

Fakultät Informatik

Prüfer: Prof. Dr. Rothermel

Betreuer: Dipl.-Inform. Fritz Hohl

Beginn am : 1.12.1996

Beendet am : 31.5.1997

CR-Klassifikation: C.2.4;H.4.3;H.5.3

Diplomarbeit Nr. 1511

**Konzeption und Implementierung  
eines Active-Mail-Systems  
auf der Basis Mobiler Agenten**

Jan-Gregor Nitsche

## **Kurzfassung**

Unter Mobilien Agenten versteht man „Programmkonstrukte“, welche die Fähigkeit haben, autonom zu handeln und von Rechner zu Rechner zu migrieren, falls auf beiden Seiten ein „Agentensystem“ installiert ist. Active Mails sind elektronische Briefe, die nicht nur statische Daten, sondern auch Programme transportieren und zu bestimmten Zeitpunkten ausführen können.

In dieser Arbeit wurden zunächst verschiedene Ansätze von Active Mail Systemen vorgestellt und ein allgemeiner Anforderungskatalog eines Active Mail Systems erstellt, sowie die Vor- und Nachteile mit Mobilien Agenten verglichen.

In einem nächsten Schritt wurde aufgrund ähnlicher Eigenschaften von Active Mails und Mobilien Agenten ein System erstellt, mit dem Active Mails, auf Grundlage des Mobile-Agenten-Systems Mole, verschickt und verarbeitet werden können.

Dabei wurde die Struktur der Active Mail in Verbindung mit Mobilien Agenten entwickelt und ferner Werkzeuge realisiert, mit dem der Benutzer Active Mails empfangen, lesen und erzeugen kann. Damit das Erstellen von Active Mails erleichtert wird, wurde ein Konzept integriert, das die Benutzung vorgefertigter Active Mails ermöglicht, ohne sie neu programmieren zu müssen.

In einer Beispielanwendung (Vereinbarung eines Treffens zwischen mehreren Teilnehmern) wurde abschließend aufgezeigt, wie das erstellte System benutzt werden kann.

## Inhaltsverzeichnis

<b>ABKÜRZUNGSVERZEICHNIS</b> .....	<b>V</b>
<b>ABBILDUNGSVERZEICHNIS</b> .....	<b>VI</b>
<b>TABELLENVERZEICHNIS</b> .....	<b>VII</b>
<b>1 EINLEITUNG</b> .....	<b>1</b>
<b>2 ELEKTRONISCHE POSTSYSTEME</b> .....	<b>3</b>
2.1 NACHRICHTEN .....	3
2.2 ELEKTRONISCHE POST (E-MAIL) .....	3
2.3 BESTEHENDE MESSAGE HANDLING SYSTEME .....	4
2.3.1 X.400 Message Handling System.....	5
2.3.2 SMTP (Simple Mail Transfer Protokoll) .....	6
2.3.3 Eigenschaften der elektronischen Post.....	8
2.4 E-MAIL ERWEITERUNGEN.....	8
2.4.1 Multi-purpose Internet Mail Extension (MIME) .....	10
2.5 ACTIVE MAIL.....	12
2.5.1 Verschiedene Anwendungsbereiche von Active Mail .....	14
2.5.2 Active Mail zur Unterstützung von Gruppenarbeit in verteilten Systemen .....	15
2.5.2.1 Computational Mail .....	17
2.5.2.2 Active Mail von Goldberg.....	19
2.5.3 Active Mail für Multimedia .....	23
2.5.3.1 ActiveM <sup>3</sup> .....	23
2.5.3.2 Mobile Agenten mit ActiveM <sup>3</sup> .....	26
2.5.4 Sprache einer Active Mail .....	27
2.5.5 Anforderungen an ein Active Mail System .....	30
2.5.6 Vorteile und Nachteile von Active Mail.....	33
<b>3 AGENTENTECHNOLOGIE</b> .....	<b>35</b>
3.1 DEFINITIONEN .....	35
3.2 SOFTWARE AGENTEN .....	36
3.3 MOBILE AGENTEN .....	37
3.3.1 Vorteile und Nachteile von Mobilten Agenten.....	39
3.3.2 Mobile Agentensysteme .....	41
3.3.2.1 Telescript.....	41
3.3.2.2 Das Agentensystem Mole.....	42
3.4 VOR- UND NACHTEILE VON ACTIVE MAIL IN VERBINDUNG MIT MOBILTEN AGENTEN .....	44
<b>4 SPEZIFIKATION</b> .....	<b>48</b>
4.1 ZIELSETZUNG .....	48
4.2 ALLGEMEINE BESCHREIBUNG .....	48
4.3 INFORMELLE BESCHREIBUNG DER FUNKTIONALITÄT VON ACTIVEMAILMOLE.....	50
4.3.1 Aspekte der Benutzerschnittstelle .....	50
4.3.2 Aspekte des Systemagenten.....	52
4.3.3 Aspekte der „Active Mail Anwendungen“ .....	53

4.4 WEITERE ANFORDERUNGEN .....	54
<b>5 ENTWURF UND IMPLEMENTIERUNG.....</b>	<b>55</b>
5.1 DIE ACTIVE MAIL IN ACTIVEMAILMOLE.....	55
5.1.1 Die Klasse <i>AMGeneralForm</i> .....	57
5.1.1.1 Attribute .....	57
5.1.1.2 Methoden.....	59
5.1.2 Das dynamische Modell.....	62
5.2 DER SYSTEMAGENT VON ACTIVEMAILMOLE .....	71
5.3 DIE BENUTZERSCHNITTSTELLE .....	73
5.3.1 Gestaltung der Oberfläche .....	74
5.3.2 Die <i>Inbox</i> von <i>ActiveMailMole</i> .....	76
5.3.3 Die <i>Outbox</i> von <i>ActiveMailMole</i> .....	78
5.3.4 Setup .....	79
5.4 BEISPIELANWENDUNGEN.....	81
5.5 ZUSAMMENFASSENDE BEWERTUNG VON ACTIVEMAILMOLE .....	85
<b>6 ZUSAMMENFASSUNG .....</b>	<b>88</b>
6.1 ERGEBNISSE.....	88
6.2 AUSBLICK .....	89
<b>ANHANG .....</b>	<b>91</b>
A GLOSSAR.....	91
<b>LITERATURVERZEICHNIS .....</b>	<b>95</b>

## Abkürzungsverzeichnis

ActiveM <sup>3</sup>	Active Multimedia Mail
AM	Active Mail
AMM	ActiveMailMole
CM	Computational Mail
CSCW	Computer-Supported Cooperative Work
E-Mail	electronic Mail, elektronische Post
IMAP	Internet Mail Access Protocol
MA	Mobile Agenten
MHS	Message Handling System
MIME	Multi-purpose Internet Mail Extensions
MTA	Message Transfer Agent
MTS	Message Transfer System
OMT	Object Modeling Technique
PDA	Personal Digital Assistant
PEM	Privacy Enhanced Mail
PGP	Pretty Good Privacy
POP3	Post Office Protocol - Version 3
Safe-Tcl	Safe Tool Command Language
SMTP	Simple Mail Transfer Protocol
Tcl	Tool Command Language
UA	User Agent

## Abbildungsverzeichnis

Abbildung 1: E-Mail .....	4
Abbildung 2: Vereinfachtes Modell eines Message Handling Systems.....	6
Abbildung 3: SMTP Modell.....	7
Abbildung 4: Eine MIME E-Mail .....	11
Abbildung 5: Active Mail Architektur von Goldberg .....	20
Abbildung 6: Formular in ActiveM <sup>3</sup> .....	24
Abbildung 7: Die Struktur einer Active Mail in ActiveM <sup>3</sup> .....	25
Abbildung 8: Die Struktur eines Agentensystems .....	37
Abbildung 9: Architektur des Mole Systems .....	43
Abbildung 10: Informelle Beschreibung der Architektur von ActiveMailMole.....	50
Abbildung 11: ActiveMailMole Klassendiagramm .....	55
Abbildung 12: ActiveMailMole Modell .....	56
Abbildung 13: OMT Notation für Zustandsdiagramme.....	63
Abbildung 14: Zustandsdiagramm eines AMAgenten.....	64
Abbildung 15: Erzeugungszustand.....	65
Abbildung 16: Versandzustand.....	65
Abbildung 17: Ankunftszeitpunkt .....	66
Abbildung 18: Annahmezustand.....	66
Abbildung 19: Ausführungszustand.....	67
Abbildung 20: AMSystemAgent-AMAgent Kommunikation .....	72
Abbildung 21: Startfenster von ActiveMailMole.....	74
Abbildung 22: Hauptfenster in ActiveMailMole .....	75
Abbildung 23: Die Inbox von ActiveMailMole .....	77
Abbildung 24: Die Outbox von ActiveMailMole .....	79
Abbildung 25: Setup-Fenster von ActiveMailMole.....	80
Abbildung 26: Initialisierung des AMMeetingScheduler .....	82
Abbildung 27: Aktivierung des AMMeetingScheduler .....	83
Abbildung 28: Ergebnis des AMMeetingSchedulers.....	84

## **Tabellenverzeichnis**

Tabelle 1: Gegenüberstellung der einzelnen Active Mail Systeme.....	34
Tabelle 2: Gegenüberstellung der Systeme „Mobilen Agenten“ und „Active Mail“ .....	47
Tabelle 3: Gegenüberstellung der Systeme „ActiveM <sup>3</sup> “ und „ActiveMailMole“ .....	86

## 1 Einleitung

Der Traum von „Active Mail“ existiert schon seit langem. Vittal war einer der ersten der diesen Traum vor Augen hatte: elektronische Briefe, die nicht nur passive Daten übermitteln, sondern auch eigenständige Handlungen ausführen können.<sup>1</sup> Briefe also, die mit dem Empfänger „reden“, ihm Fragen stellen und auf seine Antworten reagieren können. Seitdem wurde dieser Wunsch, nämlich elektronische Post um eigenständige Programmausführung zu erweitern, bereits in vielen Systemen integriert. Active Mails können zu bestimmten Zeitpunkten ausgeführt werden und bieten außerdem die Möglichkeit, als Plattform für andere Anwendungen (z.B. im Bereich der Gruppenarbeit) eingesetzt zu werden.

Mobile Agenten stellen ein erfolgversprechendes Programmierparadigma in Verteilten Systemen dar. In einem Mobile-Agenten-System werden Informationen nicht nur in Form von statischen Daten, sondern es werden auch „Programmkonstrukte“ übertragen, die die Fähigkeit haben, autonom zu handeln und von Rechner zu Rechner zu migrieren. Das Agentensystem bietet dabei dem Agenten und dem darunterliegenden System Schutz und einige Basisdienste an. Einer dieser Basisdienste ermöglicht es dem Mobilten Agenten, den Rechner zu wechseln und dort aktiv zu werden.

Aufgrund der Nähe von Active Mails zu Mobilten Agenten wird im Rahmen dieser Diplomarbeit ein Active-Mail-System entworfen und implementiert, mit dem Active Mails auf der Grundlage des Mobilten-Agenten-Systems Mole verschickt und verarbeitet werden können.

Um diese Active Mails lesen zu können, muß die Struktur der Active Mail entwickelt werden, sowie die Frage des Transports und die der benötigten Werkzeuge geklärt werden.

Die Diplomarbeit gliedert sich in sechs Teile. Nach dem Einführungsteil werden im zweiten Teil bestehende elektronische Postsysteme sowie verschiedene Ansätze bestehender Active-Mail-Systeme vorgestellt und ein allgemeiner Anforderungskatalog für Active-Mail-Systeme aufgestellt.

Im dritten Teil wird die Agententechnologie beschrieben, sowie die Vor- und Nachteile von Active Mail in Verbindung mit Mobilten Agenten gegenübergestellt.

---

<sup>1</sup> vgl. Vittal (1981)

Im vierten Abschnitt wird zunächst das zu entwickelnde System spezifiziert und im abschließenden fünften Abschnitt der Entwurf und die Implementierung des Systems dargestellt.

Die Zusammenfassung am Ende der Arbeit zeigt die wesentlichen Ergebnisse auf und gibt einen Ausblick auf die Erweiterungsmöglichkeiten des Systems.

## 2 Elektronische Postsysteme

Da eine nahe Verwandtschaft von Active Mail zu bestehenden elektronischen Postsystemen existiert, sollte zunächst eine Abgrenzung der wichtigsten verwendeten Begriffe erfolgen.

Der Austausch von Information spielt eine wichtige Rolle in unserem Leben. Uns ist es möglich, Informationen über viele verschiedene Dienste, wie z.B. Telefon, Telex, Fax und Computer auszutauschen. Nicht nur die Anzahl der verschiedenen Informationsaustauschdienste hat in den letzten Jahren zugenommen, auch die Benutzung dieser Dienste nimmt rapide zu. Speziell die Nutzung des Informationsaustausches über den Computer wird zunehmend wichtiger, da immer mehr Computer mit Netzwerken oder mit dem Internet verbunden sind. Ein wesentlicher Bestandteil der Kommunikation zwischen Anwendern geschieht mittels Austausch elektronischer Nachrichten (engl.: messages).

### 2.1 Nachrichten

Was sind Nachrichten (messages)?

Als Nachricht kann man das „Gefäß“ der Information unter Einschluß der Information selbst bezeichnen. Die übermittelte Nachricht ist dann von Bedeutung, wenn eine Abbildung bekannt ist, die eine Nachricht mittels einer Interpretationsvorschrift auf eine Information abbildet.<sup>2</sup> Bei elektronischen Nachrichten handelt es sich um eine endliche Zeichenfolge, die eine Information vermittelt. Bei der Übertragung von Nachrichten (mittels Netzwerk, oder Datenbus im Rechner) müssen oft feste Regeln, sog. Protokolle eingehalten werden. Zum Beispiel bestehen Nachrichten meist aus fest formatiertem Nachrichtenkopf, der eigentlichen Nachricht und einem Nachrichtenende.<sup>3</sup>

### 2.2 Elektronische Post (E-Mail)

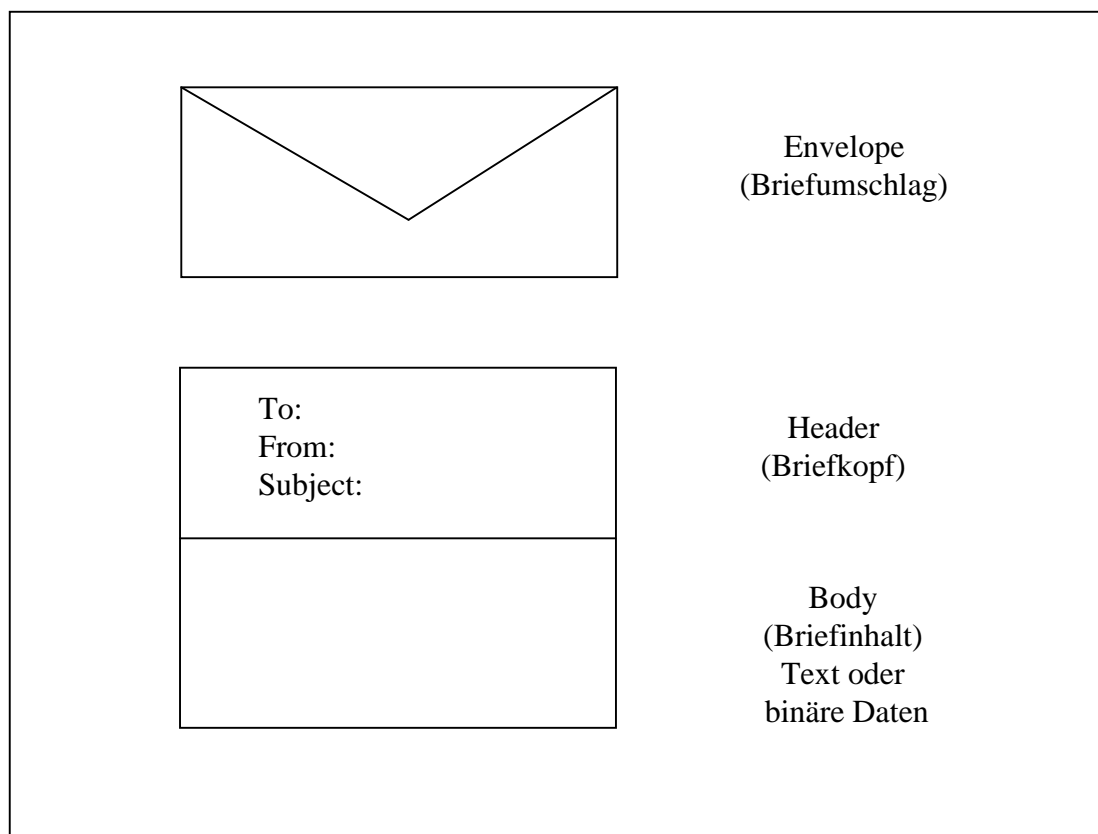
Die elektronische Post (engl.: electronic mail, oder E-Mail) ist der Austausch einer Nachricht zwischen Personen über ein Rechnernetz. Sie funktioniert ähnlich der normalen Briefpost. Wir schreiben einen Brief, stecken ihn in einen Briefumschlag, schreiben den Empfänger und den Absender auf den Briefumschlag und geben den Brief der Post, die für uns den Brief dem Empfängers überbringt.

---

<sup>2</sup> vgl. Ludewig (1991)

<sup>3</sup> vgl. Duden (1989)

Die elektronische Post hat ebenso einen Briefumschlag (envelope) mit Absender, Empfänger und weiteren Informationen, die wichtig sind für den Briefübermittler, wie Zeitpunkt der Aufgabe u.v.a.. Der Briefkopf (header) enthält wichtige Daten für das empfangende Programm, welches den Brief entgegennimmt, und für den Benutzer, an den der Brief gerichtet ist (siehe auch Abbildung 1). Dies sind Informationen wie Absender, Empfänger, Betreff und Art der Daten, die übermittelt werden. Die eigentliche Nachricht, die für den Empfänger bestimmt ist, steht im Briefinhalt (Body). Dieser kann nur einen Text oder bei entsprechender Erweiterung des E-Mail Programms auch andere Daten (siehe auch Kapitel 2.4 E-Mail Erweiterungen) enthalten.



**Abbildung 1: E-Mail**

### **2.3 Bestehende Message Handling Systeme**

Innerhalb eines bestehenden Netzwerks können viele verschiedene Protokolle verwendet werden, um Nachrichten bzw. E-Mails auszutauschen. Diese Protokolle müssen sich nach bestimmten Standards richten, die entweder von unabhängigen Organisationen definiert sind, oder sie werden von Herstellern, meist beschränkt auf bestimmte Rechner, implementiert.

Heute sind die wichtigsten Standardprotokolle X.400 und SMTP (Simple Mail Transfer Protokoll). Es gibt noch eine ganze Reihe anderer Protokolle. Um die Grundfunktionalität von bestehenden Electronic Mail Systemen zu erklären, wird im folgenden nur auf diese beiden System eingegangen.

### **2.3.1 X.400 Message Handling System**

Elektronische Post wird gesendet und empfangen von Elektronischen Postsystemen, gleichbedeutend mit Message Handling Systemen (MHS).

Der Standard X.400 für MHS wurde von der International Telecommunication Union (ITU-T) definiert und in der ITU-T Recommendation X.400 beschrieben.<sup>4</sup>

Die allgemeine Architektur von MHS wird in X.402 beschrieben. Nachfolgend ein kleiner Überblick dieser Architektur.<sup>5</sup>

Ein MHS ist ein Programm, welches das Erzeugen einer Nachricht unterstützt und diese an ein vorherbestimmtes Ziel, nachdem die Nachricht fertiggestellt ist, schickt. Ein MHS übernimmt den Transport von Nachrichten und speichert eingehende Nachrichten ab, um sie je nach Wunsch des Benutzers anzuzeigen.

Ein MHS besteht grundsätzlich aus zwei Teilen, dem Message Transfer System (MTS) und dem User Agent (UA), der Benutzerschnittstelle.

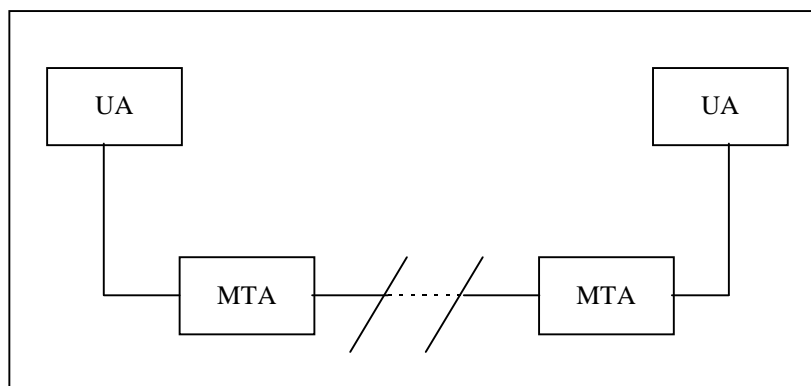
Das MTS schickt Nachrichten an den Bestimmungsort und gibt Meldungen aus, wenn neue Nachrichten ankommen. Es besteht aus Message Transfer Agents (MTA), die den tatsächlichen Transport, das Routing und die Übergabe der Nachricht via einem Rechnernetz, bewerkstelligt.

Die Benutzerschnittstelle (UA) stellt dem Benutzer Werkzeuge zur Verfügung, um Nachrichten zu erzeugen, zu empfangen, anzuzeigen und zu archivieren. Das Benutzerinterface kann im MTS realisiert sein (siehe auch Abbildung 2).

---

<sup>4</sup> vgl. ITU-T X.400 (1996)

<sup>5</sup> vgl. ITU-T X.402 (1995)



**Abbildung 2 : Vereinfachtes Modell eines Message Handling Systems**

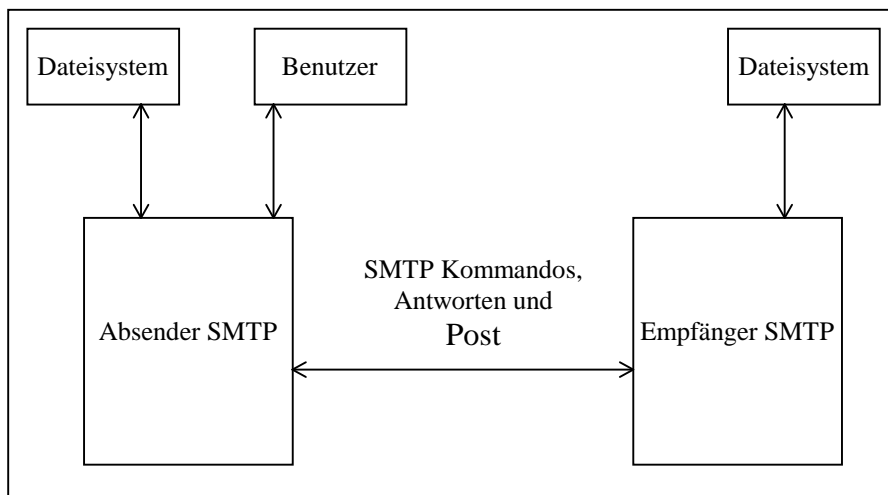
### 2.3.2 SMTP (Simple Mail Transfer Protokoll)

SMTP wird benutzt, um Post (engl.: mail) über das Internet auszutauschen und ist in akademischen Kreisen sehr verbreitet. Beschrieben wird es im Internet Dokument „request for comment“ RFC821.<sup>6</sup> Praktisch alle Rechner, die TCP/IP Dienste (somit alle UNIX Systeme) benutzen können sich dieses Protokolls bedienen. Es ist ein sehr einfaches Protokoll, reicht aber für viele Anwendungsfälle der elektronischen Post aus. Um auch E-Mail an Benutzer in Subnetzen, d.h. Netze die nicht direkt an das Internet angeschlossen sind, zu schicken, gibt es Übergänge (engl. Gateways) zwischen den Netzen und Protokollen, die den Datenaustausch regeln. Dies soll aber hier nicht weiter betrachtet werden.

Das schon recht alte SMTP beschränkt sich bei der Übertragung von Zeichen auf den amerikanischen ASCII-Zeichensatz (7-bit). Andere Zeichen werden durch ASCII-Zeichen ersetzt oder schlicht weggelassen. Somit können E-Mails keine Sonderzeichen enthalten, ebenso müssen binäre Daten als ASCII-Daten verschlüsselt werden, um sie als E-Mail verschicken zu können. Um diese Nachteile zu beseitigen und die E-Mail-Kommunikation für ein multimediales Zeitalter tauglich zu machen, wurde das MIME - Protokoll entwickelt (siehe auch Kapitel 2.4.1).

---

<sup>6</sup> vgl. Postel (1982)



**Abbildung 3: SMTP Modell<sup>7</sup>**

SMTP<sup>8</sup> übernimmt auf Anfrage des Benutzers den Transport von Mails. SMTP schickt diese direkt vom Absender SMTP zum Empfänger SMTP, wenn beide den gleichen Transportdienst (z.B. TCP/IP) benutzen (siehe auch Abbildung 3), oder wenn dies nicht der Fall ist, über ein oder mehrere Gateways.

Einige Knotenpunkte im Internet sind jedoch nicht geeignet um einen SMTP-Server oder ein Message Transfer System (MTS) zu unterhalten, z.B. Workstations mit unzureichender Festplattenkapazität, oder PCs, die nicht ständig im Internet erreichbar sind. Um diese Probleme zu lösen, kann das MTS diesen Knotenpunkten ein Post-Abholdienst anbieten. Das Post Office Protocol - Version 3 (POP3) bietet hier dem Benutzer die Möglichkeit, seine Post vom MTS-Server abzuholen, ggf. zu löschen und diese dann lokal zu verwalten. Es ist allerdings nicht für umfangreiche Operationen auf dem MTS-Server gedacht.<sup>9</sup>

Das Internet Message Access Protocol (IMAP4) hingegen ist ein weitaus komplexeres und umfangreicheres Protokoll. Es erlaubt die Verwaltung der Post auf dem Server. Der Benutzer kann dort eigene Verzeichnisse einrichten (suchen, löschen, umbenennen u.v.a.) und diese nach dem Client/Server-Prinzip verwalten. Es bietet somit auch reinen Netzwerkrechnern, die keine eigene Festplatte besitzen, das Verwalten der eigenen Post.<sup>10</sup>

<sup>7</sup> vgl. Postel (1982)

<sup>8</sup> Hiermit ist nicht das Protokoll gemeint, sondern Programme, die dieses Protokoll umsetzen (z.B. sendmail, ESMTP, ...). Postel (1982) beschreibt SMTP als Host-Host Verbindung.

<sup>9</sup> vgl. Myers (1996)

<sup>10</sup> vgl. Crispin (1996)

### 2.3.3 Eigenschaften der elektronischen Post

#### Asynchrone Kommunikation

Eine der Haupteigenschaften der elektronischen Post ist die asynchrone Übermittlung, ähnlich der normalen Briefpost. Wenn wir eine Nachricht an eine Person schicken, so kann der Empfänger die Nachricht lesen, wann immer er will. Dies gilt ebenso für die Antwort auf eine spezielle Nachricht. Asynchrone Übermittlung ist der größte Vorteil von elektronischer Post: der Empfänger muß weder die Nachricht sofort lesen, noch muß er zur gleichen Zeit im Netzwerk präsent sein. Der Briefträger, in unserem Falle der MTA, legt die Post in den Briefkasten (Mailbox) und sein Auftrag ist erledigt. Dies erlaubt somit eine Kommunikation über die ganze Welt mit Personen, unabhängig von den unterschiedlichen Zeitzonen.

#### 1-1, 1-n und n-m Kommunikation

Eine weitere wichtige Eigenschaft der E-Mail ist neben der 1-1 Kommunikation, d.h. eine Person schickt einer anderen Person eine Nachricht, die 1-n und die n-m Kommunikation: man kann also mehreren Personen gleichzeitig eine Nachricht zukommen lassen, ebenso ist es möglich, über Verteilerlisten zu kommunizieren, wo mehrere Personen mit vielen Personen in Kontakt stehen. Ein bekanntes Beispiel sind hier die News-Groups.

Aufgrund der vorgestellten Postsysteme und der eben genannten Eigenschaften der elektronischen Post kann man ein **elektronisches Postsystem** folgendermaßen definieren<sup>11</sup>:

*Ein System, welches ein Informationsaustausch anbietet, im engeren Sinne den Austausch von elektronischen Nachrichten. Diese Nachrichten werden vom Absender (eine Person oder ein Programm) zu einem oder mehreren Empfängern (einer Person oder einem Programm) übermittelt, die im voraus bestimmt wurden. Nachrichten, die mit einem solchen System übermittelt werden, können aus unterschiedlichen Datentypen bestehen (Text, Grafik, Video oder Tondaten).*

### 2.4 E-Mail Erweiterungen

Die normale E-Mail beschränkt sich zunächst auf die Übertragung von reinem Text; dies ist jedoch nicht alles. Mit der elektronischen Post kann man auch andere Dienste im

---

<sup>11</sup> vgl. Geesink (1992)

Internet nutzen. Dem Benutzer steht es offen, weitere Dienste über das Transportmittel E-Mail anzufordern. Er kann z.B.:

- Datenbankabfragen von einem Server generieren lassen.
- Dateien eines FTP-Servers übermitteln lassen.
- World-Wide-Web Seiten über einen Server zuschicken lassen.
- mittels Fax-Übergängen, Nachrichten als Fax verschicken lassen.
- u.v.a.

Bilder, Tondaten und Videodaten können ebenso verschickt werden, jedoch treten hier Probleme auf, da sich das schon recht alte SMTP-Protokoll sich auf die Übertragung von Zeichen des amerikanischen ASCII Zeichensatz (bestehend aus 128 Zeichen) beschränkt; andere Zeichen werden ersetzt oder einfach weggelassen. Somit entstehen zwei Probleme bei der Verschickung von E-Mails:

- Es können keine Sonderzeichen, wie z.B. die deutschen Umlaute übertragen werden
- binäre Daten, z.B. Bilder oder Programme, müssen als ASCII-Daten verschlüsselt werden

Nachrichten, die in diesen Problembereich fallen, müssen deshalb verschlüsselt werden, d.h. die Nachricht muß in US-ASCII-Zeichen umgewandelt werden. Dazu gibt es verschiedene Methoden: UUEncode/UUDecode und BinHex.

UUEncode/UUDecode ist die älteste und weitverbreitetste Methode, um binäre Daten für den Versand per E-Mail aufzubereiten. Die Daten werden vom Absender mit UUEncode verschlüsselt, abgeschickt und der Empfänger entschlüsselt diese wieder mit UUDecode. BinHex ist eine Methode um Daten hauptsächlich einem Apple/Macintosh Benutzer zugänglich zu machen<sup>12</sup>. MIME ist etwas leistungsfähiger als die eben genannten Methoden bzw. benutzt diese Methoden, um Daten zu verschlüsseln und wird im nächsten Kapitel etwas ausführlicher beschrieben.

---

<sup>12</sup> BinHex gibt es auch für die Unix-Welt, wird jedoch nicht so häufig wie UUEncode/UUDecode eingesetzt.

