verarbeiten. Dies liegt vor allem an den beiden folgenden Eigenschaften:

- Die in Lisp geschriebenen Module für den Emacs müssen stets interpretativ verarbeitet werden.
- Ein Emacs Modul durchläuft jeden erhaltenen Text, um den Bearbeitungszustand durch unterschiedliche Schriften zu visualisieren.

Außerordentlich groß ist auch die Zeit, die der Wrapper zur Verarbeitung und zum Transport der Informationen benötigt. Hier könnte sicherlich ein besserer Kommunikationsmechanismus (RMI), Proxy-Optimierungen oder eine Komprimierung der Daten die Ergebniswerte verbessern.

5.5 Vergleich mit CorbaDDE

CorbaDDE und MoleDDE verwenden von der Leistung her gesehen vergleichbare Versionen des Editors Emacs. Da kein Modul für den direkten Test von CorbaDDE ohne Emacs und keine anderen Testergebnisse vorlagen, mußte auch hier auf einen eher subjektiven Vergleichstest zurückgegriffen werden.

Zuerst wurde auch Corba-DDE daraufhin überprüft, wie gut ein flüssiges Arbeiten möglich ist. Leider bietet Corba-DDE einige der hier beschriebenen Funktionen nicht an. So ist es beispielsweise nicht möglich, ein geöffnetes Kapitel auch wieder zu schließen. Auch ist die hier verwendete Version des Emacs nicht dazu imstande, beliebig im Bücherregal zu *“browsen”*. Es ist nur eine Art Einbahnstraße vom Bücherregal hin zu den Kapiteln vorhanden.

Die Anmeldung des Editors an Corba-DDE dauert im Vergleich mit Mole-DDE relativ lange. Dies liegt wahrscheinlich daran, daß bei Mole-DDE der Großteil der Anmeldung bereits beim Start der grafischen Oberfläche geschieht. Auch scheint die Reaktionszeit unter Corba-DDE schlechter zu sein. Es war mir nicht möglich zu überprüfen, inwieweit Corba selbst hierfür verantwortlich ist. Beim Laden und Speichern von großen Dokumenten erwies sich Corba-DDE hingegen als etwas schneller. Hier zeichnet sich eine ausgereifere Kommunikation des Wrappers über Corba aus.

Da nur durch den Einsatz vom Emacs getestet werden konnte, war es nicht möglich, Messungen mit parallelen Zugriffen mehrerer Klienten durchzuführen.



Durch Einsatz umfangreicher Testumgebungen konnte die Qualität der verschiedenen Komponenten von Mole-DDE verifiziert werden.

KAPITEL 6

INSTALLATION UND KONFIGURATION

Die Installation und Konfiguration von Mole-DDE beziehungsweise MoleOffice¹ läßt sich ebenfalls in die Bereiche Klient und Server aufspalten. Der Server muß systemweit genau einmal existieren, wohingegen der Klient einmal für jeden Benutzer gestartet werden muß. Im Laufe dieser Arbeit wurden, wie teilweise schon erwähnt, einige Veränderungen an den Möglichkeiten der Konfiguration des Systems vorgenommen.

6.1 Installation von Mole-DDE

Um Mole-DDE einsetzen zu können, bedarf es für den Server mindestens der folgenden Systemvoraussetzungen:

- Unix oder Linux
- Burn Again Shell (bash)
- JDK in der Version 1.1.x
- 20 MB Festplattenkapazität
- 8 MB Hauptspeicher
- 50 MHz Taktfrequenz

Für den Klient benötigt man als Mindestvoraussetzung:

- Unix, Linux, Windows 9x oder Windows NT
- JDK in der Version 1.1.x oder 1.2.x
- GNU Emacs ab Version 20.4.1 oder XEmacs ab Version 21.1
- 20 MB Festplattenkapazität (nur bei nicht vorhandenem Server)
- 32 MB Hauptspeicher
- 200 MHz Taktfrequenz

Für ein flüssiges Arbeiten empfiehlt sich allerdings der Einsatz etwas besser ausgestatteter Rechner.

1. Für dieses Kapitel sind die Begriffe gleichbedeutend und bezeichnen das gesamte Projekt, wie es einem etwaigen Benutzer vorliegt.

Archiv	Bedeutung
1) classes.tgz	vorkompilierte Klassen für den direkten Einsatz
2) config.tgz	alle für den Betrieb nötigen Konfigurationsdateien einschließlich Dokumentation
3) src.tgz	Java-Quelldateien von Mole-DDE
4) da.tgz	Das vorliegende Dokument

Tabelle 6. Archive von Mole-DDE

Mole-DDE existiert in Form von 4 Archiven. Für den erfolgreichen Einsatz von Mole-DDE benötigt man auf jeden Fall das Archiv 2) und eines der Archive 1) oder 3). Die Archive enthalten sowohl die Dateien für den Server, als auch die für den Klienten.

