

# Mobile Agenten - Grundlagen

Eddie Mönch

## Zusammenfassung

Unter Mobilen Agenten versteht man Programmkonstrukte, die die Fähigkeit haben, autonom zu handeln, zu kommunizieren und von Rechner zu Rechner zu migrieren, falls auf beiden Seiten ein Agentensystem installiert ist. Das Modell der Mobilen Agenten ist ein innovativer Ansatz aus dem Bereich der Verteilten Systeme, dem im Hinblick auf zunehmende Leistungsfähigkeit der Telekommunikationssysteme und -netze, sowie der steigenden Anzahl an mobilen Netzteilnehmern, eine Vision unterliegt: Agenten zum selbständigen Erledigen einer bestimmten Aufgabe im Netz zu programmieren. Dieser Beitrag diskutiert die wesentlichen Eigenschaften, die Vorteile und Problemstellungen, einige Anwendungsbeispiele sowie Expertenmeinungen zum Thema Mobile Agenten.

## 1 Einleitung

Das Forschungsgebiet der Mobilen Agenten beschäftigt sich mit autonom arbeitenden Softwareprogrammen, die zu mehreren parallel ein Problem lösen können und in der Lage sind, ihren Aufenthaltsort selbständig zu ändern: den *Mobilen Agenten*. Die Voraussetzung hierfür ist, dass an allen potentiellen Aufenthaltsorten eine sogenannte *Agentenplattform* installiert ist. Eine Agentenplattform ist eine Art Laufzeitumgebung für Mobile Agenten, die den Agenten unter anderem Mechanismen zur Kommunikation und Migration bereitstellt. Die Einheit aus Agentenplattform und Mobilen Agenten wird *Mobiles-Agenten-System (MAS)* genannt (siehe Abbildung 5 und vgl. [Morr98]).

Die Haupteinsatzmöglichkeiten von Mobilen Agenten werden derzeit im Bereich des elektronischen Handels (Electronic Commerce), des elektronischen Diensteverkehrs (Informationsbeschaffung in weitverteilten heterogenen Umgebungen), als Grundlage für die Anbindung mobiler Endgeräte sowie im Gebiet des komplexen Netzwerkmanagement gesehen. Nach [PhKa98] unterliegt dem Mobilen-Agenten-Ansatz auch die Idee, das Client/Server-Paradigma durch eine bessere, effizientere und flexiblere Art der Kommunikation zu ersetzen.

Da der Begriff Mobiler Agent zwei separate und verschiedene Konzepte, Mobilität und Agent, beinhaltet (vgl. [PhKa98]), werde ich diese beiden Konzepte im folgenden getrennt betrachten. Zunächst werden Software Agenten, ihre Eigenschaften und die Eigenschaften von Mobilen Agenten, Ablaufumgebungen und Agentenkommunikationssprachen vorgestellt.

### 1.1 Software Agenten - Ursprung und Taxonomie

Allgemein wird unter einem Agenten „ein im Auftrag eines anderen in dessen Interesse Tätiger“ [Broc84] verstanden. Im Bereich der Informatik existieren viele verschiedene Definitionen von *Agenten*, die nach Sichtweise und Einsatzgebiet sehr differieren (siehe Abbildung 1).

Die zweite Definition, die hier vorgestellt wird, stammt von [FrGr96]:

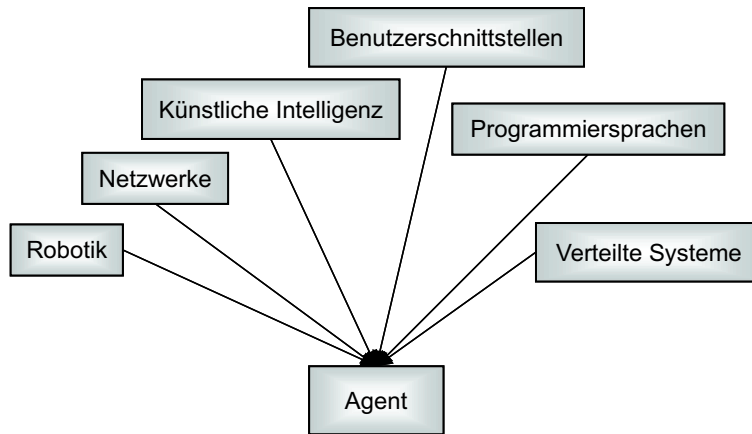


Abbildung 1: Agent: allgegenwärtiges Schlagwort

„Ein *Autonomer Agent* ist ein System, das sich in einem Teil einer Umgebung befindet, diese sich nutzbar macht, um in dieser über längere Zeit in Ausübung seines eigenen Plans hinweg so zu handeln, damit es in der Zukunft einen Nutzen davon tragen kann.“

Den Autoren gelingt eine genaue Definition von Autonomen Agenten dadurch, dass sie zwischen einem Agenten und einem Programm differenzieren. Nach [FrGr96] ist ein Agent nicht notwendigerweise ein Programm, er könnte auch Roboter oder Schullehrer sein. *Software Agenten* sind zwar per Definition Programme, müssen aber oben genannte Eigenschaften eines Agenten besitzen, um ein Agent zu sein.