### 2.4.1 Multi-purpose Internet Mail Extension (MIME)<sup>13</sup>

Anstatt das weitverbreitete SMTP-Protokoll zu ersetzen, um die im vorherigen Kapitel beschriebenen Probleme zu lösen, wurde das Multi-purpose Internet Mail Extension (MIME) entwickelt. Es erweitert das Format der Nachrichten, wie sie in den Dokumenten RFC 821/822 definiert sind, um E-Mails verschicken zu können,

1. die einen anderen Zeichensatz als den US-ASCII Zeichensatz im header-Teil oder im body-Teil benutzen
2. mit einer erweiterbaren Menge von verschiedenen nicht aus Zeichen bestehenden Formaten, d.h. binären Daten, im Body-Teil
3. mit mehrteiligen Nachrichtsinhalten

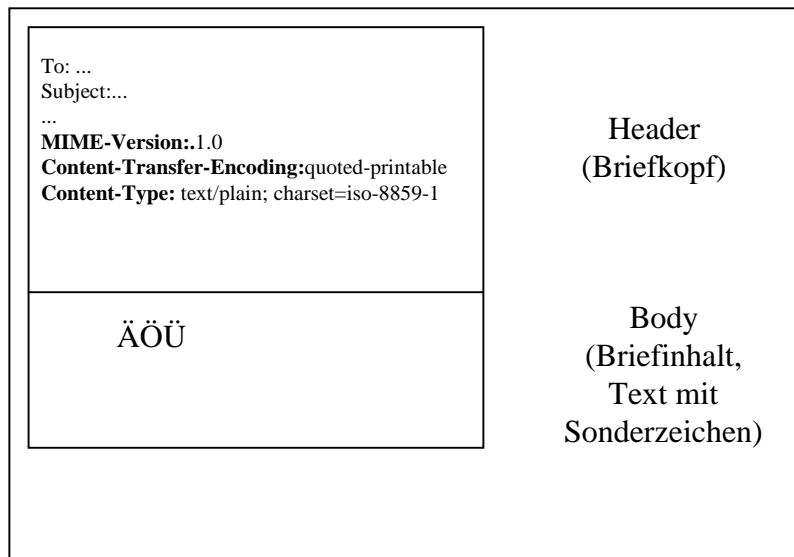
Damit der empfangende User Agent (E-Mail-Programm) weiß, welche Daten in einer Nachricht enthalten sind, definiert MIME im elektronischen Briefumschlag oder im Briefinhalt (bei mehrteiligen Nachrichten) verschiedene Bezeichnungen (Labels):

- eine MIME-Version Bezeichnung, um verschiedene Versionen von MIME behandeln zu können
- eine Content-Type Bezeichnung, die verschiedene Datentypen und Subdatentypen des Briefinhalts spezifiziert
- eine Content-Transfer-Encoding Bezeichnung, um die Verschlüsselungsart der Originaldaten festzulegen; dies ist wichtig beim Transport und der Entschlüsselung beim Empfänger
- Content-ID und Content-Description, um den Inhalt der Nachricht zu beschreiben

Der User Agent benutzt die quoted-printable encoding Methode, um Sonderzeichen durch einen Code zu ersetzen, der aus drei US-ASCII Zeichen besteht. Bei einem MIME-fähigen Empfänger UA werden diese dann wieder automatisch in die richtigen Zeichen umgewandelt (siehe auch Abbildung 4). Eine weitere oft benutzte Ver- und Entschlüsselungs Methode ist die base64-Methode. Sie benutzt 64 Zeichen aus dem US-ASCII Zeichensatz und wandelt jeweils 24-bit Gruppen (beispielsweise drei 8-bit Zeichen) in vier Zeichen des 64 Zeichenvorrates um. Dies ermöglicht eine problemlose Übermittlung jeglicher Daten in heterogenen Netzen.

---

<sup>13</sup> vgl. Borenstein (1996)



**Abbildung 4: Eine MIME E-Mail**

Der Aufbau des Content-Typ Feldes ist:

Typ / Subtyp ; parameter

Mit dem Content-Type Feld lassen sich somit genügend Eigenschaften des Briefinhalts beschreiben, damit der empfangende User Agent einen geeigneten Mechanismus auswählen kann, um die Nachricht anzuzeigen und um unbekannte Typen entsprechend zu behandeln. Über sogenannte „mailcap-based User Agents“ kann angegeben werden, was bei entsprechenden Content-Types gemacht werden soll. Dies kann auch bedeuten, daß andere Programme aufgerufen werden, um den Inhalt der Nachricht dem Benutzer entsprechend zu übermitteln.

Die vorgestellten elektronischen Postsysteme stellen ein adäquates Mittel dar, elektronische Nachrichten zu transportieren. Sie sind standardisiert und einfach zu benutzen. Fast jeder Anwender im Internet kann diese Postsysteme nutzen, um Daten zu übermitteln. Mit einem Standard wie MIME ist eine flexible Erweiterungsmöglichkeit von elektronischen Briefen gesichert.

Es fehlen jedoch Sicherheitsmechanismen, wie z.B. eine Authentifizierung oder eine Verschlüsselung der zu übertragenden Daten, die automatisch im System integriert sind. Diese müssen meist vom Benutzer selbst oder von User Agenten übernommen werden. Desweiteren fehlen Mechanismen, die Nachrichten aktiv werden lassen.

Mit der Entwicklung von Active Mail Systemen wird versucht, diese beschriebenen Nachteile der bestehenden elektronischen Postsysteme zu beheben.

## 2.5 Active Mail

In den vorherigen Kapiteln wurde die Funktionsweise der elektronischen Post vorgestellt. Hier wurde deutlich, daß eine elektronische Nachricht eine Einheit ist, die vom Empfänger gelesen oder im Falle von Multimedia Nachrichten (Tondaten, Video, Bilder) angezeigt werden kann. Dies bedeutet aber, daß Nachrichten nur passiv sind und nicht eigenständige Handlungen ausführen können. Vittal<sup>14</sup> war einer der ersten, der die Bedeutung erkannt hat, Nachrichten auch mit aktiven Eigenschaften auszustatten. Nachrichten waren für ihn eine Einheit, die fähig sind, eigene Handlungen auszuführen, mit dem Benutzer in Interaktion zu treten und infolge von Antworten des Empfängers aktiv zu werden.

Nachrichten mit diesen aktiven Eigenschaften werden aktive Nachrichten (engl.: active messages)<sup>15</sup> genannt. Sie bieten nicht nur Information an, sondern können ebenso Daten vom Empfänger sammeln, aufbereiten und als Ergebnis ausgeben. Sie können also eigene Handlungen ausführen, die nicht unbedingt dem Benutzer sichtbar sind, mit dem Benutzer in Interaktion treten (z.B. Fragen stellen) und aufgrund dieser entsprechend reagieren. Als active message kann man sich einige Programmzeilen vorstellen, die ausgeführt werden, wenn es der Empfänger will. Diese Programmzeilen halten die Kontrolle über die active message und entscheiden (teils mit den Antworten des Benutzers), welcher Teil des Programmes ausgeführt werden soll.

Ein einfaches Beispiel könnte ein Fragebogen sein. Wenn der Empfänger die empfangene Nachricht ausführt, d.h. das Programm der active message, so wird er gebeten, eine Reihe von Fragen zu beantworten. Aufgrund der Antworten, kann das Programm entscheiden, Fragen zu überspringen oder weiterführende Fragen zu stellen. Nachdem der Benutzer den Fragebogen „ausgefüllt“ hat, kann die Nachricht die Antworten dem Absender übermitteln und sich selbst zu anderen Empfängern weiterleiten (falls diese vom Absender vorher spezifiziert wurden).

Die Abarbeitung der Daten, die vom Empfänger bereitgestellt werden, hängt von der Aufgabe der active message ab. Im Falle des Fragebogens könnte die Aufgabe der active message sein, nur die Antworten des Empfängers zu sammeln und diese an den Absender wieder zurückzusenden. Es ist aber ebenso möglich, daß die active message, schon bevor sie den Absender erreicht, die gesammelten Daten aufbereitet und nachdem sie alle Empfänger erreicht hat - z.B. eine statische Analyse der Daten dem Auftraggeber als Ergebnis anzeigt. In diesem Falle kann man sich die active message als reisendes Ex-

---

<sup>14</sup> vgl. Vittal (1981)

<sup>15</sup> Im folgenden wird nur noch der gebräuchliche Begriff „active message“ verwendet.

pertensystem<sup>16</sup> vorstellen, das Daten aufgrund vordefinierter Regeln des Absenders abarbeitet. Eine der Regeln könnte sein, die Weiterleitung der Nachricht dem Empfänger zu überlassen, d.h. der Empfänger der active message muß den nachfolgenden Empfänger angeben und nicht der Initiator der Nachricht. Somit weiß der Initiator nicht unbedingt, wer außer den vorher bestimmten Empfängern diese active message erhält. Dies nennt man „dynamic message routing“ (dynamische Nachrichten Leitweglenkung) im Gegensatz zu „static message routing“ (Nachrichten werden nur an Empfänger geschickt, die vorher spezifiziert wurden), das von konventionellen elektronischen Postsystemen benutzt wird.<sup>17</sup>

Ein Beispiel einer dynamischen active message kann die Bekanntgabe eines Buches sein. Zuerst kennt man nicht alle Interessenten, die diese Bekanntgabe erhalten wollen. Man könnte sie einer großen Anzahl von Personen zukommen lassen. Jedoch ist der Erfolg meist nur beschränkt, da nicht alle Personen diese Nachricht erhalten wollen und viele nicht erreicht werden, die diese Nachricht interessiert. Um dies zu vermeiden, könnte man eine active message an eine kleinere Anzahl von Personen schicken, von denen man weiß, daß sie interessiert sind und diese active message mit einer Frage ausstatten, die nach weiteren interessierten Personen fragt. Wenn der Empfänger diese interessierten Personen kennt, kann er deren Adressen der active message mitteilen und diese schickt dann Kopien automatisch an diese Personen. Auf diese Weise kann man die Zahl der nicht interessierten Personen reduzieren und den Kreis der Interessierten erweitern.

Aufgrund der eben genannten Eigenschaften kann man ein **Active Mail System** folgendermaßen definieren:

*Ein Active Mail System ist ein System, das den Informationsaustausch von active messages unterstützt. Diese Nachrichten werden vom Absender (eine Person oder ein Programm) zu einem oder mehreren Empfängern (eine Person oder ein Programm) übermittelt, wobei mindestens ein Empfänger anfangs benannt werden muß. Die Nachrichten, die von einem solchen System versendet werden, bestehen aus Daten oder aus Programmzeilen, die ausgeführt werden können, falls der Empfänger es wünscht.<sup>18</sup>*

---

<sup>16</sup> vgl. Turban (1990)

<sup>17</sup> vgl. Geesink (1992)

<sup>18</sup> vgl. Geesink (1992)

Eine **active message** soll hierbei wie folgt definiert sein:<sup>19</sup>

*Eine active message ist eine Nachricht, die teilweise aus Programmzeilen besteht, die eigenständige Handlungen ausführen (beispielsweise eine Kopie von sich selbst an einen Empfänger schicken, der nicht vorher benannt war) und die auf diese Weise Informationen sammelt, auswählt oder verbreitet, die in Interaktion mit einem Empfänger entsteht.*

In diesem Sinn kann man die **Active Mail** als Erweiterung der normalen E-Mail mit active messages bezeichnen.<sup>20</sup>

### 2.5.1 Verschiedene Anwendungsbereiche von Active Mail

Active Mail erweitert die Funktionalität der herkömmlichen E-Mail um zusätzliche Flexibilität in der Kommunikation mit Benutzern oder Anwendungen. Active Mail wird in den meisten Ansätzen dazu benutzt, die Vorteile (hauptsächlich der asynchrone Charakter und die einfache Bedienbarkeit) der normalen E-Mail zu nutzen und diese soweit zu erweitern, daß der Anwendungsbereich von Active Mail sehr vielseitig wird. Anwendungsbereiche könnten sein:

- **Unterstützung von Gruppenarbeit**

Die Vorteile der Active Mail könnten dazu verwendet werden, z.B. *Fragebögen*, die interaktiv mit dem Benutzer beantwortet werden müssen, zu verschicken und zu bewerten oder *dynamische Nachrichtenverteiler* zu ermöglichen, bei denen nicht alle Empfänger anfangs bekannt sind (*Terminvereinbarung* mit mehreren Benutzern, *verteilte Dokumentenverwaltung/-erstellung*, usw.).

- **Darstellung von Multimedia Anwendungen**

Hier könnte die Active Mail dazu verwendet werden, die *Synchronisierung* von Bild und Ton zu übernehmen, oder eine *Videokonferenz* zu initiieren, usw.

- **Mobile Kommunikation**

Mobile Kommunikation ist charakterisiert durch eine geringe Bandbreite des Kommunikationsnetzes, kleine Anzeigeräte und nur zeitweilige Verfügbarkeit des Netzwerks. Active Mail unterstützt diesen Anwendungsbereich zum einen dadurch, daß eine neu erzeugte Active Mail immer nur dann abgeschickt werden muß, wenn der Benutzer eine Verbindung zu einem stationären System aufbaut. Danach können

---

<sup>19</sup> vgl. Geesink (1992)

<sup>20</sup> vgl. Goldberg (1992)

Active Mails zwischen verschiedenen heterogenen Systemen und unterschiedlichen Netzwerken Informationen sammeln, auswerten und aufbereiten sowie Transaktionen ausführen, ohne daß der Benutzer bzw. Initiator noch „online“ sein muß. Auf der anderen Seite muß der Empfänger nicht ständig präsent sein, sondern braucht erst bei einer Verbindung zum stationären System neu eingegangene Active Mails auszuführen. Sobald ein erfolgreicher Abschluß vorliegt, könnte das Ergebnis je nach Aufgabe der Active Mail, bei der nächsten Verbindung des Benutzers zum stationären System, angezeigt werden.

- **in der Benutzerunterstützung**

Active Mail könnte hier die Möglichkeit bieten, neuen Benutzern eine *kontextsensitive Hilfe* (z.B. zum Inhalt der Active Mail) anzubieten, um ihnen den Einstieg zu erleichtern, oder die Active Mail bietet, je nach Einstellung des Benutzers, eine *Oberfläche* für Experten oder Einsteiger an.

- **automatisierte Datenbankabfragen**

Hier könnte z.B. ein Benutzer eine Active Mail benützen, um eine *Datenbankabfrage*<sup>21</sup> asynchron zu starten. Der Datenbankserver, der diese Active Mail bekommt, könnte je nach Parametrisierung der Active Mail, diese sofort lokal starten oder erst dann, wenn er dafür genügend Ressourcen frei hat.

Es sind sicherlich noch weitere Anwendungsgebiete der Active Mail denkbar.

## 2.5.2 Active Mail zur Unterstützung von Gruppenarbeit in verteilten Systemen

Wenn zwei oder mehrere Personen eine Aufgabe oder eine Problemstellung kooperativ lösen wollen und dabei ein rechnergestütztes System verwenden, spricht man von CSCW (Computer-Supported Cooperative Work) oder Rechnerunterstützte Gruppenarbeit. Dabei teilen sich die einfachsten kooperativen Systeme einen Rechner und werden von kooperierenden Personen benutzt. Hier entstehen kaum Probleme bezüglich der Kommunikation und Koordination. Aber gerade durch die weltweit zunehmende Vernetzung und Aufhebung der Grenzen in der Kommunikation bietet sich die Möglichkeit geographisch verteilte Menschen in ihrer kooperativen Arbeit mit Hilfe von verteilten Systemen zu unterstützen. Dabei entstehen Probleme beim Übergang von kooperativen

---

<sup>21</sup> Der Spielraum ist hier sehr groß, es könnten ebenso Datenbankänderungen sein. Nur sind hier im besonderen Maße die Sicherheitsaspekte zu berücksichtigen, um vor unautorisiertem Datendiebstahl, Datenveränderung oder Datenzerstörung zu schützen.

Systemen zu verteilten kooperativen Systemen. Borenstein<sup>22</sup> erwähnt drei Kernprobleme die bei der Entwicklung von verteilten kooperativen Systemen zu berücksichtigen sind:

### **Das Problem der entfernten Installation**

Damit eine Anwendung von mehreren Personen benutzt wird, muß sie in der jeweiligen Rechnerumgebung installiert werden. Hierbei entsteht kein Problem, wenn jeder Anwender mit der Anwendung vertraut ist und diese selbst installiert oder jemanden damit beauftragen kann. Bei einer verteilten Anwendung kann man aber nicht davon ausgehen, daß alle Anwender auf dem selben Wissensstand sind und sicherstellen, daß die Anwendung auf ihrem System installiert wird. Je weiter diese Anwender verteilt sind, um so schwieriger wird es, ihre Rechnerumgebung und ihr Wissen homogen zu halten, bzw. auf dieser Homogenität aufzubauen, um eine kooperative Arbeit zu ermöglichen. Somit könnten CSCW-Anwendungen schon daran scheitern, daß sie nie richtig installiert wurden, oder nicht konsistent gehalten werden.

### **Das Problem der Integration neuer Teilnehmer**

Neben dem Problem, Software auf weitverteilten Arbeitsplätzen zu installieren, entsteht das Problem, Personen dazu zu bewegen, diese installierte Software oder Anwendung auch zu nutzen. Falls die Anwendung z.B. ein Experten-Frage-Modell ist, spielt auch die Motivation der Mitarbeiter eine entscheidende Rolle, insbesondere inwieweit die bisherigen Experten dazu bereit sind, Fragen von neuen Teilnehmern zu beantworten.

### **Das Problem von heterogenen Plattformen**

Neben einer Vielzahl von verschiedenen Betriebssystemen wie UNIX, VMS, MS-DOS, usw. gibt es noch eine größere Anzahl von Benutzeroberflächen die textorientierte oder grafische Oberflächen anbieten. Nun muß eine verteilte kooperative Anwendung allen Benutzern dieser verschiedenen Plattformen eine Benutzerschnittstelle anbieten, die sowohl auf textorientierten als auch auf grafischen Oberflächen funktionsfähig ist, um ein kooperatives Arbeiten zu ermöglichen.

All diese Probleme müssen bei der Entwicklung jeder verteilten Anwendung neu gelöst werden oder man sucht einen Mechanismus, der all diese Probleme für alle verteilten kooperativen Anwendungen einmalig löst. Benötigt wird also eine Plattform, die diese Probleme löst und auf welcher man neue Anwendungen entwickeln kann.

Active Mail könnte hier ein guter Ansatz sein, diese Problemstellungen zu lösen.

---

<sup>22</sup> vgl. Borenstein (1992)

### 2.5.2.1 Computational Mail<sup>23</sup>

Die Idee von Computational Mail ist sehr einfach: Anstatt nur Text oder Multimedia Nachrichten mittels der elektronischen Post zu verschicken, wird ein Programm in einer wohldefinierten Sprache verschickt. Computational Mail kann als Active Mail gesehen werden, welches bestehende elektronische Postsysteme als Vehikel kooperativer Systeme nutzt.

#### **Ansatz**

Der Ansatz von Computational Mail geht davon aus, daß eine ganze Reihe von Anwendungen mit einer Benutzerschnittstelle auskommen, die nur zeichenorientierte Eingabe und Ausgabe ermöglicht. Dies wird dadurch erreicht, daß nur fünf Funktionen zur Benutzerinteraktion erlaubt werden:

1. Eingabe eines Textes
2. Eingabe einer Zahl
3. Auswahl aus einer Liste von Fragen
4. Ausfüllen eines Formulars
5. Ausgabe eines Textes

Dadurch kann eine gewisse Portabilität gewährleistet werden, denn diese Operationen sind derart allgemein, daß sie auf allen text- oder grafikorientierten Systemen darstellbar sind. Jedoch ist zu bezweifeln, ob diese Operationen ausreichen, alle nötigen Anwendungen zu realisieren. Um eine gewisse Sicherheit der auszuführenden Mail gewährleisten zu können, besteht die Mail aus einem Programm in einer allgemeinen Programmiersprache. Die Ein- und Ausgabeoperationen wurden so beschränkt, daß sie nur in kontrollierter Art und Weise auf die Betriebsmittel zugreifen dürfen, z.B. dürfen sie keine existierenden Dateien überschreiben oder löschen. Durch die Kontrolle des Programminterpreters können diese Programme dem System keinen Schaden zufügen, somit ist auch keine Authentifizierung des Absenders nötig. Damit das System auch eine weite Verbreitung findet, wird von einer Benutzung eines speziellen Mail-Readers abgesehen, statt dessen wird ein Mechanismus<sup>24</sup> benutzt, mit dem viele Mail-Reader, infolgedessen auch viele Benutzer, eine Computational Mail lesen und ausführen können. Dieser Mechanismus prüft, ob der Body einer Mail (mittels dem Content-Type Header-

---

<sup>23</sup> vgl. Borenstein (1992)

<sup>24</sup> vgl. Borenstein (1991)

field) nur aus Text besteht. Wenn dies nicht der Fall ist, wird über eine „mailcap“-Datei ein Programm oder Programminterpreter aufgerufen, der diesen Datentyp darstellen kann. Dieser Ansatz macht sich diesen Mechanismus zunutze, indem es den zugehörigen Interpreter aufruft, der dann die Mail abarbeiten kann (siehe auch Kapitel 2.4.1).

### **Realisierung**

Der oben beschriebene Ansatz wurde mit einer wohldefinierten Sprache realisiert, ATOMICMAIL. ATOMICMAIL wird mit einem Interpreter (ATOMICMAIL-Interpreter) ausgeführt, der die Programmkonstrukte der Mail lesen und ausführen kann. Der Aufruf erfolgt über den oben genannten Mechanismus.

### **ATOMICMAIL**

ATOMICMAIL benutzt das grundlegende Sprachmodell von LISP. Jedoch wurde ATOMICMAIL soweit modifiziert, daß die Eingabe- und Ausgabemechanismen im Sinne der Sicherheit entfernt wurden und der Interpreter in jeder Umgebung benutzt werden kann, man spricht hier auch von einer *Sandbox-Security*.<sup>25</sup> Der Bellcore Prototyp des ATOMICMAIL-Interpreters benutzt die curses Bibliothek (eine Unixbibliothek zur Steuerung von Textterminals), um die Benutzerschnittstelle darzustellen. Somit läuft der Interpreter momentan nur auf UNIX-Terminals. Als LISP-Engine verwendet der Prototyp den „Embedded LISP Interpreter“ des Andrew Message Systems.

### **Realisierte Anwendung mit ATOMICMAIL**

In ATOMICMAIL wurden verschiedene Anwendungen implementiert:

#### 1. Umfragegenerator

Er schreibt und verschickt ATOMICMAIL Programme, die den Empfänger einige Fragen stellt und die Antworten des Empfängers als normale E-Mail automatisch zurücksendet.

#### 2. Meeting Scheduler

Erlaubt dem Absender ein Meeting einzuberufen, indem er den Grund, eine Liste der Teilnehmer, mögliche Tage und Zeiten des Meetings spezifiziert. Jeder Teilnehmer wird mit einer ATOMICMAIL aufgesucht und die Ergebnisse werden dem Initiator zurückgemeldet.

---

<sup>25</sup> vgl. Fritzing (1996)

### 3. Dokumentenverteiler

Er unterstützt den Benutzer in der Distribution von Dokumenten, die ATOMICMAIL gibt jedem Empfänger eine Liste von Dokumenten und diese können per Auswahl automatisch bestellt werden.

### 4. Expertenbefragung

Hiermit kann der Benutzer einen Experten via ATOMICMAIL um Rat fragen. Wenn dieser keine Antwort kennt, so kann er die ATOMICMAIL an andere Experten weitergeleitet werden, bis eine zufriedenstellende Antwort gefunden wurde.

### 5. Activist Alert Mechanismus

Der Benutzer wird zu einem aktuellen politischen Thema befragt und kann dann seine Meinung hinzufügen. ATOMICMAIL schickt dem Verantwortlichen, z.B. einem Abgeordneten, entweder eine E-Mail oder sendet ein Fax mit der Meinung des Benutzers.

### 6. „Erkennen Sie die Melodie?“

Dies ist ein Spiel, indem der Benutzer die Melodie einer Audio-Datei erraten muß.

Dieser Ansatz löst zwar die Probleme der „entfernten Installation“, der „Integration neuer Teilnehmer“ und die „Unterstützung heterogener Systeme“, aber es ist zu bezweifeln, ob mit ATOMICMAIL anspruchsvolle Anwendungen (auch im Bereich von Multimedia) entwickelt werden können.

## 2.5.2.2 Active Mail von Goldberg<sup>26</sup>

Dieses Konzept erweitert die normale E-Mail mit active messages, welche Kommunikations-Ports<sup>27</sup> enthalten, um das Dilemma der bisherigen CSCW-Anwendungen zu beseitigen, die entweder zu *passiv* oder zu *aufdringlich* sind. *Passiv* in dem Sinne, daß der Benutzer selber aktiv werden muß, um in einer CSCW-Anwendung kooperativ zu arbeiten. Er muß eigenständig die Anwendung starten oder eine Verbindung zu ihr herstellen. Am Beispiel einer verteilten Dokumentenerstellungsanwendung wird er nicht sofort in Kenntnis gesetzt, wenn eine neue Version eines Dokumentes vorliegt. Meist geschieht das erst, wenn er die Anwendung startet oder der Autor ihn in Kenntnis setzt, z.B. über E-Mail, daß eine neue Version vorliegt. *Aufdringlich* in dem Sinne, daß die CSCW-Anwendung den Benutzer bei seiner bisherigen Arbeit stört, indem sie plötzlich

---

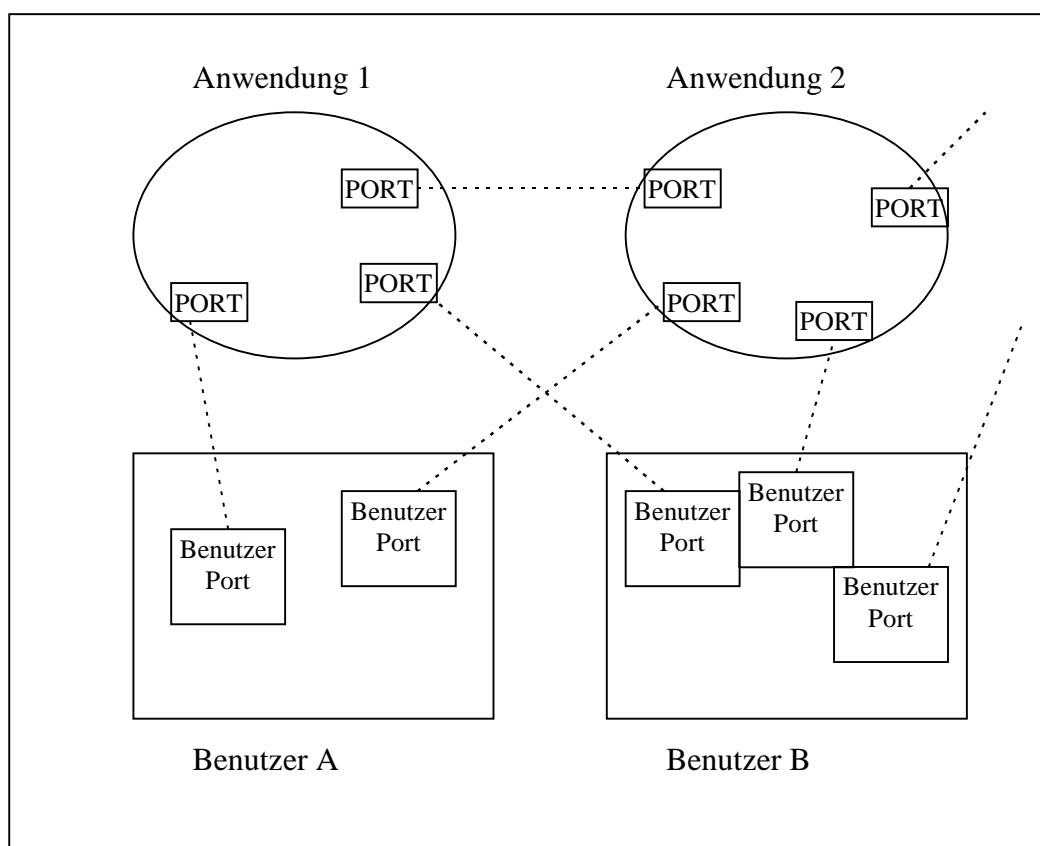
<sup>26</sup> vgl. Goldberg (1992)

<sup>27</sup> Kommunikations-Ports werden in diesem Ansatz verwendet, um eine synchrone bidirektionale Kommunikation zwischen Anwendung und Anwender bzw. mehreren Anwendern über Ports herzustellen.

auf seinem Bildschirm erscheint. Um beim Beispiel der Dokumentenerstellungsanwendung zu bleiben, könnte die Anwendung den Benutzer in Kenntnis setzen, daß eine neue Version eines Dokumentes vorliegt, auch wenn der Benutzer dies nicht unbedingt will. Der Ansatz Active Mail von Goldberg benutzt die Vorteile der normalen E-Mail, die nur benutzerinitiiert ist, verzögerte Antwortzeiten toleriert und einfach zu bedienen ist und bietet die Möglichkeit eine bidirektionale Kommunikation zu anderen Benutzern oder Anwendungen zu schaffen.

### Architektur

Die informelle Beschreibung der Architektur sieht folgendermaßen aus:



**Abbildung 5: Active Mail Architektur von Goldberg**

Die Architektur besteht aus Agenten, verbunden über Ports mit einem bidirektionalen Kommunikationskanal. Es gibt zwei Arten von Agenten, einen Benutzeragenten (in der Abbildung 5 als Benutzerport dargestellt) und einen Anwenderagenten (in der Abbildung 5 als Anwendung mit einer Anzahl von Ports dargestellt). Jeder der Active Mail Agenten ist mit einer bestimmten Anzahl von Ports ausgestattet und kann seine Verbindungen mit folgenden Grundoperationen verändern:

### 1. CREATE

Ein Agent kann einen neuen Applikationsagenten erzeugen und erhält automatisch eine Verbindung zu einem anderen Agenten.

### 2. SEND

Ein Agent kann eine Kopie einer seiner Ports an einen anderen Agenten schicken und damit dem anderen Agenten (Anwendung oder Benutzer) einen Zugang zur Anwendung verschaffen.

### 3. DISCARD

Ein Agent kann mit dieser Operation einer seiner Ports beenden oder ablegen.

Jeder Agent kommuniziert, d.h. sendet und empfängt Nachrichten über seine Ports. Abstrakt gesehen interagieren Benutzer mit Anwendungen nur über diese Ports, wobei die Darstellung dieser Ports auf der Benutzeroberfläche mit Fenstern realisiert ist. Jede Anwendung von Active Mail hat ein spezifisches Anwendungsfenster, das ihre Funktionalitäten unterstützt und eine einheitliche Oberfläche, um mit anderen Benutzern oder Anwendungen zu interagieren. Nebenbei wird eine Benutzerschnittstelle angeboten, um neue Anwendungen zu erzeugen und um die Ports dieser Anwendungen zu verwalten.