Zur Installation entpackt man die gewünschten Archive einfach in einem Verzeichnis seiner Wahl (von nun an *DIR* genannt).

6.2 Konfiguration des Server

Die Konfiguration des MoleOffice Servers geschieht in mehreren Schritten

6.2.1 Mole

Zur lokalen Anpassung von Mole existiert die Datei "*DIR/config/molerc*". Diese kopiert man in sein eigenes Heimatverzeichnis und paßt sie entsprechend der Mole Dokumentation an.

Im Verzeichnis "*DIR/config*" muß eine Datei mit dem Namen "*hostname.cfg*" existieren, wobei "*hostname*" durch den Rechnernamen, auf dem der Server laufen soll, zu ersetzen ist.

Ebenfalls angepaßt werden muß die Datei "*locations.dat*", die sich im selben Verzeichnis befindet. Hier findet eine Zuordnung der IP-Adressen zu den Rechnernamen statt.

6.2.2 Laufzeitumgebung

Zur Konfiguration der Laufzeitumgebung von Mole-DDE existiert die Datei "*DIR/config/classpath*". Auch diese Datei kopiert man in das eigene Heimatverzeichnis und paßt sie anhand der Vorlage an.

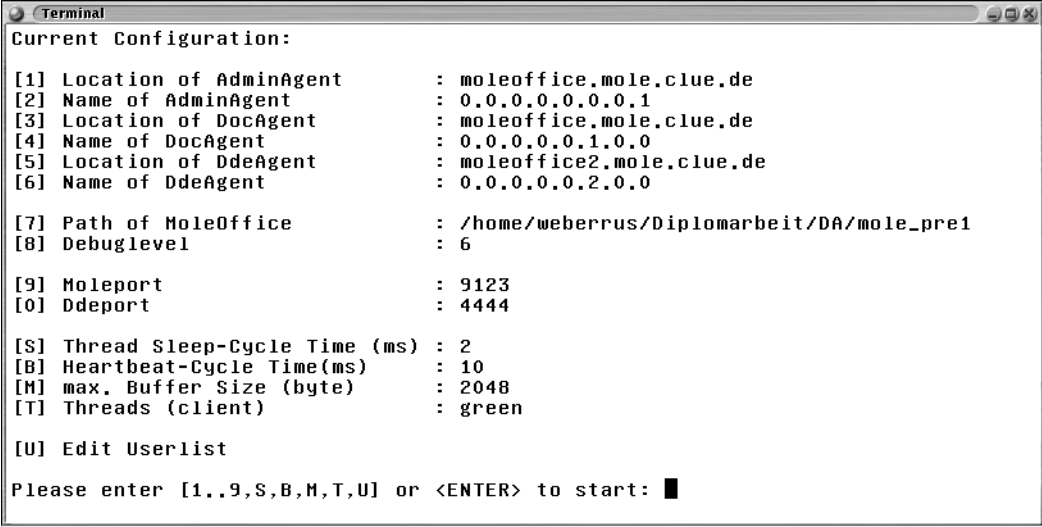
6.2.3 Neuübersetzung

Will man das System neu übersetzen, muß man zusätzlich noch die Dateien "Makefile.config" und "Makefile.standards" im Verzeichnis "DIR/etc" anpassen und anschließend das System mit dem Aufruf von make im Verzeichnis "DIR/src/mole" neu übersetzen.

6.2.4 MoleOffice Konfigurationsprogramm

Im Verzeichnis "DIR/bin" existiert ein Programm namens "mo". Mit diesem Programm können alle weiteren Einstellungen am MoleOffice Server vorgenommen werden. Durch die Benutzung des Programms ohne Parameter erhält man eine kurze Hilfe.

Der Aufruf von "mo config" öffnet einen Konfigurationsdialog. Das Programm versucht zwar, die wichtigsten Parameter zu "erraten". Es sollten aber vor allem die Lokationen der Systemagenten überprüft und angepaßt werden.



```
Terminal
Current Configuration:
[1] Location of AdminAgent      : moleoffice.mole.clue.de
[2] Name of AdminAgent         : 0.0.0.0.0.0.0.1
[3] Location of DocAgent       : moleoffice.mole.clue.de
[4] Name of DocAgent           : 0.0.0.0.0.1.0.0
[5] Location of DdeAgent       : moleoffice2.mole.clue.de
[6] Name of DdeAgent           : 0.0.0.0.0.2.0.0

[7] Path of MoleOffice         : /home/weberrus/Diplomarbeit/DA/mole_pre1
[8] Debuglevel                 : 6

[9] Moleport                   : 9123
[0] Ddeport                    : 4444

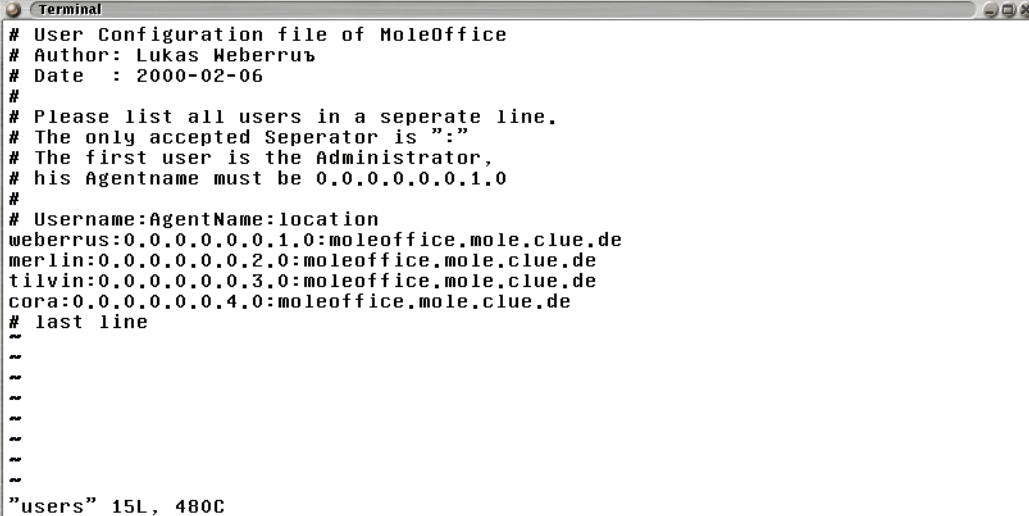
[S] Thread Sleep-Cycle Time (ms) : 2
[B] Heartbeat-Cycle Time(ms)    : 10
[H] max. Buffer Size (byte)     : 2048
[T] Threads (client)           : green

[U] Edit Userlist

Please enter [1..9,S,B,H,T,U] or <ENTER> to start: █
```

ABBILDUNG 36. Konfigurationsdialog von MoleOffice

Falls die Standard-Ports 9123 und 4444 im System schon belegt sein sollten können hier auch beliebig andere Werte eingetragen werden. Für den Debuglevel existieren Werte zwischen 0 und 12, wobei bei höheren Werten mehr Ausgaben erzeugt werden.



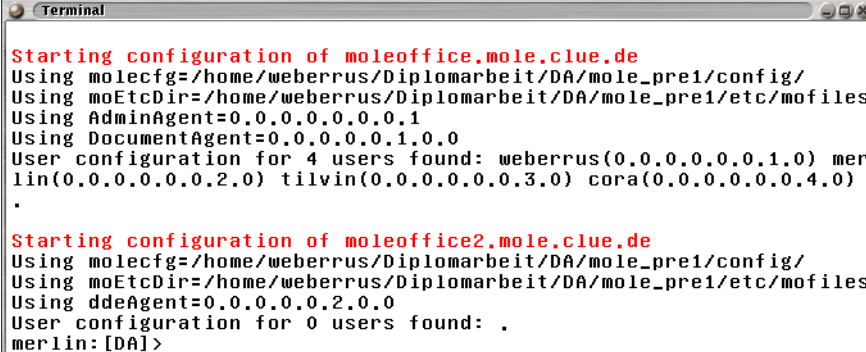
```

Terminal
# User Configuration file of MoleOffice
# Author: Lukas Weberuss
# Date : 2000-02-06
#
# Please list all users in a seperate line.
# The only accepted Seperator is ":"
# The first user is the Administrator,
# his Agentname must be 0.0.0.0.0.1.0
#
# Username:AgentName:location
weberrus:0.0.0.0.0.0.1.0:moleoffice.mole.clue.de
merlin:0.0.0.0.0.0.2.0:moleoffice.mole.clue.de
tilvin:0.0.0.0.0.0.3.0:moleoffice.mole.clue.de
cora:0.0.0.0.0.0.4.0:moleoffice.mole.clue.de
# last line
..
..
..
..
..
..
..
..
"users" 15L, 480C

```

ABBILDUNG 37. Benutzerkonfiguration von MoleOffice

Die Liste der Benutzer sollte ebenfalls von hier aus editiert werden. Der erste Eintrag stellt den Administrator dar. Die erste Spalte stellt lediglich den im System verwendeten Referenznamen dar. Die zweite Spalte ordnet jedem Benutzer seinen persönlichen Agenten zu. In der dritten Spalte wird schließlich die Lokation dieser Agenten angegeben.



```

Terminal
Starting configuration of moleoffice.mole.clue.de
Using molecfg=/home/weberrus/Diplomarbeit/DA/mole_pre1/config/
Using moEtcDir=/home/weberrus/Diplomarbeit/DA/mole_pre1/etc/mofiles
Using AdminAgent=0.0.0.0.0.0.1
Using DocumentAgent=0.0.0.0.0.1.0.0
User configuration for 4 users found: weberrus(0.0.0.0.0.0.1.0) merlin(0.0.0.0.0.0.2.0) tilvin(0.0.0.0.0.0.3.0) cora(0.0.0.0.0.0.4.0)
.
Starting configuration of moleoffice2.mole.clue.de
Using molecfg=/home/weberrus/Diplomarbeit/DA/mole_pre1/config/
Using moEtcDir=/home/weberrus/Diplomarbeit/DA/mole_pre1/etc/mofiles
Using ddeAgent=0.0.0.0.0.2.0.0
User configuration for 0 users found: .
merlin:[DA]>