## **Beispiel**

Der Benutzer Rene will von einem anderen Gruppenmitglied einen Text geschrieben bekommen. Anstatt eine normale E-Mail zu schreiben benutzt Rene Active Mail. Er initiiert eine schon vorgefertigte Anwendung, den Conversation Agent, der eine nachvollziehbare Konversation erlaubt, und schickt diese (einen Port zu dieser Anwendung) mittels der SEND-Operation an Wendel, in der er ihn bittet, einen Text zu schreiben. Wendel bekommt diese Active Mail des Conversation Agent in seinem eigenen Active Mail Input-Folder und antwortet Rene. Er erzeugt seinerseits eine Active Mail mit einer Shared-Document Agent Anwendung, um die erste Version seines Dokumentes zu erzeugen. Nachdem er den ersten Entwurf fertig hat, schickt er Rene diese Active Mail (einen Port zu dieser Anwendung) und erlaubt eine Veränderung seines Dokumentes. Rene verändert dieses Dokument und gibt die neuere Version des Dokumentes frei, benachrichtigt Wendel über den Conversation Agent, dieses Dokument nochmals zu überarbeiten. Nachdem Wendel feststellt, daß er Probleme damit hat, schickt er Richard beide Active Mails, d.h. einen freien Port zum Conversation Agent und zum Shared-Document Agent, damit er ihm helfen kann. Richard bekommt diese Active Mails und kann nachvollziehen, welche Konversationen bisher statt gefunden haben und welche Versionen des Dokumentes entstanden sind (mit darunter liegenden Mechanismen wird dies ermöglicht). Nachdem Richard auch keinen besseren Text schreibt, beschließt Rene

ein Seminar über „Wie schreibe ich einen Text“ zu halten und ruft eine weitere Anwendung auf, den Meeting Scheduler, und schickt diese an Wendel und Richard. Mit dieser Anwendung einigen sie sich auf einen Termin für das Seminar.

### **Realisierung**

Dieses Active Mail System wurde mit Hilfe der verteilten Programmierumgebung von Logix in einer Untermenge der nebenläufigen logischen Programmiersprache FCP realisiert, die die Möglichkeit bietet, Kommunikations-Ports zu migrieren. Die Oberfläche wurde mit der XView Library, einer Schnittstelle zum X Window System, implementiert. Ein Server der Active Mail stellt alle Anwendungen bereit und betreut alle registrierten Benutzer. Der Server ist persistent mit Hilfe von Checkpoints und Recovery-Mechanismen, um Fehler aufzufangen. Um Active Mail zu starten, erzeugt der Benutzer einen Client Prozeß, der ihn mit dem Server und mit seiner Mailbox verbindet. Jeder Server kann mehrere Clients unterstützen. Das ganze System gliedert sich in einen anwendungsunabhängigen Kern und Anwendungen die in zwei Teile, einer Anwendungslogik und einer Darstellung, aufgeteilt sind.

Verglichen mit der normalen E-Mail hat dieses System nur die rudimentären Eigenschaften der E-Mail; es unterstützt die einfachen Operationen (SEND, CREATE und DISCARD), sowie zwei Speicherbereiche:

- **Inbox**  
um eingehende Mails auf dem Server zu speichern, bis sie vom Benutzer „abgeholt“ werden
  
- **lokale Verzeichnisse**  
um eingegangene Mails zu speichern und zu verwalten.

Ebenso toleriert das System längere Antwortzeiten. Jedoch sind die darunterliegenden Mechanismen zur normalen E-Mail sehr verschieden. Sobald eine Mail angenommen wird, entsteht eine direkte Kommunikation mit dem Absender (keine asynchrone, wie bei der normalen E-Mail). Die Mailbox dient dazu, persistente interaktive Verbindungen zwischen Absender, Anwendungen und Empfänger zu verwalten und nicht als Sicherung eingegangener Nachrichten.

### **Kritik**

Dieses Konzept der Active Mail von Goldberg bietet zwar große Vorteile zur Unterstützung von Gruppenarbeit in verteilten Systemen, ist aber unzureichend in den anderen Anwendungsgebieten der Active Mail. Es wird keine lokal abarbeitende Active Mail

unterstützt. Eine Active Mail kann nach diesem Ansatz nur mit einer ständigen Verbindung zu einem Server laufen. Dieser Ansatz eignet sich somit nicht für die mobile Kommunikation, sowie für alle Möglichkeiten, die in den Bereich von lokaler Kommunikation fallen. Ebenso ist die Benutzerzahl einer Anwendung beschränkt, da nur eine beschränkte Anzahl von Ports pro Anwendung unterstützt wird. Somit machen z.B. dynamische Nachrichtenverteiler oder Fragebögen mittels Active Mail keinen Sinn.

Durch die einfache Benutzung des Systems, d.h. durch die Ähnlichkeit zu bestehenden normalen E-Mail Programmen und deren Funktionalität, wird die Integration von neuen Teilnehmern in gruppenunterstützende Anwendungen erleichtert. Die Probleme der entfernten Installation und der heterogenen Plattformen bleiben aber weiterhin bestehen.

### **2.5.3 Active Mail für Multimedia**

Multimedia Mail ist eine Erweiterung der normalen E-Mail: sie ermöglicht eine Übertragung von multimedialen Daten, wie Dokumenten, Videos, Tondaten oder Bilder (siehe auch Kapitel 2.4.1 Multi-purpose Internet Mail Extension (MIME)). Diese sind jedoch alle statisch oder passiv in der E-Mail eingebettet und können nicht aktiv werden, d.h. entscheiden, welche Teile der E-Mail in welcher Reihenfolge dargestellt werden sollen, ebenso fehlt eine Interaktion mit dem Benutzer bzw. Empfänger. Um dies zu beheben, kann das Konzept der Active Mails helfen, indem sie diese multimedialen Daten der E-Mail mit den beigefügten ausführbaren Programmen strukturieren und anzeigen.

#### **2.5.3.1 ActiveM<sup>3</sup><sup>28</sup>**

ActiveM<sup>3</sup> (Aktive Multimedia-Mail) ersetzt Multimedia Mail und integriert zwei Konzepte: Active Mail und Multimedia Mail. Es wird in der Abteilung "Mobile Information Visualization" im Zentrum für Graphische Datenverarbeitung (ZGDV) in Darmstadt entwickelt.

Auf der Basis elektronischer Post (E-Mail) werden aktive multimediale Botschaften unterstützt. Diese können beispielsweise selbständig auf der Empfängerseite ein Formular aufbauen, die Antworten des Empfängers entgegennehmen und auswerten, um dann automatisch eine Antwort an den Absender zurückzusenden. Ein Autorenwerkzeug erlaubt die grafisch-interaktive Erstellung aktiver multimedialer Botschaften.

---

<sup>28</sup> vgl. Schirmer (1996a)

## Modell

Die Idee ist, eine Active Mail mit Multimedia Elementen auszustatten, und diese via E-Mail zu verschicken. Der aktive Teil („Hiker“) übernimmt die Strukturierung der Multimedia Daten und stellt sie entsprechend dar. Dies ermöglicht ein Austausch von interaktiven multimedialen Anwendungen über den Kommunikationskanal E-Mail. Die zentrale Komponente des entwickelten Prototypen ist der intuitiv bedienbare „mailhandler“, der ActiveM<sup>3</sup>-Composer, der eine Active Mail leicht erstellbar macht. Das zugrundeliegende Konzept des ActiveM<sup>3</sup>-Composers basiert auf Formularen („Forms“), um eine Active Mail zu generieren. Auf dieses Formular können verschiedene Komponenten „gelegt“ werden. Multiple Choice Felder, statische und editierbare Felder erlauben eine Interaktion mit dem Benutzer. Spezielle „action buttons“ ermöglichen eine Sammlung oder Auswertung der eingegebenen Daten und schicken die Antwort zum Absender zurück. Bilder können ebenso integriert werden.

Jede Active Multimedia-Mail besteht aus einem vorgefertigtem Formular, welches der Benutzer nach seinen Wünschen verändern kann (siehe auch Abbildung 6).

Dieses Formular hat vier Bereiche, eine **SubjectArea**, eine **WorkArea**, eine **MessageArea** und eine **CommandArea**. Die SubjectArea enthält den Betreff der Active Mail und die Signatur des Absenders, die MessageArea gibt dem Benutzer eine Rückmeldung, wenn bestimmte Aktionen ausgeführt werden. Die WorkArea besteht aus verschiedenen Elementen (Text, Eingabefelder, Label, Bilder und Multichoice Feldern). Die CommandArea stellt einige Standard-Buttons zur Verfügung (Exit, Save, Open, Reply, Forward). Die Antworten werden automatisch gesammelt und an den Absender zurückgeschickt, entweder als Active Mail oder von einer Anwendung gesammelt und ausgewertet.

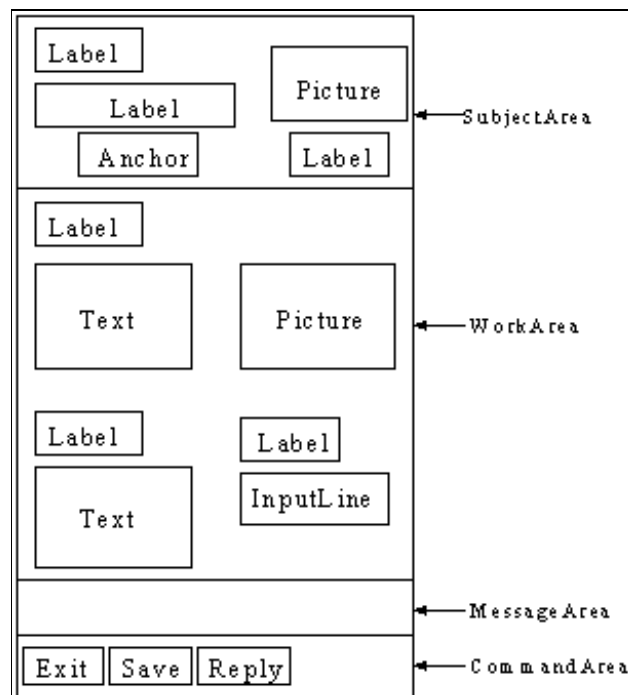
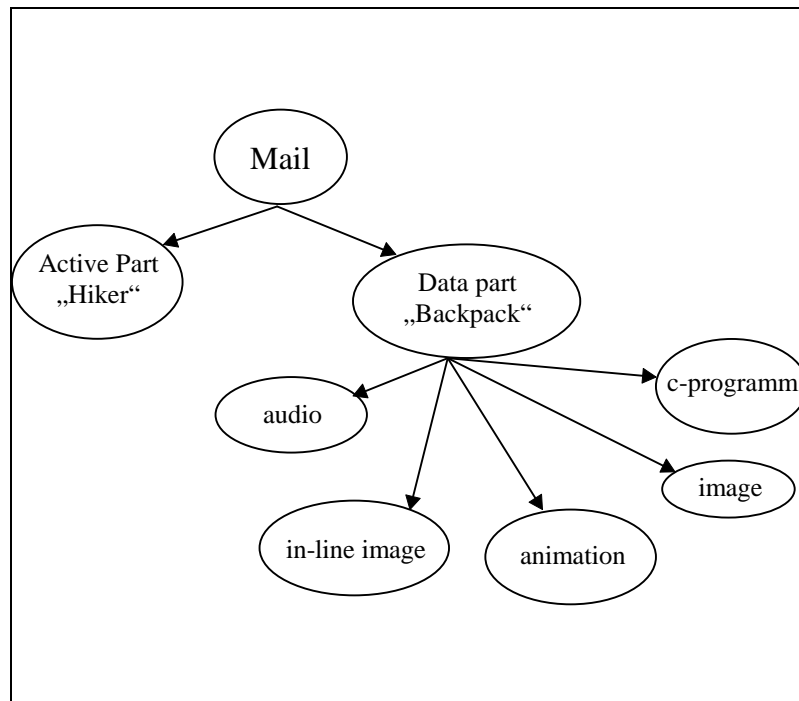


Abbildung 6: Formular in ActiveM<sup>3</sup>

Das Formular dient als Quelle für Kommentare und Anmerkungen zu den mitgelieferten Multimedia Objekten, den Backpack-Objekten (siehe auch Abbildung 7). Die Back-

pack-Objekte werden als Attachment der Active Mail beigelegt und der Aktive Part („Hiker“) präsentiert diese Objekte am Zielort. Ein Anker im Formular läßt die Präsentation der Objekte entweder einzeln oder je nach Synchronisierungsregeln strukturiert in Raum und Zeit ablaufen. Der „Hiker“ könnte ebenso ein mitgeliefertes Programm beim Empfänger installieren und eine persistente Verbindung zu einer verteilten Anwendung herstellen.



**Abbildung 7: Die Struktur einer Active Mail in ActiveM<sup>3</sup>**

### Realisierung

Um das Verhalten von Active Mail Messages in ActiveM<sup>3</sup> darzustellen, benutzt das System einen LISP Dialekt, HCL, entwickelt am selben Institut. HCL beinhaltet Motif/Windows Funktionalität, um grafische Benutzeroberflächen zu generieren, ebenso ist eine Socket basierte Kommunikation möglich. Das Composing Tool ist auch in HCL geschrieben und kann somit selbst als Active Mail verschickt werden, um eine einfache Installation zu ermöglichen.

Das Format der Active Mail ist der Multi-purpose Internet Extension (MIME) angepaßt und besteht aus zwei Teilen. Der erste Teil besteht aus einer linearen Anordnung der multimedia Objekten, dem Backpack. Der zweite Teil beinhaltet die aktive Komponente, das Skript, das die Anzeige des Backpacks organisiert. Die ganze Nachricht wird als MIME multipart/X-hcl gesendet. Das Backpack wird im definierten Standard multipart/mixed von MIME geschachtelt und kann somit auch ohne den Aktiven Teil der

Nachricht empfangen und ausgewertet werden. Der aktive Part wird als Applikation/X-hcl gesendet und zeigt dem Empfänger an, diesen Teil mit einem vorinstallierten HCL-Interpreter auszuführen. Der User Agent des Empfängers wertet die ganze Nachricht aus und macht dem Aktiven Teil das Backpack zugänglich.

Der Synchronisierungsmechanismus basiert auf einem endlichen Zustandsautomaten, um die einzelnen Objekte des Backpacks in Raum und Zeit darzustellen. Um die Synchronisierungsregeln aufzustellen, erlaubt das Composing Tool eine einfache grafische Manipulation der multimedia Objekte, angelehnt an die Logik von Petrinetzen.

### 2.5.3.2 Mobile Agenten mit ActiveM<sup>3</sup><sup>29</sup>

In der Abteilung "Mobile Information Visualization" im Zentrum für Graphische Datenverarbeitung (ZGDV) in Darmstadt, die auch das ActiveM<sup>3</sup> entwickelt (siehe oben), wird zur Zeit untersucht, wie sich die Konzepte der grafischen Generierung von Active Mails auch für die Generierung von Mobilien Agenten eignet.

Mit dem ActiveM<sup>3</sup> Werkzeug lassen sich Active Mails grafisch programmieren, d.h. ein Benutzer kann ohne Programmiererfahrung „Programme“ entwickeln. Active Mails können in einer abstrakteren Sichtweise auch als *aktive Objekte* angesehen werden, die im Netzwerk „wandern“ können, um Informationen zu suchen, zu verbreiten oder zu sammeln. Sie können ferner mit Empfängern interagieren, Entscheidungen aufgrund dieser Interaktion treffen, sich selbst vervielfältigen usw.. Mobile Agenten haben somit ähnliche Eigenschaften (siehe auch Kapitel 3) wie Active Mail.

Schirmer<sup>30</sup> sieht die Konstruktion von Mobilien Agenten ähnlich wie die Gestaltung von Active Mails. Das Composing Tool von ActiveM<sup>3</sup> könnte hier als Ausgangspunkt benutzt werden, um Mobile Agenten auch grafisch interaktiv zu entwickeln. Dazu muß jedoch das Composing Tool von ActiveM<sup>3</sup> angepaßt werden, d.h. es müssen zusätzliche Komponenten eingebunden werden, z.B. eine Leiste und eine zusätzliche Sicht, in der ein Benutzer seinen Agenten konstruieren kann und direkt eine Rückmeldung seiner Entwicklung bekommt. Diese zusätzlichen Komponenten könnten auch von Dritten angeboten werden, um damit die Befehle bzw. Eigenheiten der Mobilien Agenten zu realisieren (z.B. Aussagen wie go, places, ticket oder password<sup>31</sup>). Das Verhalten eines Agenten läßt sich mit den Synchronisierungsmechanismus (siehe oben) von ActiveM<sup>3</sup>

---

<sup>29</sup> vgl. Schirmer (1996b)

<sup>30</sup> vgl. Schirmer (1996b)

<sup>31</sup> Dies sind Befehle des General Magic's Magic Cap und Telescript System.

grafisch modellieren, jedoch treten Schwierigkeiten bei der grafischen Programmierung vom Grad der Autonomie, Intelligenz, Verhalten und Mobilität auf.

Die Verschickung des entwickelten Agenten könnte ebenso mit dem Composing Tool erfolgen, welches eine MIME konforme Verschickung übernimmt. Der Vorteil ist, daß Mobile Agenten in eine weitverbreitete Kommunikationsumgebung, die der E-Mail, eingebettet wird.

Die Konstruktion eines Mobilen Agenten in einer grafisch intuitiven Weise ist mit Sicherheit ein schwieriges Unternehmen und es bleibt zu bezweifeln, ob sämtliche Funktionalität eines Mobilen Agenten grafisch zu realisieren ist. ActiveM<sup>3</sup> ist mit Sicherheit ein erster Schritt, dies zu untersuchen. Leider liegen zur Zeit der Diplomarbeit keine weiteren Forschungsergebnisse vor, um diesen Sachverhalt näher zu bestimmen.

#### **2.5.4 Sprache einer Active Mail<sup>32</sup>**

Die Sprache ist eine der wichtigsten zu berücksichtigenden Aspekte, denn sie bildet das Herz einer Active Mail. Nur mit einer Programmiersprache kann eine Active Mail aktiv werden und in Interaktion mit dem Benutzer treten. Hier sind jedoch viele Punkte zu berücksichtigen, wie man schon in den vorherigen Kapiteln sehen konnte. Zum einen steht hier die Frage der Sicherheit einer Programmiersprache an erster Stelle. Unter diesem Aspekt ist auch die „Mächtigkeit“ einer Sprache zu sehen. Je mächtiger eine Sprache ist, desto mehr kann man mit ihr auf der einen Seite realisieren; auf der anderen Seite wächst die Gefahr, daß sie für das System nachteilige Auswirkungen haben kann. Ebenso darf der Aspekt der Portabilität nicht vernachlässigt werden. Authentifizierung und die Qualität der Benutzerschnittstelle sind weitere Eigenschaften einer Sprache, die im Hinblick auf Active Mail zu berücksichtigen sind.

##### **Sicherheit**

Wie schon erwähnt, ist die Sicherheit die wichtigste Bedingung für Active Mail. Einfach ausgedrückt, muß es möglich sein, einem anderen Benutzer eine Active Mail zu lesen, ohne daß nach Ausführung der Active Mail Schaden angerichtet wird. In diesem Zusammenhang ist es nicht akzeptabel, eine außergewöhnlich „mächtige“ Sprache zu benutzen, mit der es einem Benutzer möglich ist, Active Mails zu verschicken, die beim Empfänger gleichzeitig Daten löscht, vertrauliche Daten stiehlt und an andere weiter-

---

<sup>32</sup> vgl. Borenstein (1994)

leitet, Viren einschleust oder sonstigen Schaden anrichtet. Generell kann man drei große Bereiche ausmachen, die eine potentielle Gefahr darstellen<sup>33</sup>:

1. Löschen von Ressourcen (z.B. Dateien)
2. Diebstahl von Ressourcen (z.B. privaten Dateien)
3. Entziehung von Ressourcen (z.B. Vollschieben einer Festplatte oder Auslastung der CPU)

Wenn die Sprache in diesen Bereichen keine Funktionen anbietet, kann sie als sicher gelten. ATOMICMAIL (siehe auch Kapitel 2.5.2.1) ist ein Beispiel einer Sprache für Active Mail, die soweit „beschnitten“ wurde, daß sie dem System, in den oben genannten Bereichen, nicht Schaden kann. Die Idee ist sehr einfach, aber schwer umzusetzen: Man entfernt alle Merkmale einer Sprache, die einem System Schaden zufügen können und ersetzt diese durch weniger allgemeine Funktionalität. Somit erhält man eine sichere Untermenge einer Sprache, z.B. könnte man den allgemeinen Mechanismus für Lesen und Schreiben von Dateien so umschreiben, daß er nur noch auf „öffentlichen“ Dateien seine Funktionalität behält. Diese Sicherheit führt aber zu einem weiteren Problem, dem der „Mächtigkeit“ einer Sprache.

### **Mächtigkeit der Sprache**

Nachdem eine Sprache für Active Mail sicher gemacht ist, kommt die Frage nach der Mächtigkeit der Sprache auf. Die gefährlichsten Merkmale einer Sprache in Bezug auf Active Mail sind gleichzeitig die allgemeinsten einer Sprache. Ohne diese wird eine Sprache weniger mächtig bzw. ohne sie wäre es schwer, sinnvolle Anwendungen mit dieser Sprache zu entwickeln. Eine Sprache so zu gestalten, daß sie auf der einen Seite vollkommen sicher ist und auf der anderen Seite genügend Merkmale hat, um sinnvolle Anwendungen zu schreiben ist ein Balanceakt besonderer Art, da auch die Entwickler einer Sprache nicht vorhersagen können, welche Merkmale für bestimmte Anwendungen wichtig sein werden und welche nicht. Prinzipiell wäre es wünschenswert, eine Active Mail Sprache zu haben, in der ein Benutzer auch spezielle Operationen definieren kann, die in einer unbeschränkten Umgebung laufen. Dieser Punkt läßt jedoch die Frage der Authentifikation aufkommen.

### **Authentifizierung**

Sobald die Active Mail mit Funktionen, die nur in unbeschränkten Umgebungen laufen, ausgestattet ist, muß eine Authentifizierung vorhanden sein. D.h. es muß sichergestellt

---

<sup>33</sup> vgl. Borenstein (1992)

sein, daß der Absender einer solchen Active Mail auch wirklich dieser ist, damit der Empfänger entscheiden kann, ob er dem Absender „trauen“ kann, und diese Active Mail akzeptiert oder im anderen Falle einfach abweist. Dem Mail Header einer E-Mail zu trauen wäre in diesem Zusammenhang gefährlich naiv, da dieser mit Leichtigkeit geändert bzw. gefälscht werden kann. Hier könnte die Technologie von Privacy Enhanced Mail<sup>34</sup> (PEM) oder Pretty Good Privacy<sup>35</sup> (PGP) eingesetzt werden, um eine Authentifizierung sicherzustellen. Idealerweise könnte eine Sprache von Active Mail automatisch diese Technologie miteinbinden, oder der Benutzer muß dies manuell machen.

### **Portabilität und Qualität der Benutzerschnittstelle**

Portabilität ist von großer Bedeutung für Active Mail, da sie auf unterschiedlichsten Rechnerplattformen laufen muß. Portabilität betrifft nicht nur, eine portable Software für unterschiedliche Plattformen zu schreiben, sondern ist gerade für Active Mail eine besonders wichtige Anforderung, da ein Absender einer Active Mail meist nicht vorher-sagen kann, auf welcher Plattformart seine Active Mail geschickt wird. ATOMICMAIL hat hier einen Anfang gemacht, gewisse Funktionen zu abstrahieren, um diese auf den unterschiedlichsten Rechnerplattformen darzustellen. Jedoch fehlt ATOMICMAIL jegliche Qualität in der Darstellung und es sind auch nur einige sehr allgemein gehaltene Funktionen verfügbar (siehe Kapitel 2.5.2.1). Um moderne Anwendungen schreiben zu können und eine breite Akzeptanz beim Benutzer zu erlangen, muß eine Sprache die Möglichkeit bieten optisch wie auch funktionell anspruchsvolle Anwendungen bzw. Active Mails entwickeln zu können. Safe-Tcl scheint vielen der oben genannten Anforderungen zu genügen.

### **Safe-Tcl**<sup>36</sup>

Safe-Tcl ist ein Mechanismus, der die Ausführung eines Programmes, welches in Tcl geschrieben ist, kontrollieren kann. Safe-Tcl kann in unbekanntem „Scripts“ (Applets) ausgeführt werden, ohne das die Rechnerumgebung beschädigt wird oder private Informationen preisgegeben werden. Jedes Applet wird dabei in einen sicheren Interpreter isoliert und kann nicht direkt mit anderen Anwendungen interagieren. Die Ausführungsumgebung des sicheren Interpreters wird von sicheren „Scripts“ in einem Master-Interpreter gesteuert. Safe-Tcl stellt Mechanismen zur Verfügung, mit denen die unbekanntem Applets Dienste auf eine kontrollierte Art und Weise anfordern können. Safe-Tcl erlaubt auch die Definition von einigen Sicherheitsmaßnahmen, wie z.B. Authentifizierung der eingehenden „Scripts“. Safe-Tcl kann auch als eine Teilmenge von Tcl aufgefaßt werden, wobei die gefährlichen Funktionen von Tcl entschärft wurden. Die Be-

---

<sup>34</sup> vgl. Linn (1993)

<sup>35</sup> vgl. Pretty Good Privacy (1997)

<sup>36</sup> vgl. Borenstein (1994), Ousterhout (1996)

nutzerschnittstelle kann mit den Befehlen von Tcl generiert werden, es können aber auch generische Befehle in der Art von ATOMICMAIL für beliebige Benutzerschnittstellen (grafische oder textbasierte) definiert werden. Safe-Tcl ist auch MIME-fähig und bietet somit die einfache Einbindung von multimedialen Daten sowie die einfache Versendung der Active Mail ohne einen dazugehörigen User Agenten zu benötigen. Das Lesen einer Active Mail kann mit jedem Mail-Reader geschehen, der MIME-fähig ist und mittels der „mailcap“-Datei (siehe auch Kapitel 2.4.1) einen Safe-Tcl Interpreter aufrufen kann.

### **2.5.5 Anforderungen an ein Active Mail System**

Welche Anforderungen kann man nun allgemein an ein Active Mail System stellen?

#### **Sicherheit**

Sicherheit in der Ausführung muß gegeben sein. Ist diese nicht gegeben, wird eine Authentifikation der Active Mail selbst und ggf. des Absenders, sowie eine Autorisierung<sup>37</sup> notwendig.

#### **Unterstützung von heterogenen Systemen**

Es muß gewährleistet sein, das Active Mails auf heterogenen Plattformen nutzbar sind. Auf der einen Seite muß der User Agent bzw. das Mail-Reading-Programm für die verschiedenen Betriebssysteme vorliegen, auf der anderen Seite muß die Sprache der Active Mail auf heterogenen Systemen gleich ablaufen<sup>38</sup>.

#### **Qualität der Benutzerschnittstelle**

Eine hohe Qualität in der Darstellung der Active Mails wird notwendig, um moderne Anwendungen entwickeln zu können und eine breite Akzeptanz bei Benutzern zu erzielen.

---

<sup>37</sup> Unter Autorisierung versteht man das Zulassen bestimmter Aktionen durch Benutzer oder Programme.

<sup>38</sup> Durch einen Interpreter, der das Programm der Programmiersprache nach der notwendigen syntaktischen Überprüfung sofort ausführt, kann dies gewährleistet werden. Dabei muß jedoch für die verschiedenen Systeme ein Interpreter vorliegt.

## **Unterstützung von verschiedenen Anwendungsbereichen**

Diese Anforderung wird hauptsächlich durch die *Mächtigkeit der Sprache* definiert. Es sollte möglich sein, Anwendungen für die Bereiche CSCW, Multimedia, Mobile Kommunikation u.a. zu entwickeln.

## **Funktionalität der Benutzerschnittstelle wie normale E-Mail Programme**

Eine benutzerfreundliche Schnittstelle sollte bereitgestellt werden, um die Verwaltung, Erzeugung, Verschickung und Kontrolle der Active Mails einfach zu ermöglichen. Wenn die Ausführung und Erzeugung von Active Mails von einem Programm aus möglich ist, sollte die Grundfunktionalität des Programms herkömmlichen E-Mail Programmen gleichen. Auf der einen Seite erleichtert es dem Benutzer den Einstieg bzw. die Nutzung des Programms und auf der anderen Seite hat man einen fließenden Übergang von normalen Mail-Readern zum Active Mail User-Agenten.

## **Verschiedene Möglichkeiten der Erzeugung**

Für die Erzeugung einer Active Mail sollte auf der einen Seite für den unerfahrenen Benutzer mit einer „**high-level**“ Sprache, z.B. mit *vorgefertigten Anwendungen*<sup>39</sup> oder sogenannten Formularen möglich sein, die nur noch parametrisiert werden müssen. Auf der anderen Seite sollte einem Experten die Möglichkeit geboten werden, selbst Anwendungen bzw. Formulare in einer „**low-level**“ Sprache zu schreiben, die einfach in das System eingebunden und auch von einem unerfahrenen Benutzern verwendet werden können.

## **Nicht aufdringlich**

Eine Active Mail sollte den Arbeitsablauf eines Benutzers nicht eigenständig stören, d.h. nicht plötzlich als neue Meldung auf dem Bildschirm auftauchen und den Benutzer dazu veranlassen, ungewollt seine derzeitige Tätigkeit zu unterbrechen und auf die Active Mail einzugehen. Wünschenswert wäre hier eine Funktionalität, wie sie bei normalen E-Mail Programmen schon gegeben ist: Die **Benachrichtigung**, daß eine neue Mail vorliegt erfolgt, entweder nur, wenn dies der Benutzer explizit nachfragt (z.B. mit einem Checkmail Befehls) oder wenn der Benutzer durch eine Statusänderung eines kleinen Bildes (Icons) auf dem Bildschirm davon in Kenntnis gesetzt wird.

---

<sup>39</sup> Da ausführbare Programme mit einer Active Mail verschickt werden können, können Anwendungen entweder als „eigenständige“ Anwendungen eines Active Mail Systems verstanden werden oder aber als Anwendungen, welche ein Active Mail System nur dazu benutzen, eine Verbindung zu einer anderen Anwendung herzustellen, gewissermaßen Active Mail als Vehikel, um andere Teilnehmer zu einer „außenstehenden“ Anwendung einzuladen.

## **Einfache Installation**

Damit eine Anwendung erfolgreich wird bzw. auch benützt wird, ist es erforderlich, daß sie nicht von Experten installiert werden muß, sondern auch von Einsteigern installiert werden kann. Dies ist im besonderen Maße notwendig, wenn das Active Mail System für Anwendungen im Bereich der Gruppenunterstützung in einer verteilten Umgebung verwendet werden soll.

## **Verschiedene Antwortzeiten tolerieren**

Unter Antwortzeiten ist die Zeitspanne zu verstehen, die zwischen dem Abschicken einer Active Mail und der Antwort des Empfängers oder der Empfänger liegt. Diese kann bei normaler E-Mail zwischen Minuten und Wochen liegen und ein Active Mail System sollte diese Eigenschaft auch unterstützen. Jedoch muß hier differenziert werden, in welchem Kontext die Antwort und die daraus entstehende Bedingung an eine Antwortzeit, entsteht; je nach Gestaltung und Aufgabe einer Active Mail kann die Bedeutung einer Antwort wichtig, weniger wichtig oder unwichtig sein.