```

ABBILDUNG 38. Konfigurationsergebnisse

Wurde MoleOffice erfolgreich eingerichtet, sollte ein Ergebnis, ähnlich dem aus Abbildung 38 zu sehen sein. Das Mole Setup durchläuft hier alle für diesen Rechner gefunden Lokationen und füllt diese mit den dafür konfigurierten Systemagenten.

6.2.5 Starten und Beenden des Servers

Nachdem die Konfiguration des Systems erfolgreich abgeschlossen wurde, kann nun der Server mit dem Kommando “*mo start*” gestartet werden. Hierbei starten die Systemagenten selbst alle benötigten mobilen Agenten. Das Kommando “*mo see*” gibt die Ausgabe der Agenten auf dem Bildschirm aus. Beim ersten Lauf sollte vor allem auf alle roten Ausgaben geachtet werden, da diese Fehlermeldungen der Agenten signalisieren. Bei einer ungeschickten Wahl der Namen der Systemagenten kann es zu Kollisionen mit den von Mole automatisch generierten Namen der mobilen Agenten kommen.

Mit “*mo stop*” läßt sich der MoleOffice Server jederzeit wieder beenden.

6.2.6 Konfiguration des Klienten

Wenn sichergestellt ist, daß der Server richtig konfiguriert ist, kann nun für jeden Benutzer anhand einer HTML-Datei der Klient konfiguriert werden.

```

<APPLET

CODEBASE="file:/home/weberrus/Diplomarbeit/DA/mole_pre1/classes"

CODE="mole.moleschedule.Gui"

WIDTH=605 HEIGHT=630 NAME="Gui">

<PARAM NAME=AgentName      Value="0.0.0.0.0.0.1.0">
<PARAM NAME=LocationName  Value="merlin.clue.de">
<PARAM NAME=LocationPort  Value="9123">
<PARAM NAME=Beep           Value="true">
<PARAM NAME=EmacsPath     Value="/home/weberrus/Diplomarbeit/
DA/mole_pre1/src/mole/dde/emacs">
<PARAM NAME=DiskWriter    Value="/home/weberrus/bin/diskwri-
ter">

</APPLET>

```

ABBILDUNG 39. Konfigurationsbeispiel eines MoleOffice Klienten

Neu sind hier die beiden Einträge “*EmacsPath*” und “*Diskwriter*”. “*EmacsPath*” zeigt auf das Verzeichnis, in dem sich die Zusatzmodule für den Emacs befinden. “*Diskwriter*” gibt das Programm zum Schreiben auf die Festplatte an. Wichtig ist vor allem der

Eintrag des Namens seines persönlichen Agenten. “*LocationName*” bezeichnet nicht die Lokation des Agenten, sondern den Rechnernamen des MoleOffice Servers.

6.2.7 Starten und Beenden des Klienten

Ebenso wie der Server, kann auch der Klient über das Programm “*mo*” gestartet werden. Das Kommando “*mo client -jdk1.2 me*” startet zum Beispiel den *APPLETVIEWER* in der Version 1.2.x mit der HTML-Konfigurationsdatei “*me.html*”. Bei einer Fehlfunktion kann unter “*DIR/debug/messages*” geprüft werden, aus welchem Grund der Start mißlungen ist.

Durch Beenden des Applets meldet sich der Klient automatisch vom System ab. Es ist allerdings weiterhin möglich, mit einem geöffneten und angemeldeten Editor zu arbeiten. Eine erneute Anmeldung des Editors ist jedoch nicht möglich.

KAPITEL 7

ZUSAMMENFASSUNG

7.1 Bewertung

Mole-DDE beschreibt *einen* Weg, ein Verteiltes Dokumentensystem auf der Basis mobiler Agenten zu entwerfen. In dieser Arbeit wurde ein prototypisches DDE-System entwickelt und untersucht, in wie weit sich das mobile Agentensystem Mole eignet, um als Grundlage für derartige Entwicklungen eingesetzt zu werden. Gerade im Vergleich zu Corba-DDE zeigen sich die Vor- und Nachteile dieser Arbeit sehr deutlich.