### „Antwort ist für Active Mail (für den Initiator) unwichtig“

Dieser Fall kommt z.B. bei einer Verteilerliste vor: hier ist es dem Initiator nicht wichtig, daß jeder der eine Active Mail bekommt, dem Initiator eine Rückmeldung gibt. Es ist sogar mit Sicherheit nicht wünschenswert, daß jeder Nachrichtenempfänger dem Initiator eine Rückmeldung gibt und der Initiator in einer Flut von Rückmeldungen ertrinkt.

### „Antwort ist weniger wichtig für Active Mail“

Bei einer interaktiven Bestellliste könnte die Antwort zwar wünschenswert sein, jedoch ist sie nicht von zentraler Bedeutung, um die Funktionalität der Active Mail sicherzustellen.

### „Antwort hat zentrale Bedeutung für Active Mail“

Bei einer Active Mail, wie z.B. bei einem Fragebogen oder einer Terminvereinbarung, die unbedingt die Antworten des Empfängers benötigt, kann es sinnvoll erscheinen, die Antwortzeit des Empfängers zu beschränken. Bei einer Terminvereinbarung würde es keinen Sinn machen, wenn der Initiator erst nach Wunschtermin eine Rückmeldung der Teilnehmer bekommt. Somit macht die Active Mail keinen Sinn mehr, wenn sie nach diesem Zeitpunkt noch aktiv ist und sollte eigentlich deaktiviert werden.

### 2.5.6 Vorteile und Nachteile von Active Mail

Nach der ausführlichen Einführung in den vorherigen Kapiteln, lassen sich nun die wesentlichen Vor- und Nachteile eines Active Mail Systems wie folgt zusammenfassen:

#### Vorteile:

- **kann Programme ausführen**  
(In Interaktion mit Benutzern treten, eigene Benutzeroberfläche mitführen, Aufgaben in heterogenen Netzwerken erledigen, ...)
- **Flexibilität des Ansatzes**  
(Einsatz und Nutzung des Systems in verschiedenen Bereichen, asynchrone oder synchrone Kommunikation)
- **verschiedene Anwendungsbereiche**  
(CSCW, Multimedia, Mobile Kommunikation, automatisierte Datenbanktransaktionen, ...)
- **Nutzt Vorteile von E-Mail Konzepten**  
(Asynchrone Kommunikation, einfache Bedienung, 1-1,1-n, n-m Kommunikation)
- **Erweiterbarkeit**  
(durch Entwicklung neuer Anwendungen)
- **Aktivierung nur durch Benutzerinteraktion oder explizitem Befehl**
- **einfache Benutzung von verschiedenen Anwendungen**  
(mit vorgefertigten Formularen)

#### Nachteile:

- **keine allgemeine Architektur**  
(keine Standards, Mechanismen der Systeme unterscheiden sich, verschiedene Sprachen, Erzeugung, Verschickung und Aktivierung können unterschiedlich sein)
- **zusätzliche Komponenten notwendig**  
(Interpreter, ggf. eigener User-Agent)
- **Sicherheit ist problematisch**  
(keine Standards, unterliegt der Sprache oder dem System und Benutzer muß „vertrauen“)

- **Unterstützung von heterogenen Systemen hängt vom Interpreter ab bzw. für welche Systeme der Interpreter vorliegt**
- **nicht alle Benutzer können System nutzen**  
(Bedingt durch zusätzliche Komponenten und Portabilität)

Zusammenfassende Bewertung der beschriebenen Systeme:

(Die Bewertung erfolgt in 4 Stufen (++) sehr gut, (+) gut, (-) weniger gut, (--) nicht ausreichend.)

<b>System</b>	Computational Mail (CM)	Active Mail von Goldberg (AM)	ActiveM3 (AM3)	Bemerkungen
<b>Eigenschaften</b>				
Sicherheit	++	-	+	CM hat keine Funktionen, die als unsicher gelten
Unterstützung von heterogenen Systemen	+	-	+	AM verwendet keinen Interpreter, CM und AM3 verwenden Interpreter, die auf unterschiedliche System portiert werden können.
Mächtigkeit der Sprache	--	+	+	CM ist in der Funktionalität sehr beschränkt
Asynchrone Kommunikation	ja	beschränkt	ja	AM läuft nur mit einem Server (C/S Prinzip)
Synchrone Kommunikation	nein	ja	ja	Nicht unbedingt erforderlich, bietet aber große Flexibilität, z.B. im Bereich Gruppenarbeit
Lesbar mit normalem Mail-Reader	ja	nein	ja	AM benötigt speziellen Mail-Reader
einfache Erstellung von Active Mails	+	+	++	AM3 bietet die einfachste Erstellung mit einem speziellen Composer
Erweiterbarkeit	+	+	+	Entwicklung neuer Anwendungen
einfache Installation	++	-	++	CM und AM3 erlauben eine Installation mit einer Active Mail

**Tabelle 1: Gegenüberstellung der einzelnen Active Mail Systeme**

Die Gegenüberstellung macht deutlich, daß ActiveM<sup>3</sup> die wesentlichen Anforderungen für ein Active Mail System im Durchschnitt am besten erfüllt.

### 3 Agententechnologie

*„Agents will be the most important computing paradigm in the next ten years ... By the year 2000, every significant application will have some form of agent-enablement“<sup>40</sup>*

#### 3.1 Definitionen

Im Bereich der Informatik existieren viele verschiedene Definitionen von Agenten, die je nach Einsatzgebiet und Sichtweise sehr differieren. Um nicht das auf ganze Spektrum eingehen zu müssen, werden hier nur zwei Definitionen exemplarisch herausgegriffen.

Laut Definition des Brockhaus<sup>41</sup> wird ein Agent als „[lat. ‘Handelnder’], bezeichnet, wer im Auftrag eines anderen in dessen Interesse tätig wird.“

Woodridge<sup>42</sup>, beschreibt einen Agenten als ein unabhängiges Programm, das in der Lage ist, seine Entscheidungen und sein Handeln, aufgrund der Wahrnehmung seiner Umwelt, bei der Verfolgung eines oder mehrerer Ziele, selbständig zu kontrollieren. Er beschreibt auch weiterhin Eigenschaften, die ein Programm erfüllen muß, damit es als Agent angesehen werden kann<sup>43</sup>:

- **Autonomie:** Agenten lösen eine ihnen gestellte Aufgabe selbständig, d.h. ohne weiteres Einwirken des Benutzers.
- **Kooperationsfähigkeit:** Agenten sind in der Lage mit anderen Agenten oder Benutzern zusammenzuarbeiten bzw. zu kommunizieren.
- **Reaktionsfähigkeit:** Agenten beobachten ihre Umwelt und reagieren auf auftretende Veränderungen oder Ereignisse.
- **„pro-activeness“:** Agenten reagieren nicht nur auf Veränderungen oder Ereignisse ihrer Umgebung, sie können sogar, basierend auf ihren definierten Zielen, die Initiative ergreifen.

In der Literatur werden viele verschiedene Ansätze von Agenten und Agentenarten<sup>44</sup> genannt, wie z.B. Intelligente Agenten, Interface Agenten, Softbots, Believable Agenten, mobile Agenten, kooperierende Agenten. Auf dem der Künstlichen Intelligenz, der

---

<sup>40</sup> vgl. Gilbert (1995)

<sup>41</sup> Brockhaus (1952)

<sup>42</sup> Woodridge (1995), Jennings (1996)

<sup>43</sup> Die Autoren selbst beschreiben diese Eigenschaften als eine „schwache Vorstellung“ von Agenten, um die Eigenschaften so allgemein wie möglich zu halten.

<sup>44</sup> vgl. Magedanz (1996), Maes (1995), Hohl (1995), Hyacinth (1996)

Verteilten Systeme und der Benutzeroberflächen wurde der Begriff der **Software Agenten** geprägt. Eine Einteilung oder Taxonomie der einzelnen Agentenarten gestaltet sich recht schwer und wird von jedem Autor anders vorgenommen. Deshalb soll im folgenden Kapitel nur der Begriff der Software Agenten kurz erläutert werden.

### 3.2 Software Agenten

Nach Magedanz<sup>45</sup> sind Intelligente Agenten selbstenthaltende, relativ autonom operierende Softwarekomponenten, die für einen Teil eines Berechnungsablaufs verantwortlich sind<sup>46</sup>. Sie weisen eine oder mehrere der folgenden Eigenschaften auf:

1. Intelligenz, aufgrund der einprogrammierten Logik zur eigenen Steuerung und Wissensgewinnung. Diese Eigenschaft hängt stark von der benutzten Sprache ab, mit der ein Agent programmiert wird.
2. Asynchrone Operation, d.h. ein Agent kann seine Aufgabe oder sein Ziel entkoppelt von seinem Benutzer oder anderen Agenten ausführen.
3. Kommunikation mit Benutzern und verschiedenen Systemressourcen.
4. Kooperation mit anderen Agenten.
5. Mobilität des Agenten, um bestimmte Aufgaben auszuführen.

Man kann die verschiedenen Agententechnologien in zwei verschiedene Klassen einteilen. Zum einen in die Klasse der Einzelagentensysteme, zu der die Lokalen Agenten und die Netzwerkagenten gehören, zum anderen in die Klasse der Multiagentensysteme, zu der Agenten auf der Basis der Verteilten Künstlichen Intelligenz (VKI)<sup>47</sup> zählt sowie die Mobilen Agenten.<sup>48</sup>

Diese Diplomarbeit baut auf dem Konzept der Mobilen Agenten auf und wird deshalb im nächsten Kapitel speziell auf das Gebiet der Mobilen Agenten eingehen. Die anderen Agentenarten werden nicht weiter betrachtet.

---

<sup>45</sup> vgl. Magedanz (1996)

<sup>46</sup> Diese Definition trifft nach meiner Einschätzung auch auf die Software Agenten zu. Eine allgemeingültige und genaue Definition der Software Agenten lies sich leider zu diesem Zeitpunkt in der Literatur nicht finden.

<sup>47</sup> vgl. Müller (1993)

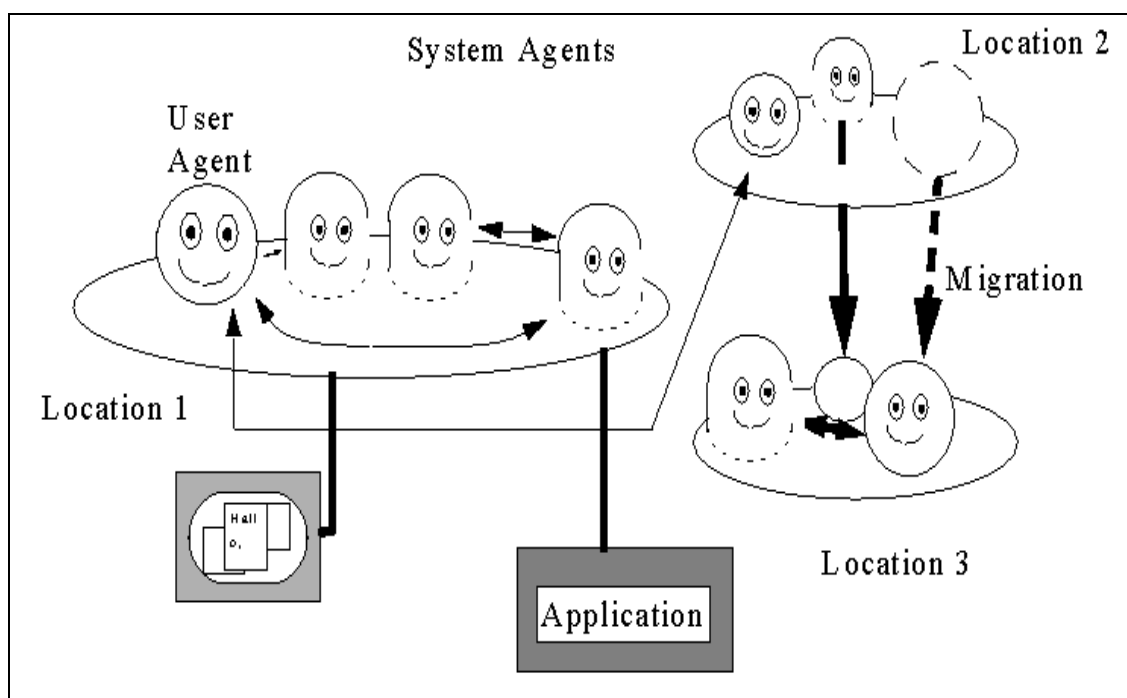
<sup>48</sup> vgl. Magedanz (1996)

### 3.3 Mobile Agenten

Mobile Agenten werden in großen verteilten Netzwerken eingesetzt und können in vielfältiger Weise verwendet werden (fortgeschrittene Internet Filter, Suchagenten, intelligenter Nachrichtenaustausch, intelligente Kommunikation, Management, Trading und Lastbalancierung).<sup>49</sup>

Die Eigenschaften, die in Kapitel 3.2 für Intelligente Agenten genannt wurden, gelten auch für mobile Agenten; ergänzend soll hier eine etwas genauere Definition bzw. Eingrenzung von Mobilen Agenten vorgenommen werden.

Mobile Agenten sind aktive, autonome Objekte, die sich zwischen Orten<sup>50</sup> in einem sogenannten Agentensystem bewegen können. Unter einem Agentensystem versteht man eine verteilte abstrakte Schicht, die auf der einen Seite dem darunterliegenden System Sicherheit bietet und auf der anderen Seite Konzepte und Mechanismen für Mobilität und Kommunikation zur Verfügung stellt (siehe auch Abbildung 8).<sup>51</sup>



**Abbildung 8: Die Struktur eines Agentensystems**

Mobile Agenten werden meist in einer interpretierten Hochsprache entwickelt, wie Safe-Tcl, SUN Microsystems Java und General Magics Telescript, um einerseits eine ma-

<sup>49</sup> vgl. Magedanz(1996), Hohl(1995), Maes(1995), Straßer(1996)

<sup>50</sup> Orte werden auch oft als Locations bezeichnet

<sup>51</sup> vgl. Straßer(1996)

schinenunabhängige Sprache zu haben, die auf verschiedenen Plattformen in heterogenen Netzen eingesetzt werden kann und andererseits gewisse Sicherheitsmechanismen bereitstellt, damit ein Agentensystem oder die sich darin bewegenden Agenten keinen Schaden anrichten können.

Eine der herausragenden Eigenschaften der Mobilien Agenten ist, wie der Name schon sagt, ihre Mobilität. Diese Mobilität kann auf zwei verschiedene Arten implementiert sein, durch „Remote Execution“ und durch „Migration“<sup>52</sup>.

**Remote Execution** ist ein Mechanismus, der einen Agenten (d.h. Programmcode und Daten) zu einem entfernten System transferiert, in welchem er aktiviert und ausgeführt wird. Der Transportmechanismus der Agenten kann von TCP/IP bis zur Elektronischen Post variieren.

Bei der **Migration** unterbricht der Agent seine Ausführung und „wandert“ von einem Ort zu einem anderen Ort. Technisch gesehen müssen dabei drei Zustände berücksichtigt werden:

- Programmcode bzw. Programmzustand
- Inhalt der Instanzvariablen (Datenzustand)
- Ausführungszustand

Alle drei müssen in ein Transportformat gebracht werden, welches eine unabhängige Zustandsbeschreibung für heterogene Plattformen bereithält; es muß außerdem einen Zugriff auf den Ausführungszustand möglich sein. Um dies zu gewährleisten, sind komplexe Operationen notwendig, da der Ausführungszustand vom jeweiligen Interpreter abhängt. Am Zielort nimmt der Agent an dem Punkt wieder seine Arbeit auf, an dem er sie unterbrochen hat. Vergleichbar ist dieser Ansatz mit einer Prozeßmigration, wie man sie z.B. in verteilten Systemen zur Lastbalancierung verwendet. Hier kommen allerdings noch Eigenschaften, wie Autonomie und Heterogenität hinzu, die bei einer Prozeßmigration nicht einfach zu realisieren sind. Die Migration unterscheidet sich fundamental vom herkömmlich verwendeten Ansatz des RPC (Remote Procedure Call) im Client/Server Bereich. Dieser Migrationsansatz erlaubt das Modellieren der Funktionalität von RPC und kann entscheidende Vorteile, durch die lokale Ausführung des Programmcodes auf dem Server und der damit möglicherweise verbundenen Einsparung von Datentransporten bieten, da weniger Fehlerbedingungen berücksichtigt werden

---

<sup>52</sup> vgl. Magedanz (1996), Straßer (1996)

müssen<sup>53</sup>. Bemerkenswert ist, daß an einem entfernten Ort ein neuer Code, der vorher nicht vorhanden war, generiert wird.

Um die Mobilität der Agenten bereitzustellen, müssen natürlich auch Ziele ausgemacht werden, zu denen ein Agent innerhalb eines Agentensystems migrieren kann. Diese Ziele werden im Allgemeinen Orte, „places“ oder „locations“ genannt.

Jeder **Ort** repräsentiert eine Ausführungsumgebung für Agenten, in der diese mit anderen Agenten kommunizieren, wie auch Ressourcen und Dienste des darunterliegenden Systems in Anspruch nehmen können. Die Ressourcen und Dienste werden innerhalb eines Agentensystems von Systemagenten bereitgestellt, die dann von anderen Agenten via Kommunikation angefordert werden können.

Eine der Basisfähigkeiten von Mobilten Agenten sollte die **Kommunikation** mit anderen Mobilten Agenten, Diensten und Benutzern sein. Hier kann man zunächst zwischen globaler<sup>54</sup> und lokaler Kommunikation unterscheiden. Zwei Mechanismen können dabei benutzt werden, die sich aber in Bezug auf die beiden Kommunikationsarten nicht unterscheiden:

- RPC (RMI)<sup>55</sup>: Agenten können hier mit anderen Agenten kommunizieren, indem sie eine Methode des anderen Agenten aufrufen.
- Message passing: ein datenorientierter Mechanismus, um Daten(Objekte) zwischen Prozessen auszutauschen. Dieser Mechanismus kann entweder asynchron oder synchron ablaufen

### 3.3.1 Vorteile und Nachteile von Mobilten Agenten<sup>56</sup>

In diesem Kapitel sollen die allgemeinen Vor- und Nachteile von Mobilten Agenten untersucht werden, mit dem Ziel, diese mit den Vor- und Nachteilen von Active Mail zu vergleichen.

---

<sup>53</sup> vgl. Harrison (1995)

<sup>54</sup> Eine globale Kommunikation muß nicht unbedingt gegeben sein, da sie auch durch eine Migration und die darauffolgende lokale Kommunikation ersetzt werden kann. Telescript von General Magic z.B. unterstützt nur die lokale Kommunikation.

<sup>55</sup> Remote Procedure Call (Remote Method Invocation)

<sup>56</sup> vgl. Harrison (1995), Hohl(1995), Straßer (1996)

**Vorteile von Mobilien Agenten:**

- **Mobile Agenten unterstützen „mobile computing“**  
Gerade für Mobile Geräte, wie Laptops, Notebooks oder PDAs, die gekennzeichnet sind durch schmale Bandbreiten, begrenzte Speicher- und Rechnerkapazität und eine temporäre Präsenz im Netz haben, eignen sich Mobile Agenten, denn sie unterstützen gerade diese Eigenschaften durch die asynchrone Verarbeitung.
- **Nachrichten können als Agenten aktiv sein**  
In Mobile-Agenten-Systeme können Nachrichten auch als Agenten programmiert sein. Sie haben den vollen Umfang der Agentenprogrammiersprache und bieten im Gegensatz zu den im wesentlichen statischen Nachrichten, die im Client/Server Bereich eingesetzt werden, einen großen Vorteil, da sie auch dynamisch auf Veränderungen (oder Events) der Umgebung reagieren können.
- **Kommunikation zwischen Teilnehmern (Agent oder Benutzer) kann asynchron stattfinden**  
Teilnehmer müssen nicht immer präsent sein, damit Agenten ihre Aufgabe erledigen können. Die Aufgaben oder die Nachricht können „zwischengelagert“ werden, bis einer der Teilnehmern (dies kann ein Benutzer, ein Server bzw. ein Dienstbringer oder einer anderer Agent sein) eine Interaktion wünscht. Somit muß ein Initiator einer Aktion nicht auf die Ergebnisse seines Auftrags warten und kann weiterarbeiten (auch inaktiv sein), bis das Ergebnis vorliegt.
- **Kommunikation kann lokal und nicht nur entfernt stattfinden**  
Aufgrund Migrationsfähigkeit der Agenten, kann eine Kommunikation zwischen weitverteilten Agenten bzw. Orten nicht nur über RPC geschehen, sondern auch lokal. Hierbei ist es möglich, Netzbandbreite zu sparen, wenn z.B. mehrere Anfragen an einen Server lokal bearbeitet werden und das Ergebnis einmalig zurückgeschickt wird.

**Nachteile von Mobilien Agenten:**

- **Sicherheitsaspekte müssen berücksichtigt werden**  
Ein fremder Agent muß sich beim Agentensystem authentifizieren, damit überprüft werden kann, daß er für gewisse Dienste (Belegung von Speicherkapazität, CPU-Zeit, u.a.) autorisiert ist. Seine Funktionen müssen überprüft werden, damit sie dem System keinen Schaden zufügen. Ebenso muß sichergestellt werden, daß der Agent keinem anderen Agenten Schaden zufügen kann

- **Kein Schutz vor „böswilligen“ Orten**

In einem offenen Agentensystem kann jeder Orte programmieren oder eröffnen, welche in das System integriert werden. Keiner kann jedoch garantieren, daß diese Orte auch sinnvoll, fehlerfrei und funktional sind. Im schlimmeren Fall könnten sogar Orte entstehen, die bewußt Agenten, die diesen Ort „besuchen“, zerstören, verändern<sup>57</sup> oder ihnen Informationen stehlen bzw. falsche Informationen zukommen lassen.

- **Funktionalität von Agenten läßt sich nur schwer abschätzen**

Schwierigkeiten entstehen bei der Programmierung von Agenten, da keine effektive Überprüfbarkeit der Funktionalität von Agenten gegeben ist. Unbeabsichtigte Fehler in der Programmierung können das Löschen von Daten in einem Agenten verursachen und direkt zu Nachteilen des Eigentümers führen, z.B. wenn der Agent wichtige Informationen über seinen Eigentümer oder elektronisches Geld mit sich führt.

- **keine zentrale Autorität, die „Regeln überwacht“**

Gerade vor dem Hintergrund einer weltweiten Verteilung eines Agentensystems wird es schwierig, sein eine zentrale Autorität zu installieren, die alle Betreiber eines Agentensystems hinsichtlich der Einhaltung der „Spielregeln“ überprüft.

### 3.3.2 Mobile Agentensysteme

Es existieren schon viele Projekte, die Mobile Agentensysteme erforschen bzw. implementieren. Erwähnt sein hier das Projekt **Ara**<sup>58</sup> (Agent for Remote Actions), das an der Universität Kaiserslautern seit August 1994 läuft, oder **Java-To-Go**<sup>59</sup>, ein Projekt an der Berkley Universität in Kalifornien, sowie **Telescript**<sup>60</sup> von General Magic und **Mole**<sup>61</sup>, um nur einige wenige zu nennen. Im folgenden soll jedoch Telescript kurz und Mole etwas ausführlicher erläutert werden.

#### 3.3.2.1 Telescript<sup>62</sup>

Im Telescript-System der Firma General Magic sind die Basiskonzepte mobiler Agenten zusammengestellt worden. Der kommerzielle Blickpunkt der Telescript-Technologie ist

---

<sup>57</sup> Es wäre denkbar, das ein Ort den Code des Agenten so manipuliert, daß er sein Verhalten ändert und unvorhersehbar wird.

<sup>58</sup> vgl. Peine (1996)

<sup>59</sup> vgl. Li (1996)

<sup>60</sup> vgl. General Magic (1996)

<sup>61</sup> vgl. Hohl(1996)

<sup>62</sup> vgl. White (1996), General Magic (1996)

der „electronic marketplace“, ein öffentliches Netzwerk, welches Anbieter und Konsumenten von Waren und Dienstleistungen zusammenführt und elektronische Geschäftsabschlüsse erlaubt. Der Elektronische Markt existiert noch nicht, aber die ersten Ansätze sind im Internet zu erkennen.

Innerhalb von Telescript werden Agenten, Orte, Migration, Treffen verschiedener Agenten, Kommunikation zwischen Orten, Vollmachten und Genehmigungen eines Agenten unterstützt. Das Netzwerk wird in Telescript als eine Sammlung von Orten gesehen, die Mobilen Agenten Dienste anbieten. Agenten werden als bewegliche Dienstbringer oder Dienstanbieter gesehen, die zwischen den Orten migrieren können und Dienste der Orte in Anspruch nehmen. Sie können auch mit bestimmten Vollmachten ausgestattet sein, um auf das umgebende System zuzugreifen oder sogar anderen Agenten einen Zugriff zu erlauben. Damit Agenten ihren Auftrag erfüllen können, können sie selbständig zu einem oder mehreren Orten migrieren, um dort ein Treffen mit einem anderen Agenten an einem Ort abzuhalten. Dadurch können die Agenten Informationen austauschen bzw. Dienste des anderen Agenten in Anspruch nehmen.

### 3.3.2.2 Das Agentensystem Mole<sup>63</sup>

Mole ist ein auf Java basierendes Agentensystem und liegt als Version Alpha 1.0 vor. Es wurde an der Universität Stuttgart in der Abteilung „Verteilte Systeme“ am Institut für Parallele und Verteilte Höchstleistungsrechner entwickelt. Das System dient als Forschungsprototyp für mobile Agenten und Agentensysteme. Die erste Implementierung des Agentensystems benutzt Java als Implementierungssprache und als Agentenentwicklungssprache.

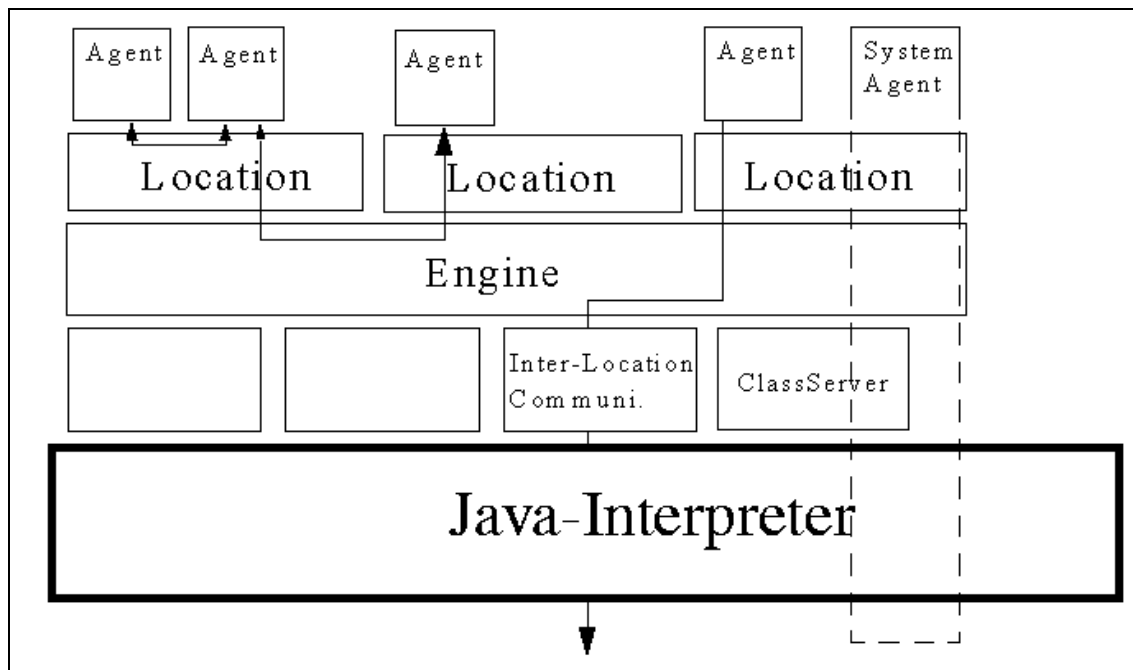
Im **Molesystem** existieren zwei Typen von Agenten, die ortsfesten **Systemagenten**, die eine Schnittstelle zum umgebenden System darstellen und die **Benutzeragenten**, die von Ort zu Ort migrieren können.

Orte werden in Mole auch als Locations bezeichnet. Die **Location** verwaltet die Agenten, die in der Location gestartet werden sollen. Sie startet ankommende Agenten und bietet grundlegende Dienste wie Migration, Kommunikation, „Yellow Pages“ und andere an. Der Location liegt eine **Engine** zugrunde, welche die Ausführung der Location verwaltet, dabei können mehrere Locations auf einer Engine laufen und die Engine läuft innerhalb eines Betriebssystemprozesses (siehe auch Abbildung 9). Der Vorteil dieser

---

<sup>63</sup> vgl. Hohl(1995), Hohl (1996), Straßer(1996)

Aufteilung ist, daß nur ein Java-Interpreter pro Engine benötigt wird und somit Performancegewinne resultieren.



**Abbildung 9: Architektur des Mole Systems**

Die **Migration** wird in Mole als eine etwas schwächere Form der in Kapitel 3.3 vorgestellten implementiert. Mole-Agenten migrieren nur den Code (die Klassen) und den Inhalt der Instanzvariablen, keine Threads oder temporäre Variablen. Nach dem Aufruf des Migrationswunsches, wird eine Stop-Methode aufgerufen, die jeder Agent implementieren kann. Nach der Migration wird am Zielort die Start-Methode des Agenten aufgerufen und der Agent kann seine Arbeit fortsetzen. Die Logik des Ausführungszustand der Mobilien Agenten muß somit manuell codiert werden. Dies macht die Migration sehr einfach<sup>64</sup>, aber die Programmierung eines Agenten wird erschwert.

Mole unterstützt zwei Mechanismen zur **Kommunikation**, RPC (RMI) und „Message Passing“. Der Agent kann selbst entscheiden, ob er lokal oder global kommunizieren will, abhängig von der Datengröße, die übertragen werden soll.

Ein Agent in Mole kann folgende Methoden implementieren.:

- `start`: wird durch das System nach einer Neuerzeugung oder Migration aufgerufen

<sup>64</sup> Es wird nicht notwendig, den Java-Interpreter umzuschreiben, damit er den Ausführungszustand speichert und kann damit auch auf allen Systemen, die einen Java-Interpreter bereitstellen, realisiert werden.

- `stop`: wird durch das System aufgerufen. Hier kann der Agent einige unnötige Daten löschen oder Threads beenden.
- `end`: wird aufgerufen, wenn der Agent gelöscht wird.
- `receiveMessage`: wird aufgerufen, wenn der Agent eine Nachricht erhält.
- `dispatch`: wird aufgerufen, wenn der Agent einen RPC-Aufruf erhält.
- `heartbeat`: wird auf Wunsch des Agenten periodisch vom System aufgerufen, damit der Agent periodische Arbeiten machen kann.

### 3.4 Vor- und Nachteile von Active Mail in Verbindung mit Mobilten Agenten

Folgende **Vorteile** bieten Mobile Agenten (MA) bei Einsatz als Active Mail:

- **MA sind schon aktiv**  
MA haben den Vorteil, daß sie schon aktiv sind, d.h. Programme ausführen können. Active Mails könnten diesen Vorteil nutzen, indem sie selbst als MA realisiert werden und somit Programmcode ausführen können.
- **Unterstützung von heterogenen Systemen**  
Heterogene Systeme werden von MA unterstützt. Active Mails würden automatisch im Rahmen dieser unterstützen Systemen arbeiten.
- **Sicherheit**  
MA können nur in einem Agentensystem agieren. Dieses Agentensystem sorgt für die Sicherheit der Agenten und des darunterliegenden Systems. Werden Active Mails als MA realisiert, gelten diese Sicherheitsmaßnahmen ebenso für Active Mails.
- **Autonomie**  
MA sind in der Lage eigenständig mit anderen Agenten oder Benutzern zusammenzuarbeiten bzw. zu kommunizieren. Bei Active Mail wird diese Fähigkeit erst durch den Benutzer aktiviert. Es könnten diese Fähigkeit der MA dazu benutzt werden, Active Mails noch leistungsfähiger zu machen (sie könnten z.B. andere Agenten erzeugen, die Aufgaben übernehmen).
- **Mobilität**  
MA sind mobil, d.h. es sind Mechanismen realisiert, die eine Mobilität des Agenten gewährleisten (via Migration oder E-Mail). Diese Mechanismen können automatisch von Active Mail benutzt werden, um einen Transport zu ermöglichen.

- **Erweitertes „mail handling“**

MA könnten als Briefträger fungieren, um Active Mails an einen oder mehrere Empfänger zu schicken. Wenn in einem Agentensystem, bei dem diese Funktionalität bereits realisiert ist, die Klasse des „Briefträger“-Agenten vorhanden ist, muß der Agent sich nur als *Instanz* der Klasse „identifizieren“. Es wird dann nicht notwendig, den Programmcode<sup>65</sup> des Agenten zu verschicken. Der Agent könnte außerdem mit dem Empfängeragenten, der die Mailbox des Empfängers verwaltet, in einen *Dialog* treten, um z.B. die Präferenzen<sup>66</sup> des Empfängers zu beachten. Der Vorteil, einen MA als Briefträger zu benutzen (anstatt z.B. SMTP), kann in der *Objektkapselung*<sup>67</sup> und in *Sicherheitsdiensten* (wie z.B. Authentifikation, Ablehnung, Anonymität, Bezahlung) liegen.

- **Kooperationsfähigkeit**

Kommunikation und Kooperation zwischen Teilnehmern (Agent oder Benutzer) ist für MA vorhanden. Diese Mechanismen könnten benutzt werden, um der Active Mail weitere Fähigkeiten zu geben, z.B. könnte eine Active Mail zusätzliche Dienste des Agentensystems (Yellow Pages) in Anspruch nehmen, mit anderen Agenten kommunizieren oder mit dem Absenderagenten in Dialog (um evtl. Kontrollinformationen auszutauschen) treten.

- **Einfacher Versionsabgleich bzw. Installation**

Ein *Versionsabgleich*, d.h. im Sinne der Installation aktueller Versionen des Agentensystems oder des Active Mail Systems, könnte in Interaktion mit dem Benutzer geschehen. Dabei kann der MA den Programmcode des ganzen Agentensystems oder nur einzelner Komponenten mit sich führen. Während der Ausführung könnte der MA dem Benutzer eine Hilfestellung geben und abfragen, welche Teile benötigt werden und letztendlich die gewünschten Komponenten installieren oder evtl. Teile bzw. Komponenten der alten Version des Agentensystems entfernen.

---

<sup>65</sup> In diesem Ansatz müßte nur noch der Datenzustand, d.h. der Inhalt der Nachricht übermittelt werden.

<sup>66</sup> Hier kommt man schon in den Bereich von Intelligenten Agenten, die Regeln bzw. Präferenzen definieren können, um dem Benutzer von zusätzlicher Arbeit zu befreien. Intelligente Agenten werden z.B. als Mail-Filter benutzt, um eingehende Mails automatisch einzusortieren oder Mails gleich abzuweisen. Diese Eigenschaften haben Mobile Agenten jedoch nicht und deswegen wird dieser Teilbereich nicht weiter verfolgt.

<sup>67</sup> D.h. das Objekt kann als eigenständige Einheit angesehen werden.

Folgende **Vorteile** bietet Active Mail gegenüber Mobilten Agenten:

- **Einfache Bedienung**

Die Funktionsweise der Benutzerschnittstelle eines Active Mail Systems gleicht herkömmlichen E-Mail Programmen. Da sehr viele Benutzer schon mit diesen E-Mail Programmen vertraut sind, wird neuen Benutzern die Verwendung eines Agentensystems erleichtert werden, wenn sie mit einem schon vertrautem Programm arbeiten können.

- **Einfache Erzeugung von aktiven Einheiten**

Das Konzept von vorgefertigten Anwendungen (Formularen<sup>68</sup>), ermöglicht neuen Benutzern eine einfache Generierung von Active Mail Agenten, welche nicht programmiert, sondern nur noch parametrisiert werden müssen. Dieses Konzept könnte auch auf andere Mobile Agenten übertragen werden, so daß der Benutzer diese auf einfache Weise generieren kann. Die Formulare der jeweiligen Agenten können in der normalen Agentensprache von Experten entwickelt und anderen Benutzern zur Verfügung gestellt werden.

Folgende **Nachteile** ergeben sich bei Einsatz von Active Mail in Verbindung mit Mobilten Agenten:

- **Spezielle Benutzerschnittstelle notwendig**

Ein spezielles Programm bzw. Benutzerschnittstelle wird benötigt, um die Erzeugung und Ausführung von Active Mails dem Benutzer auf einfache Weise zu ermöglichen. Sie bildet die Schnittstelle zwischen Agentensystem und Benutzer.

- **Sicherheit unterliegt dem Agentensystem**

Die Sicherheitsmechanismen für Authentifikation, Autorisierung und Ausführung der Active Mail Agenten bleibt dem Agentensystem überlassen. Wenn die Sicherheit im Agentensystem nicht gewährleistet wird, betrifft dies auch Active Mail Agenten.

- **Agentensystem zur Ausführung notwendig**

Active Mail funktioniert nur in der Umgebung eines Agentensystems. Damit sind Benutzer, die kein Agentensystem installiert haben, nicht erreichbar. Dieser Ansatz benötigt somit zusätzliche Komponenten.

---

<sup>68</sup> Dieses Konzept wird in dieser Arbeit noch vorgestellt.

- **Portabilität hängt von der Portabilität des Agentensystems ab**

Da die Active Mail nur in der Umgebung des Agentensystem funktioniert, ist sie von Sprache, die im Agentensystem verwendet wird bzw. in der Agenten programmiert werden, abhängig. Insofern hängt die Portabilität von Active Mail von der Portabilität der Sprache bzw. des Agentensystems ab.

Zusammenfassende Bewertung der wesentlichen Eigenschaften von Active Mail und Mobilten Agenten:

(Die Bewertung erfolgt in 4 Stufen (++) sehr gut, (+) gut, (-) weniger gut, (--) nicht ausreichend.)

<b>System</b>	Active Mail (AM)	Mobile Agenten (MA)	Bemerkungen
<b>Eigenschaften</b>			
Sicherheit	+	+	gleiche Eigenschaften werden gefordert
Unterstützung heterogener Systeme	+	+	
Mächtigkeit der Sprache	+	++	MA bieten meist vollen Funktionsumfang einer Sprache an
Programmausführung	ja	ja	
Lesbar mit normalem Mail-Reader	evtl.	evtl.	Spezieller Mechanismus wird benötigt („mailcap“-Datei)
einfache Erstellung	++	-	AM erlaubt einfache Erstellung von aktiven Einheiten
Erweiterbarkeit	+	++	MA sind flexibel erweiterbar
einfache Installation	+	+	AM und MA erlauben einfache Installation von Komponenten
Kooperationsfähigkeit	nein	ja	MA ermöglichen: „erweitertes mail handling“, Agent-Agent Kommunikation (Yellow Pages), evtl. synchrone Kommunikation mit anderen Teilnehmern
Autonomie	beschränkt	ja	MA können autonom handeln, AM nur durch Interaktion mit Benutzer
spezielle Benutzerschnittstelle	evtl. zur Erzeugung	ja	MA benötigen Schnittstelle zum Benutzer

**Tabelle 2: Gegenüberstellung der Systeme „Mobilten Agenten“ und „Active Mail“**

Die Gegenüberstellung macht deutlich, daß MA schon viele Eigenschaften der Active Mail beinhalten und darüber hinaus noch weitere Eigenschaften besitzen, mit der die Leistungsfähigkeit von Active Mails gesteigert werden kann, wenn eine Verbindung zwischen Mobilten Agenten und Active Mail realisiert wird. Auf der anderen Seite kann ein Active Mail System das Konzept der Mobilten Agenten ergänzen (einfache Bedienung und Erzeugung).

## 4 Spezifikation

Um die Komplexität des Softwaresystem zu verstehen und näher zu untersuchen, soll die Spezifikation zunächst beschreiben, **was** realisiert werden soll.<sup>69</sup>

### 4.1 Zielsetzung

Im Rahmen der Diplomarbeit soll ein Active Mail System „**ActiveMailMole**“ auf Basis von mobilen Agenten des Projektes Mole (siehe auch Kapitel 3.3.2.2), ein Forschungsprototyp der Abteilung Verteilte Systeme am Institut für Parallele und Verteilte Höchstleistungsrechner der Universität Stuttgart, entwickelt werden.

Mit diesem System soll es möglich sein, elektronische Briefe, die nicht nur statische Daten, sondern auch Programme enthalten, zu transportieren bzw. zu verschicken. Diese Briefe werden zu bestimmten Zeitpunkten ausgeführt und können wesentlich flexibler Inhalte darstellen und in Interaktion mit dem Leser treten.

Hierzu müssen die Anforderungen an ein Active Mail System analysiert und spezifiziert werden. Ebenso muß die Schnittstelle zum System Mole definiert werden, d.h. wie das Active Mail System in dieses Konzept eingebunden werden kann und wie sich das System die Funktionalität des bestehenden Systems nutzbar machen kann.

### 4.2 Allgemeine Beschreibung

Die Briefe bzw. „Active Mails“ sollen mit dem System Mole verschickt und verarbeitet werden. In diesem Zusammenhang muß geklärt werden, wie diese transportiert werden und welche *Werkzeuge* der Benutzer benötigt, um eine solche Botschaft zu lesen und zu erzeugen.

Die *einfache Erstellung* von Active Mails soll mit vorgefertigten Formbriefen unterstützt werden.

Eine *Kontrollkomponente* soll bereitgestellt werden, die eine Kontrolle über die verschickten Active Mails ermöglicht. Hierbei ist zu beachten, daß nur der Initiator einer Active Mail die Kontrolle über eine Active Mail behält. Die Funktion der Kontrollkomponente soll im wesentlichen beinhalten:

- „Terminierung einer schon abgeschickten Active Mail“

---

<sup>69</sup> vgl. Sommerville (1992)

- „Statusbericht der abgeschickten Active Mails“ (Active Mail hat Empfänger erreicht, Active Mail wurde gelesen, Active Mail ist terminiert, Active Mail hat Aufgabe erfüllt, z.B. im Falle eines Fragebogen).

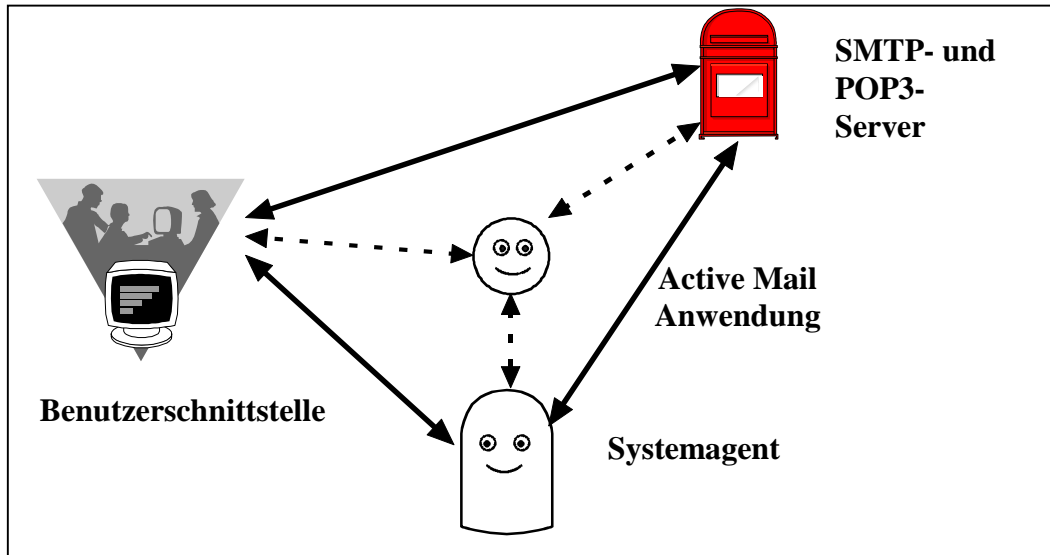
Zur Demonstration dient eine *Beispielanwendung*, die eine Vereinbarung eines Treffens zwischen mehreren Teilnehmern ermöglicht. Dabei soll der Absender mehrere Termine angeben können, aus denen die Empfänger auswählen können. Die Auswertung wird automatisch übernommen und alle Teilnehmer werden benachrichtigt, wenn ein Termin gefunden wurde oder nicht.

**Primäre Ziele sind somit:**

- Ein Active Mail System entwickeln, welches Active Mails lesen, verarbeiten und verschicken kann.  
Dabei ist zu beachten, daß die Active Mails gespeichert und gelöscht werden können.
- Die Erstellung einer Active Mail sollte auf einfache Weise möglich sein. Dabei können vorgefertigte Formulare oder Formbriefe benutzt werden, die nur noch parametrisiert werden müssen. Ebenso sollte es möglich, Formulare selbst zu programmieren und einfach in das System einzubinden, damit sie als Vorlage für Active Mails dienen können.
- Versenden einer Active Mail an eine, zwei oder mehrere Personen muß unterstützt werden.
- Das Active Mail System muß auf Basis von Mole realisiert werden, d.h. die Vorteile des Agentensystems sollen genutzt werden.
- Die Kontrolle über eine Active Mail muß beim Initiator liegen, damit er eine Active Mail terminieren kann (wenn dies für gewisse Active Mails Sinn macht) und auch eine Rückmeldung über den Status seiner verschickten Active Mails hat.
- Eine Demonstrationsanwendung (Terminvereinbarungsanwendung) ist bereitzustellen.

### 4.3 Informelle Beschreibung der Funktionalität von ActiveMailMole

Im folgenden wird die Funktionalität von ActiveMailMole beschrieben. Anhand dieser Beschreibung soll der Leser einen Eindruck über die Leistungsfähigkeit des Systems erhalten.



**Abbildung 10: Informelle Beschreibung der Architektur von ActiveMailMole**

Im wesentlichen existieren in dem zu entwickelten System vier Objekte:

1. Die **Benutzerschnittstelle**, die eine Verbindung zwischen System und Benutzer herstellt.
2. Der **Systemagent**, der benutzerspezifische Daten verwaltet.
3. **Active Mail Anwendungen**, die ausführbare Programme transportieren.
4. **SMTP- und POP3-Server**, Verschickung und Empfang von Active Mail Agenten bzw. normaler E-Mail. Dabei müssen die Protokolle SMTP und POP3 (siehe auch Kapitel 2.3.2) eingehalten werden.

#### 4.3.1 Aspekte der Benutzerschnittstelle

Dem Benutzer steht eine Benutzerschnittstelle zur Verfügung, die es ihm erlaubt, die Active Mails zu generieren, zu verwalten und zu verschicken. Sie ist das Werkzeug, welches die Verbindung von Benutzer und System darstellt. Dabei sollten dem Benutzer folgende Funktionen bzw. Komponenten zur Verfügung stehen:

- **Active Mail empfangen:** Der Benutzer hat die Möglichkeit über ein Menü seine Mailbox abzufragen, ob für ihn eine neue Active Mail vorhanden ist. Ist dies nicht der Fall, so erscheint ein Dialog mit „no new Active Mails“. Andernfalls wird ein Fenster mit der Inbox geöffnet und aus diesem ist ersichtlich, welche Mails noch nicht gelesen wurden (Status).
- **Inbox:** Der Benutzer kann über das Menü das Fenster Inbox mit den eingegangenen Active Mail öffnen und sieht welche Mails schon gelesen sind, und welche nicht. Beim Öffnen wird überprüft, ob neue Mails in der Mailbox liegen.
- **Outbox:** Der Benutzer kann über das Menü das Fenster Outbox öffnen und findet dort die Active Mail, die von ihm geschrieben und abgeschickt wurden. Die Outbox stellt zudem noch die *Kontrollkomponente* dar. Er kann hier schon abgeschickte Active Mails, die noch nicht wieder zurück sind, abrechen (terminieren). Zusätzliche Informationen über den Status der abgeschickten Active Mails (falls vorhanden) sollten übersichtlich angezeigt werden.
- **Löschen** einer Active Mail: die Active Mail wird in der Inbox oder in der Outbox aus dem persistenten Speicher entfernt.
- **Ausführen bzw. Anzeigen** einer Active Mail: Über die Inbox kann eine Active Mail ausgewählt und dann ausgeführt werden. Der Benutzer wird vor der Ausführung noch gefragt, ob er es will. Bevor die Active Mail ausgeführt wird, wird diese überprüft, ob sie dem System keinen Schaden zufügen kann.
- Eine neue Active Mail **senden:** Hier wird dem Benutzer eine Auswahl von vorgefertigten Active Mail-Formularen bzw. Active Mail Anwendungen angeboten, aus denen er auswählen muß (z.B. Terminvereinbarungsformular oder eine Multimedia-Präsentation). Im Formular müssen die Empfänger (kann auch eine Gruppe von Personen sein), der Titel (Subject) und die Active Mail-spezifischen Objekte ausgefüllt werden. Danach kann die Active Mail an die Empfänger geschickt werden. Das Ergebnis wird automatisch generiert und ausgewertet präsentiert, wenn alle Empfänger eine Antwort geschickt haben.
- **Setup:** Mittels eines Setups sollte es dem Benutzer ermöglicht werden, spezifische Benutzerdaten und Einstellungen des Systems einzugeben und zu verwalten. Diese Daten sollten persistent sein, d.h. sie sollten auch bei einem Neustart des Systems wieder vorhanden sein.
- **Hilfe:** über das Menü soll der Benutzer eine kurze Einführung in das Programm und eine Hilfestellung zur Funktionalität des Systems bekommen.

### 4.3.2 Aspekte des Systemagenten

In diesem Abschnitt soll erklärt werden, welche Anforderungen an einen Systemagenten (AMSystemAgent) von ActiveMailMole gestellt werden.

- Es muß sichergestellt sein, daß jeder Benutzer nur einen Systemagenten hat. Er stellt die Schnittstelle zwischen dem Agentensystem und der Benutzerschnittstelle dar.
- Der Systemagent speichert die benutzerspezifische Daten. Beim erstmaligen Start des Systems, muß der Benutzer diese Daten eingeben. Diese werden bei einem wiederholten Start nicht nochmals abgefragt, sollten aber zur Laufzeit wieder geändert werden können.
- Der Systemagent empfängt, speichert und verwaltet die eingehenden Active Mails des Benutzers. Er empfängt auch automatisch Active Mails, die mit normaler E-Mail verschickt wurden und für den Benutzer bestimmt sind (über das POP3-Account des Benutzers).
- Der Benutzer soll beim Aufrufen des Systems automatisch in Kenntnis gesetzt werden, ob für ihn neue Active Mails eingegangen sind.
- Die Verwaltung der Kontrollinformation von Active Mails, die vom Benutzer abgeschickt wurden, übernimmt der Systemagent. Ebenso terminiert er abgeschickte Active Mails auf Wunsch des Benutzers.
- Der Systemagent übernimmt auch die „Verhandlung“ mit eingehenden Active Mails und prüft, ob sie wirklich für den Empfänger bestimmt sind. Gegebenenfalls weist er Active Mails ab.
- Active Mails, die angenommen wurden, werden vom Systemagenten persistent gespeichert, d.h. sie sollen einen Ausfall des Systems oder des Arbeitsplatzrechners überstehen.
- Der Systemagent erzeugt Active Mails nach Auswahl eines Formulars bzw. einer Active Mail Anwendung.
- Ebenso aktiviert der Systemagent eingegangene Active Mails auf Wunsch des Benutzers.

### 4.3.3 Aspekte der „Active Mail Anwendungen“

- Active Mail Anwendungen sind Vorlagen für elektronischen Briefe (im weiteren als Active Mail Agenten bezeichnet), die Programme transportieren und sich auf Wunsch des Empfänger aktivieren können.
- Jede Active Mail Anwendung muß ein Formular bereithalten, mit dem sich Active Mail Agenten parametrisieren lassen (Empfängerliste, Betreff und anwendungsspezifische Daten).
- Die Active Mail Agenten übernehmen den Transport des ausführbaren Programms. Dabei können sie eigenständig entscheiden, ob sie migrieren oder sich als E-Mail verschicken.
- Active Mail Agenten müssen sich beim Systemagenten des Empfängers melden, um auf weitere Interaktionen des Benutzers zu warten.
- Bei der Empfängerinitiierten Ausführung der Active Mail Agenten, aktivieren sie das mitgeführte Programm.
- Dabei ist zu beachten, daß sie sich Dienste des Agentensystems zunutze machen können, um spezifische Aufgaben der Anwendung zu erfüllen (z.B. Yellow Pages).
- Active Mail Agenten müssen jederzeit inaktiviert werden können, falls der Empfänger die Ausführung des Agenten unterbrechen will.
- Sie übernehmen die Auswertung der Interaktion mit dem Empfänger, falls eine Auswertung vorgesehen ist.
- Sie können Daten über ihren aktuellen Status an den Initiator senden, solange sie nicht beim Empfänger angekommen sind. Wenn sie beim Empfänger angenommen wurden, müssen sie die Einstellungen des Empfängers berücksichtigen (z.B. ob dieser einen Informationsaustausch erlaubt).
- Bereits abgeschickte Active Mail Agenten sollten auch noch vom Absender terminiert werden können, solange sie keine Einstellungen des Empfängers beeinträchtigen.
- Falls ein Fehler bei Verschickung, Empfang, Aktivierung, Auswertung oder Terminierung auftritt, soll ein detaillierter Bericht an den jeweiligen betroffenen Benutzer (Absender oder Empfänger) geschickt werden.

#### **4.4 weitere Anforderungen**

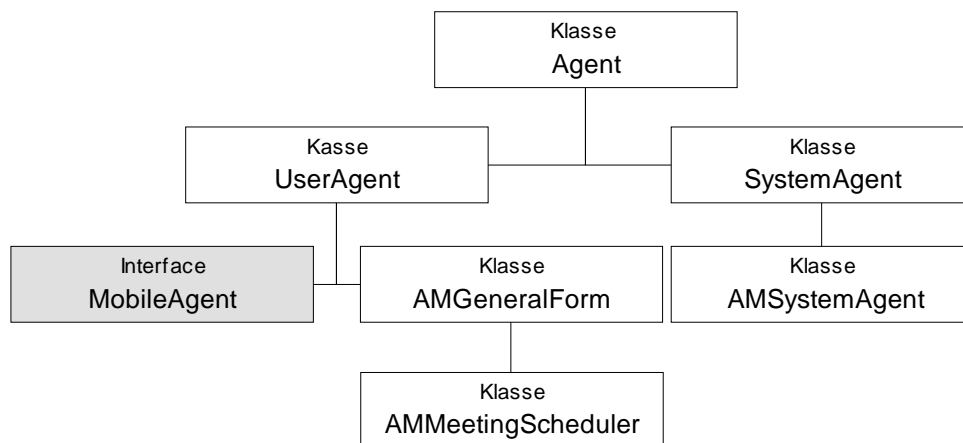
Weitere Anforderungen an ein Active Mail System wurden schon in Kapitel 2.5.5 aufgestellt und sollten auch in diesem System berücksichtigt werden. An dieser Stelle werden sie jedoch nicht nochmals erwähnt.

## 5 Entwurf und Implementierung

Der Entwurf schafft die Struktur und damit die Architektur des Software-Systems, wobei das System in einzelne handhabbare Komponenten zerlegt wird, um die Komplexität zu reduzieren.<sup>70</sup> Da die Sprache Java, in der das Agentensystem Mole geschrieben ist, benutzt werden soll und Java objektorientiert ist, muß das System auf Objektstrukturen bzw. Klassen abgebildet werden. In diesem Kapitel werden die für das System notwendigen Klassen isoliert und der jeweils spezielle Aufgaben zugewiesen.

### 5.1 Die Active Mail in ActiveMailMole

Die Active Mail wird als Mobiler Agent des Agentensystems Mole entworfen. Dadurch kann die Active Mail schon existierende Mechanismen des Agentensystems (z.B. Migration und Kommunikation mit anderen Agenten) verwenden; eine Active Mail muß dabei von der Klasse UserAgent abgeleitet sein und das Interface MobileAgent implementieren. Eine Active Mail in Mole benötigt ferner weitere Mechanismen, um die Funktionalität einer Active Mail zu erreichen. Damit nicht für jede Active Mail diese Mechanismen neu geschrieben werden müssen, wird eine generelle Klasse AMGeneralForm eingeführt. Jede Active Mail bzw. Active Mail Anwendung (z.B. die Terminvereinbarung AMMeetingScheduler, siehe auch Kapitel 5.4) muß von dieser Klasse abgeleitet sein, um die neuen Mechanismen benutzen zu können. Abbildung 11 veranschaulicht diesen Zusammenhang.



**Abbildung 11: ActiveMailMole Klassendiagramm**

<sup>70</sup> vgl. Sommerville (1992)



fänger eingegeben wurden [2]. Die AMAgenten migrieren oder verschicken sich per MIME-E-Mail automatisch zum entsprechenden Empfänger (Location oder E-Mail Adresse des Empfängers) [3]. An der Location D des Empfängers angekommen<sup>72</sup> [4], versucht der AMAgent sich beim AMSystemAgenten des Empfängers bzw. Benutzers D zu melden. Ist diese Anmeldung erfolgreich, nimmt der AMSystemAgent des Benutzers D den AMAgenten, speichert ihn persistent [5] und meldet dem Benutzer D, daß eine neue Active Mail eingegangen ist. Sobald der Benutzer D den Wunsch hat, diesen AMAgenten auszuführen (Open), wird der AMAgent vom Systemagenten aktiviert. Der AMAgent führt dann sein Programm aus (Hello) [6] und führt gegebenenfalls eine Interaktion mit dem Benutzer D durch. Hierbei wird standardmäßig der Header des AMAgenten und eine SUSPEND-Schaltfläche angezeigt, damit der Benutzer jederzeit den AMAgenten unterbrechen kann. Weitere Schaltflächen, grafische Elemente oder Programmteile, die stark differieren können, bleiben der Anwendung der Active Mail überlassen (siehe auch Kapitel 5.4). Folgezustände, die z.B. aus der Interaktion mit dem Empfänger resultieren, sind in dieser Abbildung aus übersichtlichkeitsgründen nicht mehr beschrieben; sie werden in den folgenden Kapiteln näher bestimmt.

Zunächst soll der generelle Aufbau eines AMAgenten, definiert durch die Klasse AMGeneralForm, beschrieben werden.

### 5.1.1 Die Klasse AMGeneralForm

Die Klasse AMGeneralForm beschreibt den allgemeinen Aufbau für Active Mail Anwendungen in Mole und realisiert Mechanismen, die Erzeugung, Versand, Ankunft, Annahme, Aktivierung und Terminierung von AMAgenten automatisch übernehmen.

Zunächst wird auf die allgemeinen Attribute eingegangen, die eine Active Mail Anwendung benötigt, danach die Methoden, die in der Klasse AMGeneralForm realisiert sind. Durch die Vererbung an Unterklassen wird es möglich, die Attribute und Methoden gemeinsam nutzen zu können.

#### 5.1.1.1 Attribute

Jeder AMAgent hat Attribute, die den „Lebenslauf“ des Agenten entscheidend beeinflussen. Weitere Attribute können je nach Anwendung der Active Mail, die eine Unter-

---

<sup>72</sup> Dies kann entweder durch eine Migration (von Location A nach Location D) erfolgen oder der Systemagent eines Benutzers (in der Abbildung der Systemagent des Benutzers B) holt den AMAgenten vom POP3-Server des Benutzers und lädt ihn auf die Location B.

klasse von AMGeneralForm sein muß, hinzugefügt werden (Datenfelder, Bezeichner, Interaktionsflächen, ...).

### **Die Benutzeradresse (userAddress)**

Damit der AMAgent weiß, für wen er bestimmt ist, braucht er Informationen über die Adresse des Benutzers. Da im Entwurf vorgesehen ist, daß ein AMAgent entweder migrieren kann oder sich als MIME-E-Mail verschickt, müssen zwei verschiedene Adressen des Benutzers dem Agenten vorliegen: zum einen die Location-Adresse des Benutzersystemagenten und zum anderen die E-Mail Adresse des Empfängers.

Da man davon ausgehen kann, daß jeder Benutzer eine E-Mail Adresse hat, aber nicht unbedingt eine ständige Location<sup>73</sup>, muß die E-Mail Adresse des Benutzers dem AMAgenten unbedingt vorliegen. Die Location, auf der sich sein Systemagent befindet, kann optional angegeben werden. Die Adresse eines Benutzers in ActiveMailMole muß somit die Form

„EmailAddressOfUser[@LocationName]“<sup>74</sup>

einhalten. Diese Modellierung bringt einen weiteren Vorteil mit sich, denn der Absender muß nicht unbedingt die Agentennamen<sup>75</sup> der Systemagenten der Empfänger kennen, sondern braucht nur die E-Mail-Adresse der Empfänger anzugeben. Damit kann der AMAgent den Empfänger eindeutig identifizieren und erreichen (näheres siehe Kapitel 5.2).

### **Der Header (header)**

Der Header eines AMAgenten ähnelt dem Header einer normalen E-Mail (siehe auch Kapitel 2.2). Er beinhaltet eine Liste der Empfänger (Benutzeradressen), ein Betreff-Feld, den Absender (Benutzeradresse) und das Datum der Erzeugung. Er wird bei der Verschickung und zur Anzeige des AMAgenten in der Inbox oder Outbox bzw. beim Ausführen des AMAgenten verwendet.

### **Lebensdauer eines AMAgenten (expirationTime)**

Die Lebensdauer eines AMAgenten gibt dem Agenten die Information mit, bis wann (Datum) seine Ausführung möglich ist. Wird dieses Datum überschritten, wird der AMAgent nicht mehr ausgeführt oder er kann sich selbst terminieren bzw. löschen. Ist

---

<sup>73</sup> Der Benutzer könnte z.B. die Location als Aufenthaltsort seines AMSystemAgenten wechseln.

<sup>74</sup> z.B. „nitschjn@trick.informatik.uni-stuttgart.de@boston.mole.informatik.uni-stuttgart.de“

<sup>75</sup> Die 64-bit ID von MOLE

kein Datum angegeben, so kann der AMAgent theoretisch für immer existieren und ausführbar sein. Die Eigenschaft, die Lebensdauer eines AMAgenten zu beschränken, macht natürlich nicht für jede Anwendung Sinn. Bei einer Multimedia-Präsentation beispielsweise, die eine kleine Animation mit Bild und Ton enthält, kann es sogar wünschenswert sein, sie immer wieder abspielen zu können. Dagegen ist bei gewissen Anwendungen, z.B. der Terminvereinbarung oder einer Buchbestellung, ist eine Ausführung des AMAgenten nur bis zu einem definierten Zeitpunkt sinnvoll. Danach würde der Empfänger nur mit einer unnötigen Active Mail belästigt bzw. der Absender könnte noch Antworten (Bestellungen) bekommen, wenn diese gar nicht mehr gebraucht werden.