Bei Corba-DDE kann man die Vorteile eines ausgereiften kommerziellen Produkts wie Corba nutzen. Dies zeigt sich insbesondere beim Entwurf von Schnittstellen über IDL¹. Wie Corba-DDE erlaubt auch Mole-DDE die einfache Anbindung weiterer Editoren. Parallel zu dieser Arbeit wurde in einer Studienarbeit von Bo Wu² gezeigt, daß man auch Editoren, die auf mobilen Endgeräten laufen, an Corba-DDE anschließen kann. Ebenso zeigt die Diplomarbeit von Alexander Kramer³, wie man auch unter Mole-DDE die Kommunikation mit mobilen Geräten über das IRDA-Protokoll verwirklichen kann.

Schon in seiner prototypischen Form zeigt Mole-DDE, daß mit einem mobilen Agentensystem eine Verteilte Dokumentenverwaltung verwirklicht werden kann. Es ist möglich, über die Struktur eines Bücherregals Dokumente zu verwalten, Kapitel und Bücher anzulegen, zu löschen, während der Nutzung zu sperren und diese wieder freizugeben. Hierzu dient der Einsatz des Editors Emacs, kombiniert mit einer separaten grafischen Benutzungsoberfläche. Durch die Möglichkeit, über Rechnergrenzen hinweg ein System von Agenten aufzubauen, zeichnet sich Mole besonders im Hinblick auf Transparenz und Lastbalancierung aus.

1. Interface Definition Language

2. Einbau weiterer Desktop Publishing Systeme in DDE [WuBo2000]

3. Benachrichtigung von Teamkollegen über Erreichbarkeit mittels mobiler Geräte [KrAl1999]

7.2 Ausblick

Ein wichtiges Entwurfskriterium war stets die Möglichkeit zur Erweiterung und Veränderung des Systems. Die Komponenten des Systems wurden so modular angelegt, daß sie sowohl leicht um Funktionen ergänzt als auch vollständig ersetzt werden können.

Einige der in Corba-DDE möglichen Funktionen, wie das Herausnehmen von Dokumenten und das Einbringen fremder Dokumente, müssen zukünftig auch in Mole-DDE eingebracht werden. Um bei längeren Pausen automatisch ein gesperrtes Dokument temporär frei zugeben muß eine Erweiterung des Wrappers erfolgen.

Besonders beim Einsatz größerer Dokumente hat sich gezeigt, daß die Kommunikation zwischen dem Editor, dem Wrapper und dem DDE-System ein Nadelöhr beim Informationsfluß darstellt. Eventuell kann hier der Einsatz von entfernten Methodenaufrufen die Leistung maßgeblich erhöhen.

Bisher unterscheidet das DDE-System beim Inhalt von Dokumenten nicht, um welche Art von Information es sich handelt. Die Verwendung von XML¹ könnte hier die Integration beliebiger auch multimedialer Dokumente ermöglichen.

Auch das Agentensystem könnte man unter dem Gesichtspunkt einer Lastbalancierung noch erweitern. So ist es beispielsweise denkbar, den DDE Agenten und den Dokumentenagenten pro Lokation bereitzustellen, um kürzere physikalische Kommunikationswege zu erreichen. Die Agenten müßten sich dann untereinander über mobile Agenten synchronisieren.

Um einen besseren konkurrierenden Zugriff zu ermöglichen, ist es notwendig, innerhalb des Dokumenten Agenten eine erweiterte Synchronisation der Bücherregalinhalte zu realisieren. Zur besseren Unterscheidung der Befugnisse und Möglichkeiten der einzelnen Benutzer sollte zukünftig ein auf Rollen basierender Zugriffsmechanismus eingeführt werden.

Die Funktionalität zur Bearbeitung des Bücherregals könnte völlig von der Autoren-umgebung gelöst werden. Die grafische Oberfläche müßte dann aber um einige Komponenten zur direkten Manipulation des Bücherregal erweitert werden. Auch die

1. Extensible Markup Language

Integration weiterer Editoren in Mole-DDE könnte durch diesen Schritt erheblich vereinfacht werden, da die Aufgabe der Autorenumgebung nur noch auf die direkte Bearbeitung eines Dokumentes beschränkt wäre.

Um in einer Zeit, in der die Sicherheit persönlicher Daten von großer Bedeutung ist, ein Verteiltes Dokumentensystem über ein offenes Netzwerk betreiben zu können, sollte der eingesetzte Authentifizierungsmechanismus erweitert werden. Hier könnte eine Verschlüsselung des gesamten Informationsflusses von Mole die Sicherheit erhöhen.