### **Status der Active Mail (AMStatus)**

Der Status beschreibt den aktuellen Zustand des AMAgenten in seinem Lebenslauf. Der Agent kann folgende Zustände annehmen: CREATED, SENDED, ARRIVED, RECEIVED, ACTIVE, READ, SUSPENDED, REPLY, TERMINATED, sowie einige Fehlerzustände: ER\_CREATION, ER\_SENDED, ER\_ARRIVED, ER\_RECEIVED, ER\_ACTIVE, ER\_READ, ER\_SUSPENDED, ER\_REPLY, ER\_TERMINATED. Sie werden im dynamischen Modell (siehe auch Kapitel 5.1.2) näher beschrieben.

Dieses Attribut wird auch dazu verwendet, um den Status<sup>76</sup> des AMAgenten entweder auf Anfrage oder automatisch an den Absender bzw. den Empfänger zu übermitteln.

#### **5.1.1.2 Methoden**

Methoden sind die Operationen der Klasse, die von den Instanzen gemeinsam genutzt werden können. Für Versand, Ankunft, Annahme, Unterbrechung und Terminierung einer Active Mail werden Methoden verwendet, die im dynamischen Modell durch Zustandsänderungen (siehe weiter unten) näher beschrieben werden. Dabei werden nicht unbedingt die Methoden beschrieben, sondern eher ihr Verhalten im dynamischen Ablauf.

Methoden, zur Erzeugung (executeOnCreation), Aktivierung (executeOnActivation) und zur Auswertung (executeOnReply) stellen die **Schnittstelle** für die Anwendungen dar, die von AMGeneralForm abgeleitet sind. Weitere Methoden können noch hinzugefügt, aber erst nach einer Benutzerinteraktion aufgerufen werden. Sie könnten z.B. mit weiteren Schaltflächen oder durch die Interaktion mit dem Benutzer (z.B. Eingabe eines falschen Wertes) aktiviert werden. Prinzipiell sind hier keine Grenzen für die Erweite-

---

<sup>76</sup> ggf. mit einer detaillierten Fehlermeldung, warum ein Fehler aufgetreten ist.

nung gesetzt. Nebenbei können auch andere Dienste des Agentensystems (Yellow Pages, ...) benutzt werden, um die Funktionalität der Active Mail Anwendung zu erweitern. Jedoch muß beachtet werden, daß die Sicherheitsmaßnahmen und Beschränkungen durch das Agentensystem auch für AMAgenten gelten.

### **executeOnCreation**

Diese Methode wird aufgerufen nachdem ein AMAgent instanziiert wurde. Hier kann jede Active Mail Anwendung eine eigene Eingabemaske zur Initialisierung des AMAgenten definieren. Es werden standardmäßig der Header, ein Eingabefeld für die Empfänger, ein Betreffeld und Schaltflächen für das Senden und Abrechnen des AMAgenten angezeigt.

### **executeOnActivation**

Diese Methode wird aufgerufen, wenn der Empfänger ausdrücklich die Aktivierung des AMAgenten wünscht. Neben dem Header (From, To, Subject) wird eine Schaltfläche SUSPEND angezeigt, die den AMAgenten jederzeit in seiner Ausführung unterbricht. Weitere Elemente bleiben der Active Mail Anwendung überlassen. Hier könnten z.B. Elemente wie eine REPLY- oder einer FORWARD-Schaltfläche angezeigt werden, aber auch andere Elemente (Schaltflächen, Textfelder, Listen, ...).

### **executeOnReply**

Diese Methode stellt die Auswertung einer Benutzerinteraktion dar, wenn der Benutzer dem Absender antworten will (drücken der REPLY-Schaltfläche). Sie kann keine allgemeine Funktion übernehmen, die für jede Active Mail Anwendung gültig ist, sondern muß ebenso wie die beiden oben genannten Methoden von der Anwendung realisiert sein. Nachdem der Benutzer die REPLY-Schaltfläche gedrückt hat, kann der Absender oder ggf. andere Empfänger auf unterschiedliche Weise informiert werden:

- *Auswertung der Eingaben wird vom AMAgenten übernommen* und ggf. wird dem Empfänger bei Falscheingaben eine Fehlermeldung ausgegeben. Danach schickt der AMAgent eine Nachricht (z.B. als normale E-Mail) an den Absender oder an alle Empfänger.
- *Auswertung der Eingaben wird vom AMAgenten übernommen* und ein *aktueller Stand* wird dem Empfänger angezeigt (z.B. bei einem Fragebogen, die Anzahl der richtigen Antworten). Danach könnte der Absender vom Ergebnis des Empfängers unterrichtet werden.

- *Auswertung der Eingaben wird beim Absender übernommen.* Hier könnte z.B. ein weiterer Agent beim Absender mit Informationen „gefüttert“ werden, der dann entscheidet, wann er den Absender oder die Empfänger über ein Ergebnis benachrichtigt (Dies könnte ebenfalls mit einer anderen Active Mail geschehen oder einfach nur per normaler E-Mail).

Weitere Auswertungsarten sind denkbar, hier wurden nur die wichtigsten genannt.

## **Nachrichtenabarbeitung**

In Mole gibt es zwei verschiedene Möglichkeiten für eine Kommunikation zwischen den Agenten: Nachrichten (Messages) und Remote Procedure Calls (RPC). Da ein AMAgent asynchrone Eigenschaft hat, muß eine Kommunikation ebenfalls asynchron geschehen. Deswegen werden nur Nachrichten für den AMAgenten beachtet. Der AMAgent muß auf bestimmte Nachrichten, die an ihn gerichtet sind, reagieren und dann entsprechend handeln. Dabei werden vier unterschiedliche Nachrichtentypen unterschieden:

- **giveHeaderInformation-Nachricht**  
Diese Nachricht wird vom AMSystemAgenten des Empfängers an den AMAgenten gesendet, um den Header (Absender, Empfängerliste, Betreff und Datum der Erzeugung) des AMAgenten zu erhalten. Der AMAgent antwortet dem AMSystemAgenten des Empfängers (und nur ihm), indem er ihm die gewünschten Informationen ebenfalls mit einer Nachricht zukommen läßt.
- **giveAMStatus-Nachricht**  
Mit dieser Nachricht wird der aktuelle Status des Agenten abgefragt. Diese Nachricht kann vom AMSystemAgenten des Absenders oder des Empfängers kommen. Der AMAgent entscheidet aufgrund der Einstellungen seines Empfängers, ob er diese Daten übermitteln darf (siehe auch weiter unten).
- **terminate-Nachricht**  
Diese Nachricht zeigt dem AMAgenten an, daß er terminieren soll. Sie darf nur vom Absender des AMAgenten stammen. Wenn der AMAgent schon beim Empfänger angenommen wurde, muß er überprüfen, ob er automatisch terminieren darf (der Empfänger kann die automatische Terminierung unterbinden, siehe auch weiter unten). Andere Terminierungen des AMAgenten werden im dynamischen Modell beschrieben.
- **giveExpirationTime-Nachricht**  
Bei dieser Nachricht gibt der AMAgent seine Lebensdauer zurück.

- **setPreferences-Nachricht**

Diese Nachricht bekommt der AMAgent nach Annahme beim Empfänger, um die Einstellungen des Empfängers beachten zu können (automatische Terminierung und Informationsweiterleitung).

Auch bei der Nachrichtenabarbeitung bleibt es der Active Mail Anwendung überlassen, ob sie bzw. der AMAgent, auf weitere Nachrichten reagiert, z.B. könnten die unterschiedlichen AMAgenten sich gegenseitig Nachrichten zukommen lassen, um einen aktuellen Stand der Auswertung eines Fragebogens zu haben (z.B. 5 von 10 haben bei Frage x mit Ja geantwortet).

### 5.1.2 Das dynamische Modell

Die Aspekte eines Systems, die mit Zeit und Veränderungen zu tun haben, bilden das **dynamische Modell**. Die wichtigsten Konzepte der dynamischen Modellierung sind Ereignisse, die externe Reize repräsentieren, und Zustände, die Werte von Objekten repräsentieren. Das Geflecht der Ereignisse, Zustände und Zustandsübergängen kann abstrahiert und als Zustandsdiagramm dargestellt werden.<sup>77</sup>

Um ein Zustandsdiagramm für einen AMAgenten bzw. für die AMGeneralForm Klasse zu beschreiben, müssen erst die Ereignisse identifiziert werden, die den AMAgenten betreffen.

#### **Benutzerereignisse des Absenders, die einen AMAgenten betreffen**

- Der Benutzer erzeugt neuen AMAgenten (**NEW**).
- Der Benutzer sendet den AMAgenten (**SEND**).
- Der Benutzer bricht die Erzeugung bzw. die Initialisierung des AMAgenten ab (**CANCEL**).
- Der Benutzer deaktiviert den aktiven Teil seiner abgeschickten AMAgenten (**TERMINATE**).

#### **Benutzerereignisse des Empfängers, die einen AMAgenten betreffen**

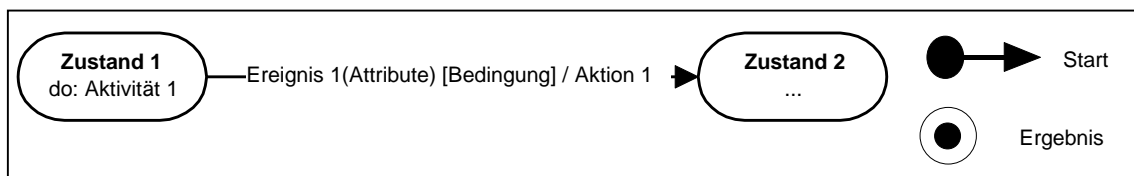
- Der Benutzer öffnet bzw. aktiviert den erhaltenen AMAgenten (**OPEN**).
- Der Benutzer unterbricht die Ausführung des AMAgenten (**SUSPEND**).

---

<sup>77</sup> vgl. Rumbaugh (1993)

- Der Benutzer löscht den erhaltenen AMAgenten (**DELETE**).
- Der Benutzer hat auf den AMAgenten geantwortet (**REPLY**). Dieses Ereignis muß nicht unbedingt auftreten; es ist abhängig von der Anwendung der Active Mail, ob eine Antwort erwünscht ist (spezielle REPLY Schaltfläche in der Anwendung, siehe oben).

Ein AMAgent kann unterschiedliche Zustände annehmen. Diese hängen von den Attributwerten des AMAgenten ab und der Ereignisabfolge, die den AMAgenten betreffen. Um ein Zustandsdiagramm für einen AMAgenten zu beschreiben wird die Notation von OMT<sup>78</sup> (Object Modeling Technique) verwendet. Zunächst eine kurze Erläuterung der Notation:

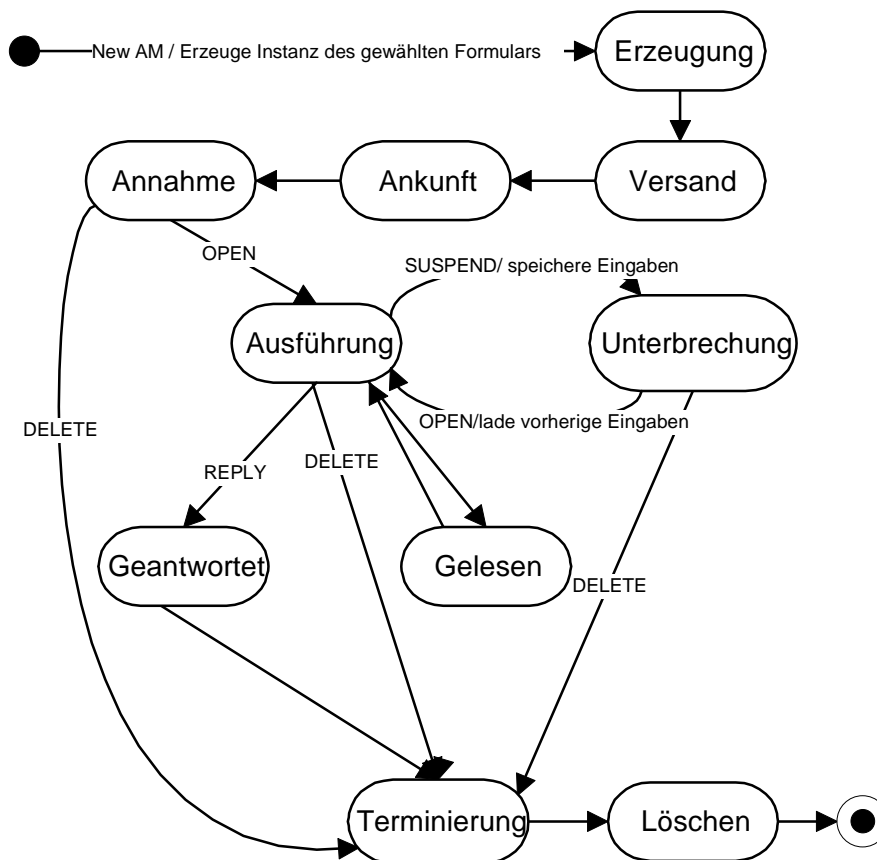


**Abbildung 13: OMT Notation für Zustandsdiagramme**

Wie Abbildung 13 zeigt, steht der Zustandsname fett in einem abgerundeten Rechteck. Eine Aktivität wird in einer Zustandsbox durch das Schlüsselwort „do:“ angezeigt, gefolgt vom Namen oder der Beschreibung der Aktivität. Der Ereignisname wird über einen Transitions Pfeil geschrieben. Bedingungen können in den eckigen Klammern angegeben werden, unter welcher ein Zustandsübergang erfolgt. Die Aktion ist eine auf den Moment beschränkte Operation ohne Zeitdauer und kann mit einem Ereignis verknüpft werden.

<sup>78</sup> vgl. Rumbaugh (1993)

Folgende Abbildung 14 zeigt das generelle Zustandsdiagramm eines AMAgenten.



**Abbildung 14: Zustandsdiagramm eines AMAgenten**

Transitionen bzw. Zustandsübergänge ohne Beschriftung bedeuten, daß der AMAgent automatisch in den nächsten Zustand übergeht, wenn der vorherige Zustand erreicht wurde. Im Diagramm sind aus übersichtlichkeitsgründen nur die wichtigsten Zustandsübergänge enthalten. Es fehlt z.B. die Transition von der Erzeugung zur Terminierung, wenn die Erzeugung eines AMAgent abgebrochen wurde. Diese Übergänge werden in den einzelnen nachfolgenden Unterzustandsdiagrammen behandelt. Diese Unterzustandsdiagramme beschreiben die in Abbildung 14 dargestellten generellen Zustände im einzelnen:

## ERZEUGUNG

Wurde der AMAgent vom Systemagenten instanziiert, führt er die Methode executeOnCreation aus, die jede Active Mail Anwendung neu schreiben kann. Dies erlaubt eine anwendungsspezifische Darstellung zur Parametrisierung der AMAgenten (Eingabe Felder für Texte, usw.). Wird die Erzeugung abgebrochen, so löst der Agent ein Terminierungsereignis mit einer Fehlermeldung aus, damit er zu einem entsprechenden Ende kommt (siehe auch Terminierung).

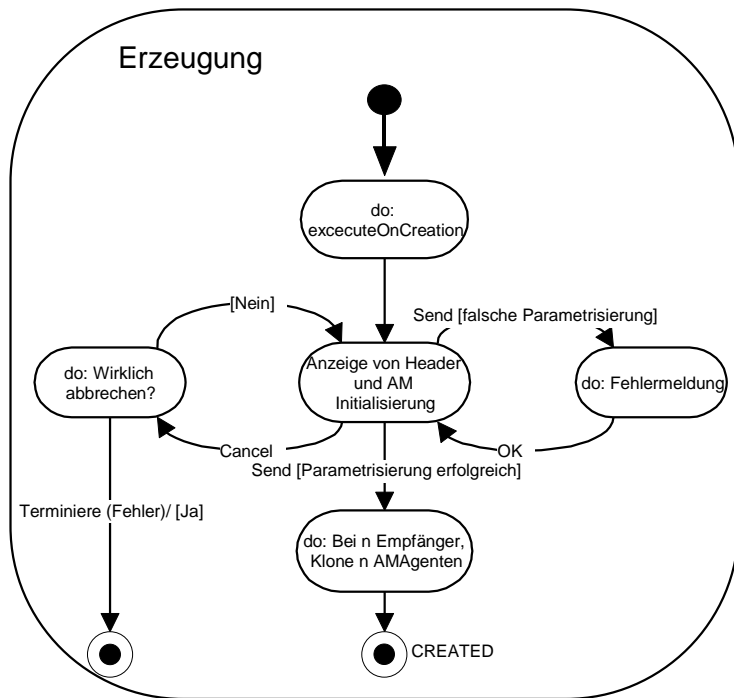


Abbildung 15: Erzeugungszustand

## VERSAND

Je nach Benutzeradresse, die der AMAgent als Empfänger hat, kann er entscheiden, ob er migriert (schnellere Variante) oder sich als MIME-E-Mail verschickt. Hierbei muß vom Agentensystem die Möglichkeit gegeben sein, daß sich Agenten als MIME-E-Mail verschicken können. Der Vorteil, daß sich Agenten mittels E-Mail verschicken können, besteht vor allem darin, daß der Benutzer über seine normale E-Mail Adresse erreichbar bleibt, auch wenn die Location des Empfängers momentan nicht gestartet ist; zum anderen könnte eine Location-Location Verbindung nicht zu Stande kommen, da der Transport der Agenten über eine TCP/IP Verbindung geschieht und manche Rechnersysteme über eine Firewall verfügen, die eine solche Verbindung ausschließen.

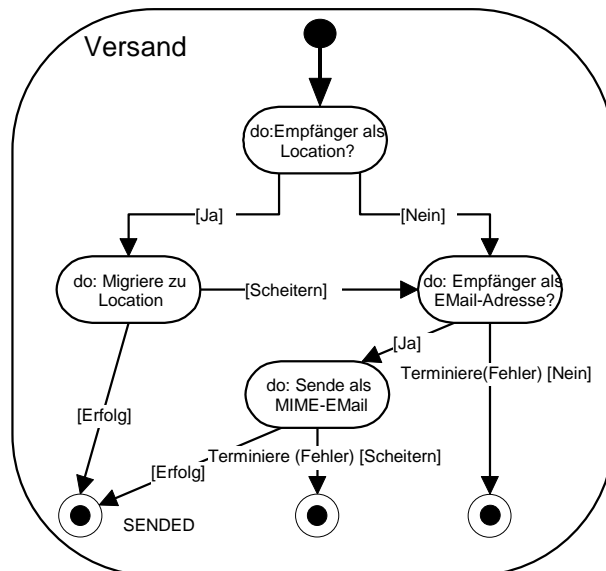


Abbildung 16: Versandzustand

## ANKUNFT

Nachdem der AMAgent auf einer Location angekommen ist (Migration oder MIME-E-Mail), prüft er, ob die vorher definierte Location erreicht wurde und der Empfänger hier seinen AMSystemAgent hat. Ist dies der Fall, dann ist der AMAgent am Zielort angekommen (ARRIVED). Ist dies jedoch nicht der Fall (z.B. wurde der LocationName des Empfängers nicht angegeben, oder Benutzer ist „umgezogen“ auf eine andere Location), überprüft der AMAgent, ob nicht doch der AMSystemAgent des Empfängers vorhanden ist. Wenn der AMSystemAgent vorhanden ist, so ist der AMAgent an der richtigen Adresse und aktualisiert in seinem Header den Namen der Location des Empfängers. An dieser Stelle könnte aber auch ein anderer Agent dem AMAgenten mitteilen, daß der Empfänger „umgezogen“ ist und ihm die neue Adresse mitteilen; in diesem Falle versendet sich der AMAgent mit der Adresse wieder neu (siehe oben). Schlägt alles fehl, terminiert der AMAgent.

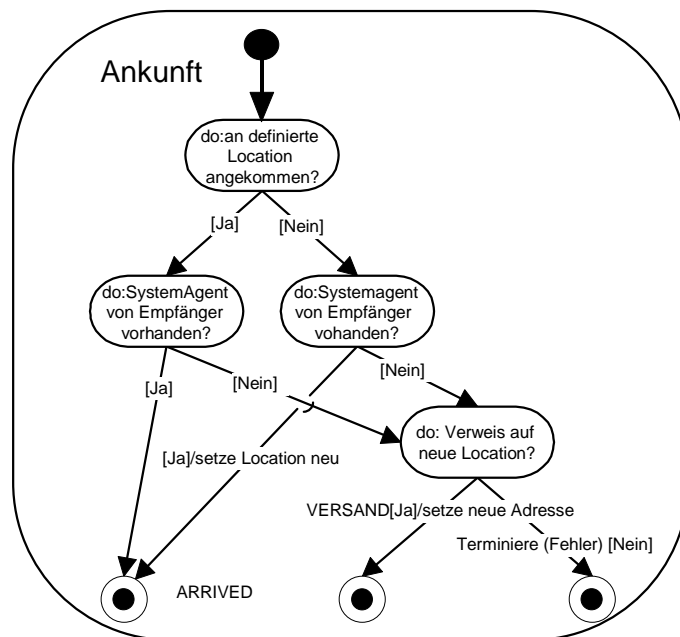


Abbildung 17: Ankunftsstatus

Wenn der AMSystemAgent des Empfängers vorhanden ist. Wenn der AMSystemAgent vorhanden ist, so ist der AMAgent an der richtigen Adresse und aktualisiert in seinem Header den Namen der Location des Empfängers. An dieser Stelle könnte aber auch ein anderer Agent dem AMAgenten mitteilen, daß der Empfänger „umgezogen“ ist und ihm die neue Adresse mitteilen; in diesem Falle versendet sich der AMAgent mit der Adresse wieder neu (siehe oben). Schlägt alles fehl, terminiert der AMAgent.

## ANNAHME

Der AMAgent gilt als angenommen (RECEIVED), sobald er sich beim AMSystemAgenten meldet und dieser ihn als neue Active Mail für seinen Benutzer akzeptiert. Dabei nimmt der AMSystemAgenten den AMAgenten, speichert ihn persistent und benachrichtigt den Empfänger, daß eine neue Active Mail eingegangen ist. Bevor der AMSystemAgent den AMAgenten akzeptiert, könnten Authentifizierungsmechanismen greifen, die sicherstellen, daß der Absender auch wirklich der ist, der

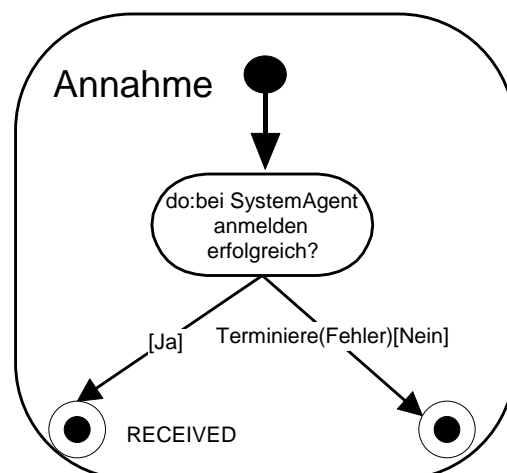


Abbildung 18: Annahmezustand

er vorgibt zu sein. Die Technologie von Privacy Enhanced Mail<sup>79</sup> (PEM) oder Pretty Good Privacy<sup>80</sup> (PGP) könnte hier eingesetzt werden, um eine Authentifizierung sicherzustellen. In dieser Diplomarbeit werden diese Mechanismen nur am Rande erwähnt, da diese auch ein Teil der Sicherheitsmaßnahmen des Agentensystems sein sollten. Nach der Annahme muß der AMAgent die Einstellungen des Empfängers beachten (z.B. „darf nicht automatisch Terminieren“).

## AUSFÜHRUNG

Ein AMAgent wird nur dann ausgeführt, wenn der Benutzer dies eindeutig will (Ereignis OPEN durch den Benutzer ausgelöst) und der AMAgent noch gültig ist. Ist dies alles gegeben, führt der AMAgent seinen mitgeführten Programmteil aus (executeOnActivation) und ist aktiv (ACTIVE). Der Programmteil, kann wie schon erwähnt, von Anwendung zu Anwendung sehr unterschiedlich sein. Nur der Header

und eine SUSPEND-Schaltfläche werden standardmäßig angezeigt. Der Agent ist vom Systemagenten völlig unabhängig und übernimmt die eigene Kontrolle. Die Sicherheitsmechanismen des Agentensystems greifen hier automatisch, damit der Agent nichts unerlaubtes macht (Vollschreiben der Festplatte, Zugriff auf unautorisierte Daten, ...).

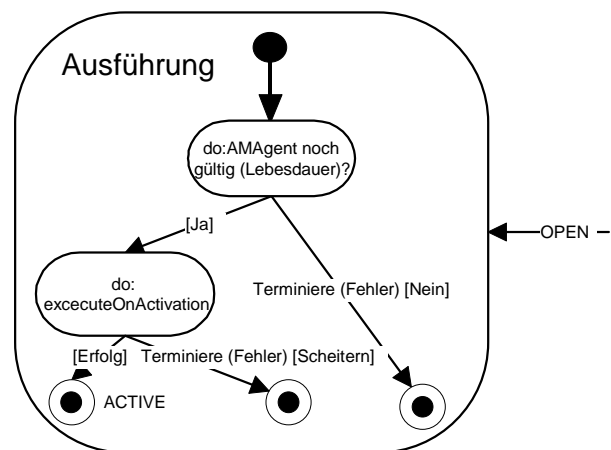


Abbildung 19: Ausführungszustand

## UNTERBRECHUNG

Sobald der Benutzer während der Ausführung des AMAgenten die SUSPEND-Schaltfläche drückt (Ereignis SUSPEND), welche jede Active Mail Anwendung neben der Anzeige des Headers durch Vererbung erhält, wird der AMAgent unterbrochen und sichert die bisher eingegebenen Daten und läßt sich vom AMSystemAgenten persistent speichern(SUSPENDED). Danach ist eine nochmalige Aktivierung des AMAgenten auf Wunsch des Benutzers (Ereignis OPEN) jederzeit möglich. Der AMAgent lädt seine bisher eingegebenen Daten und kommt in den Ausführungszustand (siehe oben).

<sup>79</sup> vgl. Linn (1993)

<sup>80</sup> Pretty Good Privacy (1997)

## GELESEN

Der Zustand soll beschreiben, wann ein AMAgent als gelesen gilt. Dieser kann natürlich erst nach Aktivierung des AMAgenten erreicht werden. Bei diesem Zustand muß man zwischen einem „expliziten Gelesen“ und einem „impliziten Gelesen“ unterscheiden. Das **explizite Gelesen** könnte dann eintreten, wenn der Empfänger dem AMAgenten ausdrücklich mitteilt: „jetzt habe ich die Active Mail gelesen“. Dies könnte der Empfänger z.B. mit einer Schaltfläche READ, die gedrückt werden muß, erklären. Hilfreich könnte dieser Mechanismus sein, um dem Absender mitzuteilen, daß die Active Mail gelesen wurde, sozusagen als „Unterschrift eines Einschreibens“. Jedoch müssen hier weitere Mechanismen beachtet werden, um wirklich sicherzustellen, daß die Rückantwort den Absender erreicht. Um die Komplexität des Systems nicht zu erhöhen, wird deswegen nur ein implizites Gelesen unterstützt. Das **implizite Gelesen** wird automatisch mit der Aktivierung des AMAgenten erreicht, was aber nicht unbedingt bedeuten muß, daß der Empfänger die Active Mail auch wirklich gelesen hat (READ).

## GEANTWORTET

Dieser Zustand kann erst nach Aktivierung des AMAgenten und durch drücken der REPLY-Schaltfläche (falls vorhanden) erreicht werden. Es bleibt der Anwendung der Active Mail überlassen, auf dieses Ereignis zu reagieren bzw. diesen Zustand zu behandeln (siehe auch weiter oben für die verschiedenen Möglichkeiten). Wird der Zustand (READ) erreicht, hat der AMAgent ein definiertes Ende gefunden und terminiert.

## TERMINIERUNG

Das Beenden oder Löschen einer Active Mail braucht eine Sonderbehandlung. Sobald

- ein Fehler auftritt
- der Wunsch des Absenders (Ereignis TERMINATE oder DELETE) oder des Empfängers (Ereignis DELETE) den AMAgenten zu beenden auftritt
- die Active Mail zu einem definierten (sie hat z.B. ihre Aufgabe erfüllt) Ende kommt

muß die Terminierung des AMAgenten behandelt werden. Prinzipiell gibt es zwei Möglichkeiten, eine Active Mail zu terminieren bzw. zu beenden: eine einfache Terminierung oder eine kontrollierte Terminierung.

### **einfache Terminierung**

Hierbei terminiert sich der AMAgent und unterscheidet nicht, ob ein Fehler aufgetreten ist, ob es benutzerinitiiert oder ein definiertes Ende erreicht ist. Der AMAgent informiert weder den Absender, daß ein Fehler aufgetreten (z.B. beim Absenden) noch den Empfänger, daß z.B. bei der Ausführung ein Fehler aufgetreten ist. Diese Möglichkeit bietet den Vorteil, daß sie einfach zu realisieren ist und eine schnelle, robuste Terminierung des AMAgenten gewährleistet, sobald ein Fehler auftritt (es müssen keine weiteren Berechnungen, die evtl. zu weiteren Fehler führen, beachtet werden). Dies könnte auch von Vorteil sein, den AMAgenten zu terminieren, bevor er etwas unvorhergesehenes macht, das aus einem Fehler resultiert. Für den Absender stellt sie jedoch eine unbefriedigende Lösung dar, da er nie eine Fehlermeldung zurückbekommt (z.B. warum seine Active Mail den Empfänger nicht erreichen konnte). Auf der anderen Seite ist es ebenso für den Empfänger unzureichend, daß plötzlich die erhaltene Active Mail terminiert, ohne ihn davon in Kenntnis zu setzen oder ihn zu fragen, ob sie automatisch terminieren darf. Es lassen sich noch einige weitere Gründe finden, warum diese Lösung unzureichend ist. Aufgrund der unzureichenden Eigenschaften kommt nur die folgende Möglichkeit der „kontrollierten Terminierung“ in Betracht:

### **kontrollierte Terminierung**

Die kontrollierte Terminierung ist etwas umfangreicher, aber auch flexibler zu handhaben. Die Kommunikationsmöglichkeiten, die das Agentensystem Mole anbietet, können hier genutzt werden, um ggf. den Absender oder die Empfänger zu erreichen und Informationen (z.B. Fehlermeldungen) zu übermitteln. Hierbei muß aber zwischen einer Eigenterminierung und einer Fremdterminierung unterschieden werden, die unterschiedliche Behandlungen benötigen.