KAPITEL 8

ANHANG

8.1 Literaturverzeichnis

- [BuCo1997] Cora Burger, **Groupware - Kooperationsunterstützung für Verteilte Anwendungen**, dpunkt, Heidelberg, 1997
- [BuSc1999] Cora Burger und Oliver Schramm, **Co-authoring in Dynamic Teams with Mobile Individuals**, International Symposium on Handheld and Ubiquitous Computing (HUC99), 1999
- [WoMi1999] Michael Wooldridge, **Intelligent Agents**, MIT Press, Cambridge, 1999
- [SpHa1999] Tammo Spalink und John Hartmann, **The Effects of a Mobile Agent on File Service**, Department of Computer Science, Universität von Arizona, USA, 1999
- [HoFr1995] Fritz Hohl, **Konzeption eines einfachen Agentensystems und Implementation eines Prototyps**, Universität Stuttgart, Fakultät Informatik, Diplomarbeit Nr. 1267, 1995
- [ZiJö1996] Jörg Zimmer, **Benutzeragenten zur Unterstützung von Teamkoordination**, Universität Stuttgart, Fakultät Informatik, Diplomarbeit Nr. 1421, 1996
- [TrSv1997] Sven Tränkle, **Realisierung eines CSCW-Benachrichtigungsdienstes mit MOLE-Agenten**, Universität Stuttgart, Fakultät Informatik, Diplomarbeit Nr. 1685, 1997
- [ScOl1999] Oliver Schramm, **Entwurf eines verteilten corbabasierten Dokumentensystems**, Universität Stuttgart, Fakultät Informatik, Diplomarbeit Nr. 1708, 1999
- [ScPe1999] Peter W. Schurr, **Erweiterung von DDE um Konsistenzverhandlung in mobiler Umgebung**, Universität Stuttgart, Fakultät Informatik, Diplomarbeit Nr. 1778, 1999
- [KrAl1999] Alexander Kramer, **Benachrichtigung von Teamkollegen über Erreichbarkeit mittels mobiler Geräte**, Universität Stuttgart, Fakultät Informatik, Diplomarbeit Nr. 1777, 1999
- [WuBo2000] Bo Wu, **Einbau weiterer Desktop Publishing Systeme in DDE**, Universität Stuttgart, Fakultät Informatik, Studienarbeit Nr. 1764, 2000

- [GNU1991] GNU Emacs Manual Edition 2.0, 1991
- [PfHo1998] Christoph Pfisterer und Fritz Hohl, **The Mole Cookbook or How to program a Mole agent**, <http://mole.informatik.uni-stuttgart.de/docs/cookbook.html>, 1998
- [SuMi1999] Sun Microsystems Inc., **Java-Online-Dokumentation**, <http://java.sun.com/doc/>, 1999
- [SuMi1996] Sun Microsystems Inc., **Security for the Java™ Platform**, <http://www.sun.com/960901/feature3/javasecure.html>, 1996
- [KoDa1998] Daniel Kowalewski, **Signieren von Applets**, <http://java.rzrn.uni-hanover.de/jug/sigapplets>, 1998
- [GrDa1999] Daniel Grimscom, **Code Signing for Java Applets**, http://www.suitable.com/Doc_CodeSigning.shtml, 1999
- [BoJo1997] Joseph Bowbeer, **Signing Applets for Internet Explorer and Netscape Navigator**, <http://ourworld.compuserve.com/homepages/jozart/article/index.html>, 1997
- [MiCo1999] Microsoft Corporation, **Trust Based Security for Java**, <http://www.microsoft.com/Java/security/default.htm>, 1999

8.2 Code Beispiele

In diesem Kapitel sollen einige kleinere typische Codefragmente aus Mole-DDE gezeigt werden.

```
protected synchronized void sendMessage(
    AgentName targetName, LocationName targetLoc,
    Serializable message)
{
    Location loc=getCurrentLocation();
    Object[] content=null;
    Message m=new Message(
        getName(),loc.locationName(), // Sender
        targetName,targetLoc,         // Recipient
        0,                             // Error Semantic
        message);                      // Content
    loc.message(m);
}
```

ABBILDUNG 40. Methode sendMessage

```
protected synchronized void callRemote
    (AgentName targetName, LocationName targetLoc,
    String method, Object[] param)
{
    Location loc=getCurrentLocation();
    Object[] content=null;
    if (param==null) {
        content=new Object[1];
    }
    else {
        content=new Object[param.length+1];
        for (int i=1;i<content.length;i++)
            content[i]=param[i-1];
    }
    content[0]=method;
    sendMessage(targetName,targetLoc,content);
}
```

ABBILDUNG 41. Methode callRemote

```
public void receiveMessage(Message m)
{
    ddeio.debug(
        "FilterAgent: I received a formatted message from "+
        m.sender.toString(),ddeio.am,8);

    Object[] param = (Object[])m.content;
    Method method = null;
    String methodName = null;
    Class[] argtypes = null;
    Object[] args = null;

    try {
        // find the right method
        methodName=(String)param[0];

        ddeio.debug(
            "FilterAgent: got call for "+
            methodName,ddeio.am,2);

        argtypes = new Class[param.length-1];
        for (int i=1;i<param.length;i++)
            argtypes[i-1]=param[i].getClass();

        method = this.getClass().getMethod(
            methodName,argtypes);

        // invoke method and receive the result
        ddeio.debug("FilterAgent: calling :"+
            method.toString(),ddeio.am,2);
        args=new Object[param.length-1];
        for (int i=1;i<param.length;i++)
            args[i-1]=param[i];
        Object result=method.invoke(this,args);

        ddeio.debug(
            "FilterAgent: "+method.toString()+
            " successfully executed.",ddeio.am,4);
        ddeio.debug(
            "FilterAgent result was "+result,ddeio.am,4);
    }
}
```