Die **Eigenterminierung** tritt auf, wenn aufgrund eines Fehlers, der Absender seine abgeschickten AMAgenten terminieren läßt oder ein definiertes Ende<sup>81</sup> erreicht wurde und der AMAgent beendet werden muß. Fehler der Eigenterminierung, wie man in den oben gezeigten Zuständen sieht, können bei Versand, Ankunft, Annahme und Ausführung auftreten.

Bei Versand, Empfang und Annahme ist es für den Absender wichtig zu erfahren, welcher Fehler aufgetreten ist. Er muß daher eine detaillierte Fehlermeldung erhalten. Hierbei kann sich der AMAgent die Kommunikationsmöglichkeiten des Agentensystems zunutze machen und den AMSystemAgenten des Absenders mittels einer Nachricht

---

<sup>81</sup> Der AMAgent hat seine Aufgabe erfüllt.

mitteilen, was für ein Fehler aufgetreten ist. Der AMSystemAgent leitet diese Nachricht wiederum dem Benutzer weiter (Anzeige in der Outbox).

Sobald der AMAgent bei seinem Empfänger angenommen wurde, müssen Fehler oder das Erreichen des definierten Endes dem Empfänger mitgeteilt werden. Damit der AMAgent nicht ohne Wissen des Empfängers automatisch terminiert, muß dem Empfänger die Möglichkeit gegeben werden, hier zu intervenieren. Bei drei Ereignissen ist dies von Bedeutung: wenn die Lebensdauer des AMAgenten abgelaufen ist, wenn der Absender seine abgeschickten AMAgenten terminieren läßt oder wenn der AMAgent ein definiertes Ende erreicht. Alle drei Ereignisse sind aus der Sichtweise des Empfängers Eigenterminierungen. Der AMAgent muß also, bevor er bei diesen Ereignissen terminiert, die Einstellungen des Empfängers beachten, um dann ggf. explizit den Empfänger zu fragen, ob er terminieren darf, falls der Empfänger keine automatische Terminierung erlaubt (siehe auch Kapitel 5.3.4). Dies soll auch verhindern, daß der Empfänger Active Mails erhält, die eine Art „Klingelputzen“ darstellt, d.h. kurz erscheinen und dann sofort wieder terminieren, ohne daß der Empfänger den Absender identifizieren kann.

Die **Fremdterminierung** liegt dann vor, wenn von Benutzerseite eine Terminierung (TERMINATE) oder ein Löschen (DELETE) der AMAgenten verlangt wird.

Wird das Ereignis TERMINATE vom Absender ausgelöst (über die Outbox des Absenders, siehe auch Kapitel 5.3.3), wenn er z.B. seine abgeschickten Active Mails „zurückrufen“ will, so werden alle AMAgenten benachrichtigt, daß sie terminieren sollen. Diese Option ist mit Sicherheit nicht notwendig, um die Funktionalität eines Active Mail Systems zu gewährleisten, soll aber verdeutlichen, welche zusätzliche Funktionalität ohne einen großen Aufwand ein Active Mail System in Verbindung mit einem Agentensystem erreichen kann, da schon bestehende Kommunikationsmöglichkeiten des Agentensystem genutzt werden können. Dieses Ereignis wird, nachdem die AMAgenten die Nachricht erhalten, in der Eigenterminierung beschrieben. Das Finden der abgeschickten AMAgenten stellt kein Problem dar, da alle Empfänger und die Namen der AMAgenten (die 64-bit ID von Mole) bei der Verschickung bekannt sind; somit kann die Terminierungsnachricht alle abgeschickten AMAgenten erreichen.

Wird das Ereignis DELETE vom Empfänger ausgelöst (über die Inbox des Empfängers siehe auch Kapitel 5.3.2), so schickt der betroffene AMAgent eine Nachricht an seinen Absender, daß er terminiert. Danach löscht sich der Agent selbständig. Der Benutzer hat die Möglichkeit diese Nachricht an den Absender zu unterbinden, indem er im Setup (siehe auch Kapitel 5.3.4) den Informationsversand zum Absender verbietet.

## LÖSCHEN

Nach der Terminierung löscht sich der AMAgent. Dabei wird auch der Eintrag in der Inbox des Empfängers gelöscht und der Lebenslauf eines AMAgenten ist beendet.

### 5.2 Der Systemagent von ActiveMailMole

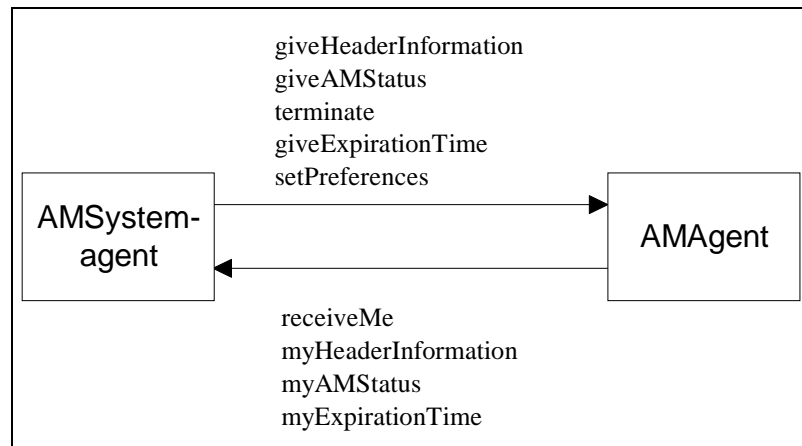
Jeder Benutzer muß einen ortsfesten Systemagenten (AMSystemAgent) auf einer Location haben. Er bildet die Schnittstelle zwischen Benutzer und Agentensystem.

Der Systemagent hat folgende Aufgaben :

1. Darstellung der Benutzerschnittstelle (siehe auch Kapitel 5.3).
2. Verwaltung der Benutzerspezifischen Daten (eingestellt durch das Setup).
3. Erzeugung neuer Active Mail Agenten (AMAgenten) auf einer Location.
4. AMAgenten vom POP3-Server des Benutzers entgegennehmen und auf der Location starten.
5. Empfang und persistente Speicherung von AMAgenten, die sich beim Systemagenten des Benutzers melden.
6. Verwaltung der Inbox und Outbox des Benutzers.
7. Auf Wunsch des Benutzers AMAgenten aktivieren, terminieren und löschen.
8. Verschickung von Kontrollnachrichten an AMAgenten
9. Empfang und Auswertung von Kontrollnachrichten von AMAgenten.

Damit der Systemagent (im weiteren Sinne der Benutzer) für AMAgenten erreichbar wird, muß er sich als **Dienst registrieren** lassen. Als **Dienstname** ist „AMSystemAgent of UserEmailAddress“, also z.B. „AMSystemAgent of nitschjn@trick.informatik.uni-stuttgart.de“, vorgesehen. Dies ermöglicht konzeptionell mehrere AMSystemAgenten auf einer Location, die damit für den AMAgenten unterscheidbar sind. Denkbar wäre auch den schon vorgegebenen Agentennamen anstatt der E-Mail-Adresse des Benutzers zu verwenden. Jedoch müßte dann der Absender einer Active Mail neben den E-Mail-Adressen der Empfänger noch die Agentennamen der Systemagenten angeben, damit die AMAgenten ihre Empfänger erreichen.

Damit ein **Nachrichtenaustausch** zwischen AMAgenten und AMSystemAgent erfolgen kann, muß der AMAgent als auch der AMSystemAgent auf Nachrichten sowie dem Inhalt der Nachricht entsprechend reagieren können. Die Nachrichten, die ein AMAgent „versteh“, wurden schon in Kapitel 5.1.1.2 behandelt.



**Abbildung 20: AMSystemAgent-AMAgent Kommunikation**

Bei einer **receiveMe**-Nachricht möchte sich ein AMAgent beim AMSystemAgenten melden, um angenommen zu werden. Sobald die Anmeldung erfolgreich ist, nimmt der Systemagent diesen Agenten, speichert ihn persistent in der Inbox und informiert den Benutzer über den Eingang einer neuen Mail. Die anderen Nachrichten (**myHeaderInformation**, **myAMStatus**, **myExpirationTime**) sind **Kontrollnachrichten**, die aufgrund der Anfrage des Systemagenten an den AMAgenten zurückgesendet werden. Die Kontrolle eines AMAgenten unterliegt dem Absender bzw. seines AMSystemAgenten, solange der AMAgent nicht den Empfänger erreicht hat. Danach unterliegt die Kontrolle des AMAgenten zwar immer noch dem Absender, aber der AMAgent muß die Einstellungen seines Empfängers beachten. Die Einstellungen (AMAgent darf automatisch terminieren und er darf Informationen über den aktuellen Status an den Absender weiterleiten) können über das Setup gemacht werden.

Generell sind diese Einstellungen dazu gedacht, die Privatsphäre des Empfängers zu schützen, der nicht unbedingt gewillt ist, daß der Absender den Status (z.B. AMAgent wurde gelesen) über seinen abgeschickten AMAgenten beim Empfänger abfragen kann.

Ein weiterer Bestandteil des AMSystemAgenten ist die Benutzerschnittstelle, die im folgenden Kapitel beschrieben wird.

### 5.3 Die Benutzerschnittstelle

Die Benutzerschnittstelle bildet die Interaktionsoberfläche für den Benutzer. Mit ihr kann der Benutzer Active Mails erzeugen, verschicken und ausführen. Mittels einer Inbox und einer Outbox kann der Benutzer Active Mails verwalten.

Prinzipiell gibt es zwei Möglichkeiten die Benutzerschnittstelle an das Agentensystem zu koppeln. Zum einen kann die Benutzerschnittstelle als Bestandteil eines Systemagenten realisiert sein: sämtliche Funktionen der Benutzerschnittstelle können dann durch programminterne Aufrufe innerhalb des Systemagenten ausgelöst werden, der den Programmcode der Benutzerschnittstelle besitzt. Ebenso können Funktionen des Systemagenten direkt von der Benutzerschnittstelle aus, durch programminterne Aufrufe ausgelöst werden. Die andere Möglichkeit ist eine Entkopplung zwischen Benutzerschnittstelle und Systemagent. Die Benutzerschnittstelle könnte dann durch ein externes Zusatzprogramm (z.B. mit einem Applet) realisiert sein, welches über ein Protokoll und eine Datenverbindung mit dem Systemagenten Nachrichten austauscht.

Die entfernte Variante (mit einem Applet) bietet einige Vorteile:

- Ein Benutzer kann sich dynamisch an seinen Systemagenten „ankoppeln“. Das Applet könnte z.B. über den Netscape Navigator gestartet werden und stellt dann automatisch eine Verbindung zu dem Systemagenten des Benutzers her.
- Die Benutzerschnittstelle muß nicht auf demselben Rechner wie das Agentensystem mit seinem Systemagent laufen. Dies kann zum einen den Rechner mit dem Agentensystem entlasten. Zum anderen muß der Benutzer nicht ständig das Agentensystem auf seinem lokalen Rechner laufen lassen, sondern bräuchte nur von Zeit zu Zeit das externe Programm zu starten, um seinen Systemagenten nach neuen Meldungen zu fragen oder ihm neue Meldungen zu übermitteln. Dies würde eine mobile Kommunikation unterstützt.

Jedoch treten (implementierungsbedingt) einige Probleme auf, die die Kommunikation zwischen Systemagent und Applet (externem Programm) betreffen. Nachrichten von einem Applet zu einem Systemagenten zu schicken, stellt kein Problem dar und ist in Mole schon realisiert. Der umgekehrte Weg ist dagegen bedeutend schwieriger, da eine eindeutige Adressierung des Applets nicht möglich ist. Zwar könnte man eine bidirektionale Kommunikation zwischen Applet und Systemagent über ein Socket herstellen, dies bedeutet aber, daß der Systemagent ständig einen Server-Socket bereitstellen muß. Ebenso muß ein Protokoll definiert werden, über welches der Client (das Applet) und

der Server (der Systemagent) kommunizieren können. Eine universelle, bidirektionale Kommunikation zwischen Applet und Systemagent ist ein Thema für sich und nicht Teil dieser Diplomarbeit. Zur Zeit wird im Rahmen einer Diplomarbeit untersucht, wie ein Benutzer ein Mini-Agentensystem auf seinen Browser laden kann. Ebenso soll ein Mechanismus geschaffen werden, der eine Kommunikation zwischen Agent und einem Browserbenutzer bzw. einer Anwendung ermöglicht, die ganz oder teilweise auf WWW-Browsern läuft.<sup>82</sup> Da aber keine Ergebnisse zur Zeit der Diplomarbeit vorliegen, habe ich mich für die interne Variante entschieden. Die interne Variante, d.h. die Benutzerschnittstelle ist Bestandteil des Systemagenten, bietet den Vorteil, daß sie einfach zu realisieren ist. Ebenfalls dürfte eine Portierung der Benutzerschnittstelle in ein Applet keine Schwierigkeiten bereiten, sobald die Problematik der Kommunikation zwischen Applet und Systemagent gelöst ist, da die Unterschiede zwischen einer Benutzerschnittstelle als Anwendung und Applet in Java nur minimal sind.

### 5.3.1 Gestaltung der Oberfläche

Die Gestaltung der Oberfläche des ActiveMailMole Systems orientiert sich maßgeblich an anderen E-Mail-Programmen um für den Benutzer den Einstieg in die Benutzung des Systems möglichst einfach zu machen. Da die Benutzeroberfläche in Java programmiert wird, kann eine Portabilität (in dem Maße, wie Java portabel ist) gewährleistet werden. Um der geforderten Qualität und einfachen Bedienung der Oberfläche gerecht zu werden, wird die Oberfläche mit grafischen Elementen realisiert, die auch intuitiv bedienbar sind (Fenstern, Menüs, Schaltflächen, Listen,...).

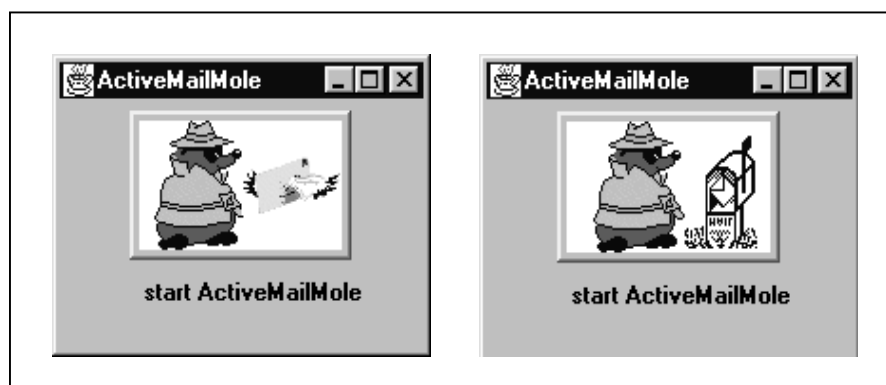
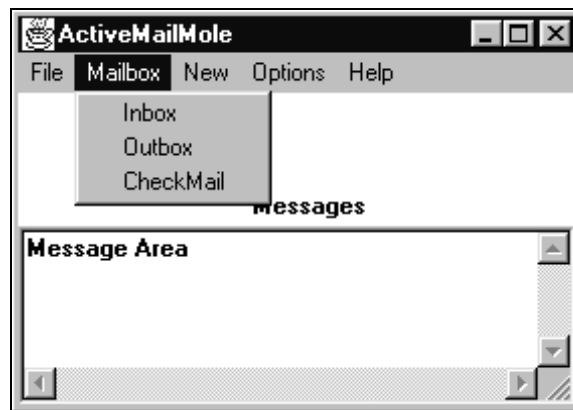


Abbildung 21: Startfenster von ActiveMailMole

Beim Start einer Location des Agentensystems Mole, auf der ein Active Mail Systemagenten des Benutzers vorhanden ist, wird automatisch ein kleines Fenster angezeigt, mit

<sup>82</sup> vgl. Aheimer (1997)

einer Schaltfläche, um das Programm zu starten. Wenn keine neue Mail für den Benutzer vorliegt, wird das linke Fenster in Abbildung 21 angezeigt, bei vorliegenden neuen Mails das rechte Fenster.



**Abbildung 22: Hauptfenster in ActiveMailMole**

Das Hauptfenster des Systems ist in Abbildung 22 zu sehen. Es erscheint, wenn der Benutzer die Startschaltfläche drückt. Die „Message Area“ dient als Informationsfeld, über das der Benutzer eine direkte Rückmeldung vom System (z.B. wenn die Verschickung einer neuen Mail erfolgreich war) bekommt. Über die Menüleiste kann er weitere Fenster öffnen. Das Hauptfenster stellt somit die Steuerzentrale des Programms dar. Folgende weitere Funktionen stehen zu Verfügung:

- **Programmende**  
Über das Menü „File“ und der Auswahl Exit, kann der Benutzer das Programm geordnet beenden.
- **Inbox**  
Über das Menü „Mailbox“ und der Auswahl Inbox, kann der Benutzer ein weiteres Fenster öffnen, das ihm die eingegangenen Active Mails anzeigt, die er gegebenenfalls starten kann.
- **Outbox**  
Möchte der Benutzer seine abgeschickten Mails betrachten, so kann er über das Menü „Mailbox“ und der Auswahl Outbox ein weiteres Fenster öffnen, welches ihm seine verschickten Mails anzeigt. Dieses Fenster enthält weitere Funktionen, die die Kontrolle der abgeschickten Mails beinhalten.
- **Eingang neuer Mails überprüfen**  
Mit der Auswahl CheckMail im Menü Mailbox kann der Benutzer explizit den Systemagenten veranlassen, nach neuen Mails für den Benutzer zu su-

chen. Eine Meldung über neue oder keine neuen Active Mails wird über die Message Area dem Benutzer mitgeteilt (Eine automatische Überprüfung erfolgt je nach Einstellung des Zeitintervalls im Setup durch den Benutzer).

- **Erzeugung neuer Active Mails und normaler E-Mails**

Über das Menü New kann der Benutzer wahlweise eine neue normale E-Mail (Auswahl E-Mail) erzeugen oder mit der Auswahl Active Mail aus einer Auswahl von bestehenden Active Mail-Formularen (z.B. dem Meeting Scheduler) eine neue Active Mail parametrisieren.

- **Einstellungen**

Jeder Benutzer muß bei der ersten Benutzung des Programms ein Setup durchlaufen, um die Funktionalität des Systems zu gewährleisten. Diese Einstellungen können aber auch nachträglich geändert werden und sind jederzeit über das Menü „Options“ abrufbar.

- **Hilfe**

Eine Erklärung der Funktionalität des Programms kann der Benutzer über das Menü „Help“ abrufen.

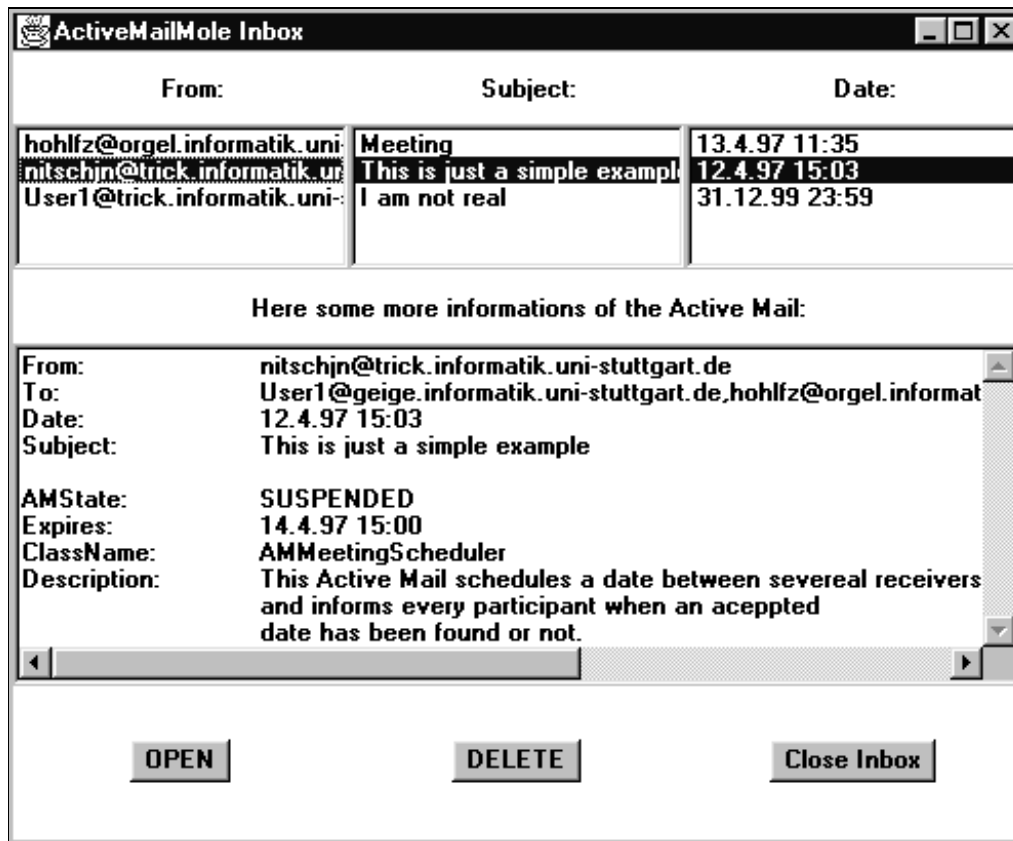
In den folgenden Kapiteln werden nur die wichtigsten Fenster der Anwendung ActiveMailMole beschrieben. Es werden bewußt die Dialogfenster nicht beschrieben, die dem Benutzer Fehlermeldungen, Erfolgsmeldungen oder weitere Informationen geben, sowie die Hilfefenster oder das Fenster zur Erzeugung normaler E-Mails, da sie für das Verständnis des Systems nicht unbedingt notwendig sind.

### 5.3.2 Die Inbox von ActiveMailMole

Die Inbox zeigt dem Benutzer die eingegangene Active Mails an, die der Active Mail Systemagent (AMSystemAgent) entgegengenommen hat (siehe auch Abbildung 23).

Der obere Teil des Fensters zeigt auf einen Blick:

- von wem (From) die Active Mail ist, die E-Mail Adresse des Absenders.
- den Betreff (Subject).
- das Datum (Date) der Erzeugung der Active Mail.



**Abbildung 23: Die Inbox von ActiveMailMole**

In der Mitte ist ein Feld, das dem Benutzer mehr Informationen über die Active Mail gibt, die er in den oberen Listen mit einem einfachen Mausklick auswählen kann. Hier werden nochmals der Absender, der Betreff, das Datum der Erzeugung und eine Liste der Empfänger dieser Active Mail angezeigt. Außerdem werden spezifischere Detaildaten angegeben:

- AMState, der Zustand, in dem sich die Active Mail momentan befindet (siehe auch Kapitel 5.1.1.1).
- Expires, gibt das Datum an, bis zu dem die Active Mail noch gültig ist. Es gibt dem Benutzer ein Indiz, bis wann er diese Active Mail beantworten soll. Nach diesem Zeitpunkt ist die Active Mail nicht mehr gültig und kann nicht mehr ausgeführt werden, wenn der Empfänger eine automatische Terminierung erlaubt. Wenn die Active Mail kein „Verfallsdatum“ hat wird im Feld ein „never“ angezeigt.
- ClassName, gibt dem Benutzer den Klassennamen der Active Mail an, von der die Active Mail eine Instanz ist. Dadurch weiß der Anwender, um welche Applikation es sich bei der Active Mail handelt.

- Description, ist eine Beschreibung der Active Mail Anwendung.

Der untere Teil beinhaltet Schaltflächen, mit denen der Benutzer Aktionen auf der ausgewählten Active Mail ausführen kann. Die Schaltfläche OPEN aktiviert die Active Mail. Die Schaltfläche DELETE löscht sie. Weitere Aktionen, die ausgeführt werden können, sind in der Active Mail selbst enthalten, die je nach Anwendungsbereich der Active Mail differieren können.

### 5.3.3 Die Outbox von ActiveMailMole

Die Outbox zeigt dem Benutzer seine bisher verschickten Active Mails an. Sie dient zugleich als ein Kontrollelement, um bisher abgeschickte Active Mails zu terminieren oder weitere Informationen über den Status der verschickten Active Mails zu bekommen (siehe auch Abbildung 24).

Der obere Teil des Fensters ist fast identisch mit dem Teil der Inbox. Statt dem Absender wird hier der Empfänger bzw. die Empfängerliste (To) angezeigt. Das mittlere Feld unterscheidet sich kaum von dem der Inbox, nur wird hier nicht der aktuelle Zustand der Active Mail angezeigt, sondern eine Information (State), ob die Verschickung der einzelnen Active Mail Agenten erfolgreich war.

Der untere Teil wird nur aktiviert, wenn der Benutzer die Schaltfläche „get info on send“ drückt. Sie zeigen dem Benutzer weitere Informationen über die abgeschickten AMAgenten an (Empfänger, Name des abgeschickten Agenten und den Status des Agenten). Diese sind allerdings nur verfügbar, wenn die jeweiligen Empfänger eine Informationsweiterleitung erlauben.

Mit der Schaltfläche „TERMINATE“ kann der Benutzer seine abgeschickten AMAgenten zur Terminierung auffordern (siehe auch Kapitel 5.1.2).



Abbildung 24: Die Outbox von ActiveMailMole

Mit der Schaltfläche „DELETE“ löscht der Benutzer die Kontrollinstanz seiner abgeschickten AMAgenten. Danach hat er keine Kontrolle mehr über sie.

### 5.3.4 Setup

Das Setup-Fenster wird bei der ersten Benutzung des Programms automatisch ausgeführt (siehe auch Abbildung 25). Die gesammelten Daten werden vom AMSystemAgenten persistent gespeichert. Sie können jederzeit zur Laufzeit des Systems geändert oder angepaßt werden.

**AMM Setup**

### ActiveMailMole -- SETUP

Defaults from your AMSystemAgent

Name of your SystemAgent (your EMail): (AgentName):	nitschjn@trick.informatik.uni-stuttgart.de (1.0.0.0.0.25.25)
Name of the Location:	boston.mole.informatik.uni-stuttgart.de
Number of the Agentport:	8000

---

<b>Your personal data</b>	<b>Mail server data</b>
Your Name: <input type="text" value="Jan-Gregor Nitsche"/>	Mail server user name: <input type="text" value="nitschjn"/>
Your EMail address: <input type="text" value="nitschjn@trick.informatik.uni-stuttgar"/>	Outgoing mail (SMTP) server: <input type="text" value="localhost"/>
Your password (for POP3 Account): <input type="password" value="*****"/>	Incoming mail server (POP3): <input type="text" value="trick.informatik.uni-stuttgart.de"/>
	Check for new mail every (min): <input type="text" value="10"/>

---

**Active Mail preferences**

- allow Active Mail to terminate automatically
- allow sender to get information on your Active Mail Status

**Abbildung 25: Setup-Fenster von ActiveMailMole**

Der obere Teil gibt dem Benutzer voreingestellte Daten über den Systemagenten, wobei hier der Name des Systemagenten neben dem Agentennamen (die 64-bit ID von Mole) noch die E-Mail-Adresse des Benutzers als weitere Identifikation benötigt wird. Sie ist bei der Anmeldung des Systemagenten an der Location notwendig, um ihn als Dienstanbieter von Active Mail des Benutzers zu identifizieren (siehe auch Kapitel 5.2).

Bei den persönlichen Daten ist die E-Mail-Adresse unbedingt notwendig, um die Funktionalität des Systems zu gewährleisten. Es kann davon ausgegangen werden, daß jeder Benutzer eine solche besitzt. Sie wird neben der Identifizierung des Systemagenten auch dazu benutzt, den Absender einer Active Mail eindeutig zu identifizieren. Damit können gegebenenfalls Kontrollinformationen von abgeschickten Active Mails an den Absender zurückgeschickt werden.

Die weiteren Felder (Paßwort, Mail Server Data) werden benutzt, um die Verschickung normaler E-Mail zu ermöglichen und Active Mails für den Benutzer von seinem POP3-

Server zu lesen. Der POP3-Server wird benötigt, wenn Active Mails nicht mit einer Migration zum Empfänger gelangen, sondern als E-Mail an die Adresse des Mail Servers des Empfängers geschickt wird (siehe auch Kapitel 5.1.1).

Über die „Active Mail preferences“ kann der Benutzer einstellen, ob er dem Absender erlaubt Informationen (z.B. ob die Active Mail schon gelesen wurde) über den Status seiner Active Mails einzuholen. Dem Benutzer steht es ebenso offen, ob er eine automatische Eigenterminierung der Active Mails erlaubt.

## 5.4 Beispielanwendungen

### Terminvereinbarung

Um die Funktionalität des Systems unter Beweis zu stellen ist eine Beispielanwendung, die eine Terminvereinbarung zwischen mehreren Benutzern realisiert, entwickelt worden. Die Terminvereinbarung als Active Mail Anwendung wird in der Klasse **AMMeetingScheduler** beschrieben. Diese Klasse erweitert die Klasse **AMGeneralForm**, um diese spezifische Aufgabe zu lösen. Die Methoden, die zur Verschickung, Ankunft, Annahme, Unterbrechung, Gelesen und Terminierung eines AMAgenten in der Klasse **AMGeneralForm** beschrieben sind, werden auch hier verwendet. Nur die Methoden für die Erzeugung, Aktivierung und Auswertung sind neu definiert. Desweiteren werden neue Attribute benötigt, um Informationen über die Termine und einen Einleitungstext, sowie den Namen des Agenten zu bewahren, der eine Auswertung der Antworten übernimmt.

Bei der Erzeugung (**executeOnCreation**), wird die Initialisierung der Terminvereinbarung vorgenommen (siehe auch Abbildung 26). Die grau unterlegte Fläche zeigt den standardmäßig angezeigten Header, die weiße Fläche ist die spezielle Initialisierung für die Terminvereinbarung. Hier kann der Absender drei Termine eingeben, zu denen er ein Treffen wünscht, sowie einen Einleitungstext für die Empfänger.

The screenshot shows a window titled "ActiveMailMole HEADER definition". The window contains the following fields and controls:

- To:** A text area containing: "User1 @geige.informatik.uni-stuttgart.de@location1.mole.de, hohlfz@orgel.informatik.uni-stuttgart.de@boston.mole.informatik.uni-User2@trick.informatik.uni-stuttgart.de"
- From:** A text field containing: "nitschjn@trick.informatik.uni-stuttgart.de@location2.mole.de"
- Subject:** A text field containing: "This is just a simple example"
- Buttons:** "Send" and "Cancel" buttons.