ABBILDUNG 42. Methode receiveMessage (Teil 1)

```

        // bring the result back to sender of message
        if (result!=null) {
            ddeio.debug(
                "FilterAgent: sending result back "+
                "to sender.", ddeio.am,2);
            Object [] rargs=new Object[1];
            rargs[0]=result;
            callRemote(m.sender,m.senderlocation,
                "receiveResult",rargs);
        }
    }

    catch(NoSuchMethodException e) {
        String p=new String("");
        for (int i=0;i<argtypes.length;i++)
            p=p+argtypes[i].toString()+", ";

        ddeio.debug("FilterAgent: No such Method: "+
            methodName+"("+p+") !", ddeio.ae,10);
    }

    catch(Exception e) {
        ddeio.debug(
            "FilterAgent: Caught Exception in "+
            "receiveMessage(Message m): "+
            e.toString(),ddeio.ae,10);

        if (e instanceof
            java.lang.reflect.InvocationTargetException) {
            ddeio.debug("FilterAgent: Exception was:"+
                ((InvocationTargetException)e).
                getTargetException(),ddeio.ae,8);
            e.printStackTrace();
        }
    }
}

```

ABBILDUNG 43. Methode receiveMessage (Teil 2)

8.3 Klassenhierarchie

Die Klassenhierarchie von MoleOffice teilt sich in mehrere Pakete. Hier sollen kurz die wichtigsten aufgeführt werden. Genauere Informationen zu den einzelnen Klassen findet man in der Online-Dokumentation

8.3.1 mole.dde

- class java.lang.Object
 - class mole.Agent (implements java.io.Serializable)
 - class mole.SystemAgent
 - class mole.dde.DdeAgent (implements mole.Periodical)
 - class mole.dde.DocumentAgent
 - class mole.UserAgent
 - class mole.dde.FilterAgent (implements mole.MobileAgent)
 - class mole.dde.BookShelfTestmole.dde.util

8.3.2 mole.dde.util

- class java.lang.Object
 - class mole.dde.util.Action (implements java.io.Serializable)
 - class mole.dde.util.ddeBook (implements java.io.Serializable)
 - class mole.dde.util.ddeBookShelf (implements java.io.Serializable)
 - class mole.dde.util.DdeFilter (implements java.io.Serializable)
 - class mole.dde.util.ddegraph
 - class mole.dde.util.ddeio
 - class mole.dde.util.ddeSection (implements java.io.Serializable)
 - class mole.dde.util.ddestring
 - class mole.dde.util.linkObject (implements java.io.Serializable)
 - class mole.dde.util.objectUsers
 - class mole.dde.util.SecurityContext
 - class java.lang.Thread (implements java.lang.Runnable)
 - class mole.dde.util.DisconnectThread
 - class java.lang.Throwable (implements java.io.Serializable)
 - class java.lang.Exception
 - class mole.dde.util.MalFormattedMessage
 - class mole.dde.util.UnknownMessageType
 - class mole.dde.util.userObjects
 - class mole.dde.util.var

8.3.3 mole.dde.gui

- class java.lang.Object
 - class java.awt.Component (implements java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable)
 - class java.awt.Container
 - class javax.swing.JComponent (implements java.io.Serializable)
 - class mole.dde.gui.JBild
 - class java.awt.Window
 - class java.awt.Frame (implements java.awt.MenuContainer)
 - class javax.swing.JFrame (implements javax.accessibility.Accessible, javax.swing.RootPaneContainer, javax.swing.WindowConstants)
 - class mole.dde.gui.DdeGui (implements java.awt.event.ActionListener, java.awt.event.ItemListener)
 - class mole.dde.gui.DdeGuiScreen (implements java.awt.event.ActionListener, java.awt.event.ItemListener, java.awt.event.MouseListener, java.lang.Runnable)
 - class mole.dde.gui.DdeGuiSetup (implements java.awt.event.ActionListener, java.awt.event.ItemListener)
 - class mole.dde.gui.PasswordCheck (implements java.awt.event.ActionListener, java.awt.event.KeyListener, java.lang.Runnable)
 - class mole.dde.gui.PasswordScreen (implements java.awt.event.ActionListener, java.lang.Runnable)
 - class javax.swing.JWindow (implements javax.accessibility.Accessible, javax.swing.RootPaneContainer)
 - class mole.dde.gui.DdeGuiW (implements java.awt.event.ActionListener, java.awt.event.ItemListener)
 - class mole.dde.gui.Startup (implements java.lang.Runnable)
 - class java.awt.event.WindowAdapter (implements java.awt.event.WindowListener)
 - class mole.dde.gui.MiniWindowListener