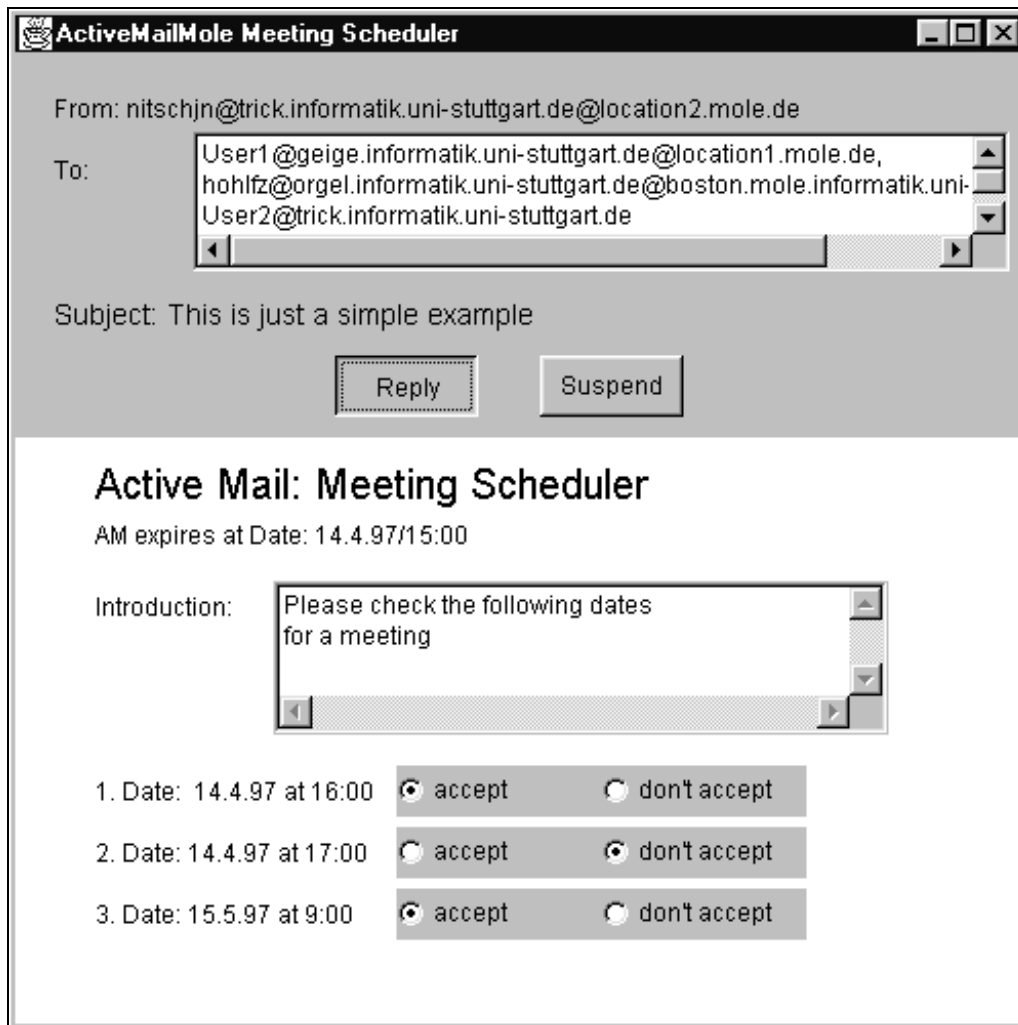
Below the header definition is a section titled "Initialisation of the Active Mail: Meeting Scheduler". This section contains:

- AM expires at Date (dd.mm.yy/hh.mm):** A text field containing "14.4.97/15:00".
- Introduction:** A text area containing "Please check the following dates for a meeting".
- Meeting Schedules:** A list of three items, each with a date and time field:
  - 1. Date (dd.mm.yy) and time(hh:mm): 14.4.97 16:00
  - 2. Date: 14.4.97 17:00
  - 3. Date: 15.5.97 9:00

**Abbildung 26: Initialisierung des AMMeetingScheduler**

Nachdem er die SEND-Schaltfläche drückt werden neben den AMAgenten noch ein weiterer Agent erzeugt, der die Auswertung der Empfängerantworten übernimmt. Der Agent **AMScheduleReply** bleibt lokal auf der Location des Absenders und wird mit allen Empfängern und den Terminen, die zur Auswahl stehen initialisiert und wartet auf eine Rückantwort der Empfänger.

Nach der Aktivierung (**executeOnActivation**) des AMAgenten beim Empfänger, wird der Benutzer interaktiv aufgefordert, die Termine auszuwählen (über anklicken der „Checkboxes“), bei denen er zur Verfügung steht (siehe auch Abbildung 27).



**Abbildung 27: Aktivierung des AMMeetingScheduler**

Drückt der Empfänger die REPLY-Schaltfläche, wird die Methode **executeOnReply** ausgeführt, in der nur der Teil der ausgewählten Checkboxes evaluiert wird und eine Nachricht an den AMScheduleReply Agenten mit dieser Auswertung zurückgeschickt.

### AMScheduleReply Agent

Er wird erzeugt, nachdem der Absender die SEND-Schaltfläche bei der Initialisierung gedrückt hat. Hierbei werden die Informationen über die Empfänger, den Absender, die Termine und der Lebensdauer der AMAgenten mit übergeben. Der AMScheduleReply wartet lokal auf die Rückantwort der Empfänger.

- **evaluationOfSchedule-Nachricht**  
Die abgeschickten AMAgenten können mittels der **evaluationOfSchedule-Nachricht**, dem AMScheduleReply die Auswertung der Empfänger mitteilen. Dabei übermitteln sie die Termine, die der Empfänger akzeptiert hat.

Haben alle Empfänger eine Rückantwort geschickt, nimmt der AMScheduleReply Agent den ersten Termin, der von allen Teilnehmern akzeptiert wurde und schickt eine Nachricht an alle Teilnehmer (siehe auch Abbildung 28).

**Date:** Tue, 13 Apr 1997 11:35:07 +0200 (MET DST)  
**From:** nitschjn@trick.informatik.uni-stuttgart.de  
**To:** nitschjn@trick.informatik.uni-stuttgart.de  
User1@geige.informatik.uni-stuttgart.de,  
hohlz@orgel.informatik.uni-stuttgart.de,  
User2@trick.informatik.uni-stuttgart.de  
**Subject:** Active Mail Meeting Scheduler: we have found a meeting date!

The Meeting Scheduler of ActiveMailMole has found a meeting date:  
Date 14.4.1997 at 16:00  
has been accepted by all participants.

### **Abbildung 28: Ergebnis des AMMeetingSchedulers**

Das Ergebnis wird als normale E-Mail abgeschickt. Denkbar wäre es auch, daß er eine weitere Active Mail an alle Teilnehmer schickt; aber der einfachere Weg ist die normale E-Mail.

Wird die Lebensdauer der Terminvereinbarung überschritten, oder die Teilnehmer können sich nicht auf einen Termin einigen, wird ebenso eine Normale E-Mail abgeschickt, die den Mißerfolg an die Teilnehmer weiterleitet.

### **Weitere Anwendungen**

Weitere Anwendungen des Active Mail Systems wären denkbar und wünschenswert, um den Nutzen eines solchen Systems zu steigern. Anwendungen, wie sie mit ATOMICMAIL (siehe auch Kapitel 2.5.2.1) entwickelt wurden (Umfragegenerator, Dokumentenverteiler, ...), könnten auch mit diesem System realisiert werden, weitere Anwendungen könnten sein:

#### **1. Multimedia-Präsentation**

Eine Anwendung, mit der der Absender bei der Parametrisierung der Active Mail, eigene Videos oder animierte Bilder, inklusive Tondaten einfügen kann und diese an-

deren Personen zuschickt (z.B. als Präsentation eines Unternehmens oder einer Abteilung).

## 2. Kooperative Erstellung von Dokumenten

Die Anwendung könnte sich hier die zusätzlichen Kommunikationsmöglichkeiten (speziell die synchrone Kommunikation) des Agentensystems zunutze machen und die Koordination zur kooperativen Erstellung eines Dokumentes übernehmen.

## 3. Erweiterung der Terminvereinbarung

Die Terminvereinbarung könnte auch erweitert werden, um Empfängern die Möglichkeit zu geben, andere Teilnehmer einzuladen. Dabei muß beachtet werden, daß die vom Absender ausgehende Terminierung der AMAgenten auch die AMAgenten berücksichtigt, die weiter geschickt wurden. Hier könnte ein Mechanismus hilfreich sein, um die weitergeleiteten AMAgenten zu finden. Im Rahmen einer Studienarbeit wurde ein Agent-Locator für Mole realisiert, der als Dienst den aktuellen Aufenthaltsort eines Mobilen Agenten bestimmt.<sup>83</sup> Dieser Dienst könnte hier verwendet werden, um AMAgenten zu finden.

Die Flexibilität des Systems erlaubt eine Vielzahl von weiteren Anwendungen, die den Nutzen für den Anwender steigern.

## 5.5 Zusammenfassende Bewertung von ActiveMailMole

Zusammenfassende Bewertung der wesentlichen Eigenschaften von ActiveMailMole in Gegenüberstellung mit dem in Kapitel 2.5.3 vorgestellten System:

(Die Bewertung erfolgt in 4 Stufen (++) sehr gut, (+) gut, (-) weniger gut, (--) nicht ausreichend.)

---

<sup>83</sup> vgl. Röhrle (1996)

System	ActiveM3 (AM3)	ActiveMailMole (AMM)	Bemerkungen
<b>Eigenschaften</b>			
Sicherheit	+	+	Sicherheit von AMM wird vom Agentensystem gewährleistet
Unterstützung heterogener Systeme	+	++	AMM verwendet Java-Interpreter. Sun Microsystems leistet große Anstrengungen, den Interpreter für viele verschiedene Systeme zur Verfügung zu stellen.
Mächtigkeit der Sprache	+	++	Java ist eine „mächtige“ Sprache für AMM
Programmausführung	ja	ja	
Lesbar mit normalem Mail-Reader	ja	nein	Spezieller Mechanismus in AMM wurde nicht berücksichtigt
einfache Erstellung	++	++	AMM erlaubt einfache Erstellung von Agenten (durch Formulare)
Erweiterbarkeit (neue AM Anwendungen)	+	++	AMM ist mit neuen Anwendungen einfach zu erweitern. AMM kann sich zusätzliche Eigenschaften von Mole zunutze machen (Dienste, Anwendungen)
einfache Installation	++	-	In AMM wäre dies möglich, ist aber nicht realisiert.
Kooperationsfähigkeit	nein	ja	AMM ermöglicht: „erweitertes mail handling, Agent-Agent Kommunikation (Kontrollinformationen, Yellow Pages), evtl. synchrone Kommunikation mit anderen Teilnehmern
Autonomie	beschränkt	ja	AMM Agenten müssen aber Einstellungen des Empfängers beachten, AM nur durch Interaktion mit Benutzer
spezielle Benutzerschnittstelle	+(nur zur Erzeugung)	-(ja)	AMM benötigt eine spezielle Benutzerschnittstelle zur Erzeugung und Verwaltung von Active Mail Agenten

**Tabelle 3: Gegenüberstellung der Systeme „ActiveM<sup>3</sup>“ und „ActiveMailMole“**

Die Gegenüberstellung macht deutlich, daß AMM viele Eigenschaften eines Active Mail Systems beinhaltet und darüber hinaus noch weitere Eigenschaften besitzt, mit der die Leistungsfähigkeit von Active Mails durch die Verwendung von Mobilien Agenten gesteigert wird.

Hier noch einige weitergehende Erläuterungen:

- Damit AM Agenten auch mittels eines normalen Mail-Readers ausführbar sind, müßte der Mechanismus, der den Interpreter zur Ausführung des Agentensystems aufruft, näher untersucht werden. Hier entstehen hauptsächlich zwei Probleme: zum einen müßte bei einem „laufenden“ Agentensystem dieses nicht nochmals aufgerufen, sondern nur eine Verbindung hergestellt werden. Zum anderen besteht die Frage, welche Location gestartet werden muß, wenn kein Agentensystem momentan läuft. Da jeder Benutzer eigene Locations aufmachen und den Aufenthaltstort seines AM-

SystemAgenten wechseln kann, müßte dieser Mechanismus des öfteren angepaßt werden. Dies stellt jedoch eine sehr unbefriedigende Lösung dar.

**Kritische Anmerkungen:**

- Die Verbindung von Active Mail und Mobilen Agenten bietet keine Lösung, eine allgemeine Architektur eines Active Mail Systems zu realisieren. Es sind weder Standards für Active Mails noch für Mobile Agenten vorhanden. Dies wird auch in nächster Zukunft so bleiben, wenn man bedenkt wie schwierig es ist, sich weltweit auf einen Zeichensatz (ASCII) zu einigen.
- Der Traum, „Active Mails für jedermann“ ist mit dieser Lösung auch nicht erreichbar, da zusätzliche Komponenten (Agentensystem Mole) benötigt werden und nicht jeder diese Komponenten installieren kann (Kompetenz und Portabilität des Systems). Es ist auch fraglich, ob dieser Traum jemals in Erfüllung gehen kann und die Benutzung eines Active Mail Systems ebenso einfach wird wie die Benutzung der normalen E-Mail.

## **6 Zusammenfassung**

### **6.1 Ergebnisse**

Im Rahmen dieser Diplomarbeit wurden bestehende elektronische Postsysteme kurz vorgestellt und analysiert, um die Herausforderungen an Active Mail Systeme aufzuzeigen. Desweiteren wurden verschiedene Active Mail Systeme vorgestellt und allgemeine Anforderungen an ein Active Mail System herausgearbeitet.

Die Vor- und Nachteile bei der Realisierung eines Active Mail Systems mit Mobilien Agenten erarbeitet und in einem Vergleich gegenübergestellt. Dabei kann festgestellt werden, daß beide Konzepte viele Eigenschaften gemeinsam haben und sich gegenseitig gut ergänzen.

In einem weiteren Schritt wurde ein System „ActiveMailMole“ entworfen, welches Active Mails auf Basis Mobiler Agenten realisiert. Es wurde eine allgemeine Struktur für Active Mail Anwendungen entwickelt, die eine Erzeugung, Ausführung und Terminierung von Active Mail Agenten in Interaktion mit dem Benutzer erlaubt. Verschickung, Empfang und Auswertung der Agenten sind dabei automatisiert. Vorhandene Mechanismen von Mobilien Agenten des Agentensystems Mole wurden integriert.

Das entwickelte System zeigt auf, daß Active Mails unter Verwendung von Mobilien Agenten gut zu realisieren sind und die wesentlichen Eigenschaften, die ein Active Mail System fordert, erfüllt werden können. Darüber hinaus erlaubt der Einsatz von Mobilien Agenten (als Active Mail) ein Active Mail System noch leistungsfähiger zu machen. Dadurch ist es z.B. möglich, dem Absender die Kontrolle und zusätzliche Informationen über den Status seiner abgeschickten Active Mail Agenten zu geben.

Das System bietet gute Einsatzmöglichkeiten zur Unterstützung von Gruppenarbeit in verteilten Systemen sowie in anderen Bereichen (z.B. Multimedia-Präsentationen).

Mit der Benutzung von vorgefertigten Formularen können Active Mails einfach generiert werden, ohne sie neu programmieren zu müssen. Ebenso besteht die Möglichkeit, neue Active Mail Anwendungen mit eigenen Formularen in das System einzubinden. Ferner wurde ein Werkzeug realisiert, mit dem der Benutzer Active Mails empfangen, lesen, erzeugen und kontrollieren kann.

## 6.2 Ausblick

Bei einem System wie ActiveMailMole existieren viele Erweiterungsmöglichkeiten, die den ohnehin knappen Zeitrahmen einer Diplomarbeit bei weitem übersteigt.

Es könnten weitere Anwendungen für das Active Mail System entwickelt werden, z.B. einem Umfragegenerator oder auch eine Integration von Videokonferenzen für mehrere Teilnehmer, wobei die Active Mail eine Verbindung zu einem externem Programm herstellt, sobald sie ausgeführt wird. Dabei könnte die Lebensdauer einer Active Mail einen Hinweis geben, bis wann eine Videokonferenz hergestellt werden kann. Weitere Anwendungen sind denkbar und würden den Nutzen, den ein Anwender aus diesem System ziehen könnte, steigern.

Weiterhin könnte eine n-m Kommunikation, d.h. mehrere Teilnehmer könnten mit vielen anderen Teilnehmern kommunizieren, mit einer Active Mail Anwendung realisiert sein, ähnlich der Funktionsweise von News-Groups. Ein spezieller Agent nimmt Active Mail Agenten entgegen und generiert automatisch HTML-Seiten, die über das Internet allgemein oder Paßwort geschützt sind und die entsprechenden Schnittstellen der eingegangenen Active Mail Agenten enthält. Über diese Schnittstellen könnten die Besucher der Seite, ohne ein Active Mail System installiert zu haben, mit Active Mail Agenten in Interaktion treten. Die Active Mail Agenten könnten eine Auswertung übernehmen und zu einem bestimmten Zeitpunkt dem Initiator (oder auch allen Besuchern) ein Ergebnis liefern.

Es wäre auch denkbar, daß abgeschickte Active Mail Agenten periodisch Informationen vom Absender holen, um etwaige Änderungen zu berücksichtigen und die Empfänger automatisch davon in Kenntnis zu setzen. Es müßten keine neuen Active Mail Agenten verschickt werden.

Durch eine Erweiterung der Benutzerschnittstelle von ActiveMailMole könnten auch Mobile Agenten von Mole auf einfache Weise erzeugt werden. Jeder Agent müßte nur ein Formular bereitstellen, mit dem er sich selbst parametrisieren läßt.

Letztendlich wäre es möglich, Active Mail Anwendung mittels einem Grafik-Editor intuitiv zu programmieren. Jedoch benötigt dieses Konzept eine eingehende Untersuchung, ob alle Funktionen eines Active Mail Agenten grafisch zu realisieren sind. Schwierigkeiten treten hier bei der Definition vom Grad der Autonomie und Kooperationsfähigkeit auf.<sup>84</sup> Wären diese Probleme gelöst, könnte eine Übertragung auf andere

---

<sup>84</sup> vgl. Schirmer(1996b)

Agenten von Mole realisierbar sein, und damit wären neue Agenten ebenso grafisch programmierbar.

## **Anhang**

### **A Glossar**

Begriffe, wie sie in der Diplomarbeit verwendet werden, sind hier erklärt.

#### **Active Mail**

Elektronische Briefe, die nicht nur statische Daten, sondern auch Programme transportieren können, die zu bestimmten Zeitpunkten ausgeführt werden und so wesentlich flexibler Inhalte darstellen und in Interaktion mit dem Leser treten können.

#### **Agent**

Ein unabhängiges Programm, das in der Lage ist, seine Entscheidungen und sein Handeln, aufgrund der Wahrnehmung seiner Umwelt, bei der Verfolgung eines oder mehrerer Ziele, selbständig zu kontrollieren.

#### **Agentensystem**

Eine verteilte abstrakte Schicht, die auf der einen Seite dem darunterliegenden System Sicherheit bietet und auf der anderen Seite Konzepte und Mechanismen für Mobilität und Kommunikation für Agenten zur Verfügung stellt.

#### **AMAgent**

Eine Instanz einer Active Mail Anwendung in ActiveMailMole.

#### **AMSystemAgent**

Der Systemagent eines Benutzers in ActiveMailMole. Er stellt die Schnittstelle zwischen Benutzer und Agentensystem dar.

#### **Asynchrone Kommunikation**

Die Übermittlung, bzw. Austausch von Informationen, zwischen verschiedenen Objekten, zu verschiedenen Zeiten, z.B. bei einem Briefwechsel zwischen Personen.

#### **Client**

Arbeitsstation, Programm oder Prozeß, welche die Dienste eines Servers in Anspruch nimmt.

#### **E-Mail**

Elektronische Post (electronic mail).

**Ereignis**

(in der dynamischen Modellierung) Ein Geschehen in einem bestimmten Augenblick.

**Internet Protocol (IP)**

Ein Protokoll, das ein verbindungslosen, datagrammorientierten Netzwerkdienst zum Versenden von Datenpaketen zwischen Rechnern realisiert.

**Internet**

Weltweites Netzwerk, welches früher hauptsächlich zu wissenschaftlichen Zwecken (im ARPA Netz) verwendet wurde, heute wird es jedoch immer mehr kommerziell genutzt. Das Internet stellt dem Benutzer viele Dienste zur Verfügung, wie elektronische Post, Informationsrecherche (WWW, Archie, Gopher), Datentransfer (FTP) und andere Dienste. Das Internet basiert auf den Netzwerkprotokollen IP und TCP, sowie UDP.

**Interpreter**

Ein Programm, welches ein Programm einer anderen Programmiersprache nach der notwendigen syntaktischen Überprüfung sofort ausführt.

**JAVA**

Eine architekturunabhängige, objektorientierte Programmiersprache. Sie wurde von SUN Microsystems entwickelt. JAVA Quellcode wird in einen Stackorientierten Bytecode übersetzt, der interpretiert wird. Die Sprache hat einige Eigenschaften, die sie für ein Agentensystem interessant macht. Insbesondere die Sicherheitsmaßnahmen von JAVA.<sup>85</sup>

**Migration**

Bei der Migration unterbricht der Agent seine Ausführung und „wandert“ von einem Ort zu einem anderen Ort.

**Mobile Agenten**

Aktive, autonome Objekte, die sich zwischen Orten in einem sogenannten Agentensystem bewegen können.

**Mole**

Ein Agentensystem für Mobile Agenten, das in der Abteilung für Verteilte Systeme des Instituts für Parallele und Verteilte Höchstleistungsrechner der Universität Stuttgart entwickelt wird.

---

<sup>85</sup> vgl. Sun Microsystems (1995)

**Netzwerk**

Ein System mehrerer untereinander Verbundener Rechner, die einen Austausch von Nachrichten und die gemeinsame Nutzung von Dienstleistungen bzw. Ressourcen nutzt.

**Port**

Ein Konzept zur Adressierung von Prozessen in Rechnern. Der Port bildet Endpunkt einer Kommunikation zwischen zwei Endsystemen im Netzwerk.

**Ressourcen**

Ein genereller Ausdruck für Hilfsmittel. Speziell In der Informatik werden damit beispielsweise Speicherplatz, Rechenzeit oder Hardwarekomponenten bezeichnet.

**Server**

Netzwerkstation, Programm oder Prozeß, welche für Clients Dienste anbietet oder Ressourcen zur Verfügung stellt.

**Socket**

Sockets stellen die Grundlage für die Abstraktion von Ein- und Ausgaben innerhalb eines Netzwerks dar. Mit diesem Begriff wird allgemein eine Software bezeichnet, mit der das Senden und Empfangen von Daten über ein TCP/IP-Netzwerk möglich ist. Zu einem Socket gehören drei grundlegende Elemente: Die IP-Adresse, zu der eine Verbindung besteht, der Port über den eine Verbindung läuft und der Socket-Typ.

**Synchrone Kommunikation**

Die Übermittlung, bzw. Austausch von Informationen, zwischen verschiedenen Objekten, zur selben Zeit. Als Beispiel könnte das Telefonieren zwischen Personen gesehen werden.

**Systemagent**

Ein spezieller Agent im Agentensystem Mole, der nicht migrieren kann und z.B. Dienste anbieten kann.

**Transition**

Eine Zustandsänderung, die durch ein Ereignis verursacht wird.

**Transmission Control Protocol (TCP)**

Ein Protokoll, das auf IP aufsetzt und eine gesicherte Datenübertragung zwischen zwei Endsystemen gewährleistet.

**Zustand**

Die Werte der Attribute und Verknüpfungen eines Objektes zu einem bestimmten Zeitpunkt.

**Zustandsdiagramm**

Ein gerichteter Graph, in dem Knoten Systemzustände und Kanten Transitionen zwischen Zuständen repräsentieren.

## Literaturverzeichnis

- Aheimer, Holger (1997): **Erschließen von WWW-Browsern und -Servern für die Ausführung von mobilen Agenten**. Universität Stuttgart, Fakultät Informatik, Diplomarbeit, Abgabe Dezember 1997.
- Borenstein, Nathaniel S. (1991): Multimedia Electronic Mail: Will the Dream become reality? **Communications of the ACM (CACM)**, April, 1991 S.117-119.
- Borenstein, Nathaniel S. (1992): Computational Mail as Network Infrastructure for Computer-Supported Cooperative Work. **Proceedings of ACM CSCW'92 Conference on Computer-Supported Cooperative Work**, S. 67-74.
- Borenstein, Nathaniel S. (1994): **EMail With A Mind of Its Own: The Safe-Tcl Language for Enabled Mail**. ULPA, Barcelona.  
<ftp://ftp.fv.com/pub/code/ohter/safe-tcl.tar>
- Borenstein, Nathaniel S., N. Freed (1996): **Multipurpose Internet Mail Extension (MIME) Part One: Format of Internet Message Bodies**. Network Working Group. RFC 2045.
- Brockhaus, der große (1952) **F.A. Brockhaus** Wiesbaden.
- Crispin, M. (1996): **Internet Message Access Protocol - Version 4**, University of Washington. RFC 2060.
- Crocker, David H. (1982): **Standard for the format of ARPA Internet text messages**. Dept. of Electrical Engineering, University of Delaware, Newark, DE 19711. RFC 822.
- Duden Informatik (1989): **Duden Informatik, ein Sachlexikon für Studium und Praxis**. Dudenverlag, Mannheim, Wien, Zürich.
- Fritzinger, J.S., M. Mueller (1996): **Java Security**, Sun Microsystems, Inc.  
<http://java.sun.com/docs/white/index.html>
- Geesink, L. H. (1992): **Active Mail Systems: distributed aspects, architecture and applications**. Universität Amsterdam, Dissertation.
- General Magic, Inc. (1996): **The Telescript Language Reference**.  
[http://www.genmagic.com/Telescript/TDE/TDEDOCS\\_HTML/telescript.html](http://www.genmagic.com/Telescript/TDE/TDEDOCS_HTML/telescript.html)

- Gilbert, D., P. Janca (1995): **IBM Intelligent Agents**. IBM Corporation, Research Triangle Park, NC USA.  
<http://www.raleigh.ibm.com/iag.iaghome.html>
- Goldberg, Yaron, Safran M., Shapiro E. (1992): Active Mail - A Framework for Implementing Groupware. **Proceedings of ACM CSCW'92 Conference on Computer-Supported Cooperative Work**, S.75-83.
- Harrison, Colin G., David M. Chess, Aaron Kershenbaum (1995b): **Mobile Agents: Are they a good idea?** IBM Research Division, T.J. Watson Research Center, Yorktown Heights, NY 10598.  
<http://www.research.ibm.com>
- Hohl, Fritz (1995): **Konzeption eines einfachen Agentensystems und Implementation eines Prototyps**. Universität Stuttgart, Fakultät Informatik, Diplomarbeit Nr.1267.
- Hohl, Fritz (1996): **Mole Alpha Documentation**. Universität Stuttgart, Abteilung Verteilte Systeme des IPVR.  
<http://www.informatik.uni-stuttgart.de/ipvr/vs/projekte/mole.html>
- Hyacinth, S. Nwana (1996): Software Agents: An Overview. **Knowledge Engineering Review 11(3)**, S. 1-40.  
<http://www.sce.carleton.ca/netmanage/docs/AgentsOverview/ao.html>
- ITU-T X.400 (7/1996): Message Handling Systems (MHS): Service and system overview **ITU-T Recommendation X.400**.
- ITU-T X.402 (11/1995): Message Handling Systems (MHS): Overall Architecture **ITU-T Recommendation X.402**.
- Jennings, N. R., M. Woodridge (1996): Software Agents. **IEE Review 42(1)**, S. 17-21.  
<http://www.doc.mmu.ac.uk/STAFF/mike/iee-review96.ps>
- Li, W., D.G. Messerschmitt (1996): **Java-To-Go. Itinerative Computing Using Java**.  
<http://ptolemy.eecs.berkeley.edu/dgm/javatools/java-to-go/>
- Linn, J. (1993): „**Privacy Enhancement for Internet Electronic Mail: Part I: Message Encryption and Authentication Procedures**“, IAB IRTF PSRG, IETF PEM WG, RFC 1421

- Ludewig, Jochen (1991): **Einführung in die Informatik**. Verlag der Fachvereine Zürich, 3. Aufl. Zürich.
- Maes, P.(1994): Agents that Reduce Work and Information Overload, **Communications of the ACM (37)7**, S. 31ff
- Magedanz T., K. Rothermel, S. Krause (1996): **Intelligent Agents: An Emerging Technology for Next Generation Telecommunication?** INFOCOM '96, San Francisco, CA, USA, März 1996.
- Meyers, J., C. Mellon, M. Rose (1996): **Post Office Protocol - Version 3 (POP3)** Dover Beach Consulting, Inc. RFC 1939.
- Müller, Jürgen Hrsg. (1993): **Verteilte Künstliche Intelligenz: Methoden und Anwendungen**. B.I. Wissenschaftsverlag, Mannheim [u.a.].
- Ousterhout, John K., Jacob Y. Levy, Brent B. Welch (1996): **The Safe-TCL Security Model**. Sun Microsystems Laboratories, Mountain View, CA 94043-1100.
- Peine, H. (1996): **Das Ara Projekt**.  
[http://www.uni-kl.de/AG-Nehmer/Ara/ara\\_D.html](http://www.uni-kl.de/AG-Nehmer/Ara/ara_D.html)
- Postel, Jonathan B. (1982): **Simple Mail Transfer Protocol (SMTP)**. Information Sciences Institute, University of Southern California. RFC 821.
- Pretty Good Privacy, Inc. (1997): „**Privacy Matters**“,  
<http://www.pgp.com/privacy/privacy.cgi>
- Röhrle, Klaus (1996): **Finden von Mobilen Agenten in einem weitverteilten System**. Universität Stuttgart, Fakultät Informatik, Studienarbeit Nr. 1539.
- Rumbaugh J., M. Blaha, W. Permerlani (1993): **Objektorientiertes Modellieren und Entwerfen**. Carl Hanser Verlag München Wien, Prentice-Hall International Inc. London.
- Schirmer, J., T. Kirste (1996a): Active M<sup>3</sup> - An authoring system for active multimedia mail. **Proceedings SPIE Multimedia Computing and Networking (MMCN'96)**. San Jose (CA), USA.
- Schirmer, J., T. Kirste (1996b): The idea: Integrating authoring concepts for mobile agents into an authoring tool for active multimedia mail. **Proceedings IFIP'96 Conference on Mobile Communications**. Canberra, Australia.

- Sommerville, Ian (1992): **Software Engineering**. Addison-Wesley Publishing Company, 4. Aufl.
- Straßer, Markus, J. Baumann, F. Hohl (1996): Mole - A Java Based Mobile Agent System. **Proceedings of the 2<sup>nd</sup> ECOOP Workshop on Mobile Object Systems**, University of Linz, Austria. dpunkt.
- Sun Microsystems (1995): „**The Java Language Environment: A White Paper**“.  
[http://javasoft.com/whitePaper/java-whitepaper\\_1.html](http://javasoft.com/whitePaper/java-whitepaper_1.html)
- Turban, E. (1990): **Decision Support and Expert Systems - Management Support Subsystems - 2<sup>nd</sup> edition**, MacMillian Publishing Company, New York, USA, S. 846.
- Vittal, J. (1981): Active Message Processing: Messages as Messengers. **Proceedings of the IFIP TC 6 International Symposium**, Ottawa, Canada, S. 175-195.
- White, J.E. (1996): **Mobile Agents White Paper**. General Magic White Paper.  
<http://www.genmagic.com/agents/Whitepaper/whitepaper.html>
- Woodridge, M.J., N.R. Jennings (1995): Intelligent Agents: Theory and Practice. **The Knowledge Engineering Review** 10(2),S. 115-152.  
<http://www.doc.mmu.ac.uk/STAFF/mike/ker95.ps>

## **Abschließende Erklärung**

Ich versichere, daß ich diese Arbeit  
selbständig verfaßt und nur die  
angegebenen Hilfsmittel verwendet habe.

Stuttgart, den 31.Mai 1997