8.3.4 mole.dde.wrapper

- class java.lang.Object
 - class mole.dde.wrapper.IE
 - class mole.dde.wrapper.PortServer
 - class mole.dde.wrapper.PortToStdout
 - class java.lang.Thread (implements java.lang.Runnable)
 - class mole.dde.wrapper.ReadWriteThread
 - class mole.dde.wrapper.SocketThread
 - class mole.dde.wrapper.wrapper (implements java.lang.Runnable)

8.3.5 mole.moleschedule

- class java.lang.Object
 - class java.util.AbstractCollection (implements java.util.Collection)
 - class java.util.ArrayList (implements java.util.List)
 - class java.util.Vector (implements java.lang.Cloneable, java.util.List, java.io.Serializable)
 - class mole.moleschedule.myCmDateVector
 - class mole.moleschedule.myRoomVector (implements java.io.Serializable)
 - class mole.moleschedule.myUsersVector (implements java.io.Serializable)
 - class mole.Agent (implements java.io.Serializable)
 - class mole.SystemAgent
 - class mole.moleschedule.DummyAgent
 - class mole.moleschedule.MOAdministratorAgent
 - class mole.moleschedule.MSSystemAgent
 - class mole.UserAgent
 - class mole.moleschedule.MSMobAgent (implements mole.MobileAgent)
 - class mole.moleschedule.CmDate
 - class java.awt.Component (implements java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable)
 - class java.awt.Container
 - class java.awt.Panel
 - class java.applet.Applet
 - class mole.moleschedule.Gui
 - class mole.moleschedule.ColoredPicture
 - class mole.moleschedule.MailScreen
 - class mole.moleschedule.RoomSetup
 - class mole.moleschedule.Screen (implements java.awt.event.ActionListener, java.awt.event.ItemListener)
 - class mole.moleschedule.SetupScreen
 - class mole.moleschedule.UserSetup (implements java.awt.event.ActionListener, java.awt.event.ItemListener)
 - class mole.moleschedule.VideoScreen

- class java.awt.Window
 - class java.awt.Frame (implements java.awt.MenuContainer)
 - class mole.moleschedule.RoomInfoWindow (implements java.awt.event.ActionListener)
 - class mole.moleschedule.UserInfoWindow (implements java.awt.event.ActionListener)
 - class mole.moleschedule.YesNoWindow (implements java.awt.event.ActionListener)
- class mole.moleschedule.Convert
- class java.rmi.server.RemoteObject (implements java.rmi.Remote, java.io.Serializable)
 - class java.rmi.server.RemoteServer
 - class java.rmi.server.UnicastRemoteObject
 - class mole.moleschedule.AgentCommEntity (implements mole.moleschedule.AgentCommEntityI)
 - class mole.moleschedule.GuiCommEntity (implements mole.moleschedule.GuiCommEntityI)
- class mole.moleschedule.Room (implements java.lang.Cloneable, java.io.Serializable)
- class mole.moleschedule.RoomAndNr (implements java.io.Serializable)
- class java.lang.Thread (implements java.lang.Runnable)
 - class mole.moleschedule.UpdateWaiter (implements java.lang.Runnable)
- class mole.moleschedule.User (implements java.lang.Cloneable, java.io.Serializable)
- class mole.moleschedule.User2 (implements java.lang.Cloneable, java.io.Serializable)
- class mole.moleschedule.UserAndNr (implements java.io.Serializable)
- class mole.moleschedule.UserBoolMask (implements java.lang.Cloneable, java.io.Serializable)

8.4 GNU General Public License

GNU GENERAL PUBLIC LICENSE

Version 2, June 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc.

59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

GNU GENERAL PUBLIC LICENSE TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for

other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program),

you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a ver-

sion number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
<one line to give the program's name and a brief idea of what it does.>  
Copyright (C) <year> <name of author>
```

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

Gnomovision version 69, Copyright (C) year name of author Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type `show w'. This is free software, and you are welcome to redistribute it under certain conditions; type `show c' for details.

The hypothetical commands `show w' and `show c' should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than `show w' and `show c'; they could even be mouse-clicks or menu items--whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright interest in the program `Gnomovision' (which makes passes at compilers) written by James Hacker.

<signature of Ty Coon>, 1 April 1989 Ty Coon, President of Vice

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.

Erklärung

Ich versichere, daß ich diese Arbeit selbstständig verfaßt und nur die angegeben Hilfsmittel verwendet habe.

(Lukas Weberruß)