[FrGr96] klassifizieren Autonome Agenten anhand eines biologischen Modells, der sog. *Systematik*. Diese Taxonomie führt auf die Form eines Baumes (siehe Abbildung 2):

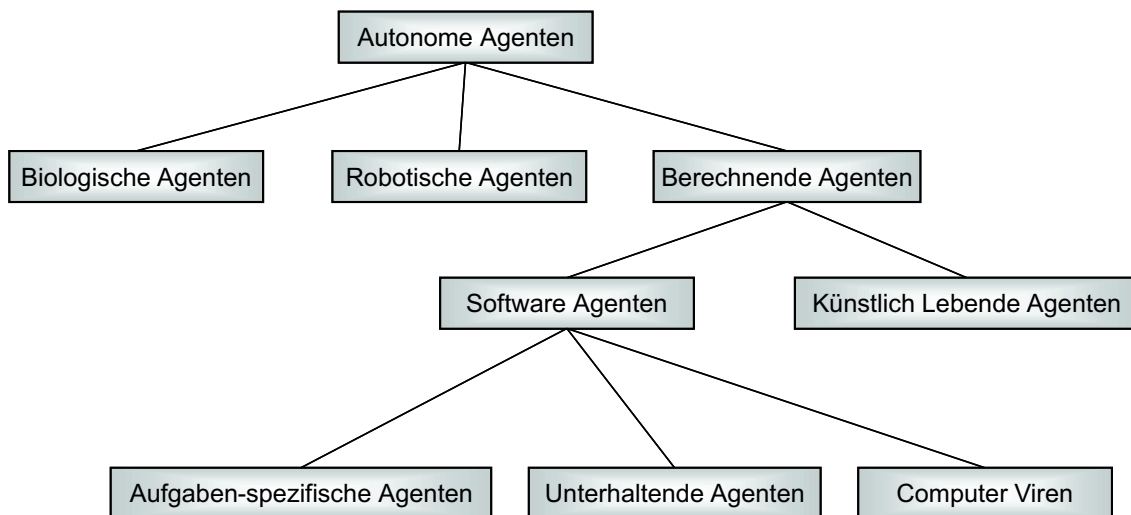


Abbildung 2: Systematik von Agenten

Die erste Unterscheidung treffen [FrGr96] indem sie Biologische Agenten (Biologic Agents), Robotische Agenten (Robotic Agents) und Berechnende Agenten (Computational Agents) unterscheiden, die natürlich vorkommende Arten darstellen. Jeder Mensch ist leicht in der Lage zwischen lebendigen Organismen, Artefakten und abstrakten Konzepten zu unterscheiden. Deshalb unterteilen [FrGr96] Berechnende Agenten in Software Agenten (Software Agents) und Künstlich Lebende Agenten (Artificial Life Agents). Auf der untersten Ebene der Systematik werden Software Agenten in Aufgaben-spezifische Agenten (Task-specific Agents),

Unterhaltende Agenten (Entertainment Agents) und Computer Viren (Computer Viruses) unterschieden.

Eine weitergehende Klassifikation ist anhand einer mathematischen Taxonomie möglich, die als *Matrix-Organisation* bekannt ist: Jede Charakteristik definiert eine Dimension. Mit n Charakteristika wird also eine n-dimensionale Matrix so aufgespannt, dass jede Zelle der Matrix mit einem Vektor von Charakteristika korrespondiert und so eine mögliche Kategorie für die Klassifikation darstellt. [Brad97] verwendet eine 3-dimensionale Darstellung der Charakteristika Intelligenz, Mobilität und Handlungskompetenz, wie in Abbildung 3 dargestellt.

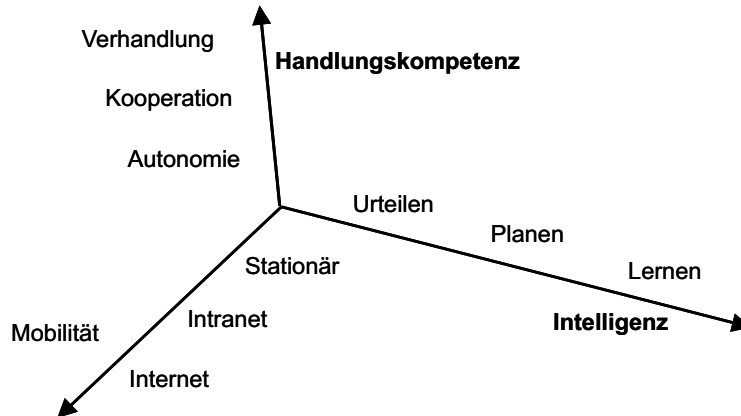


Abbildung 3: Klassifikation von Agenten nach [Brad97] als Matrix-Organisation

*Intelligente Software Agenten* haben ihren Ursprung in der Künstlichen Intelligenz (KI) der 50er Jahre. Damals wurde ein Agent als Software-Roboter definiert, der sowohl Vorlieben, unsicheres Wissen und Emotionen zeigen kann, als auch Urteilskraft, Planung und Lernfähigkeit besitzt. Forschungsarbeiten zu Mobilien Agenten kamen erst mit dem Siegeszug des Internet auf, also Anfang der 90er Jahre. Als *Intelligenten Software Agenten* definiert [WBWi98]

„eine Softwareentität, die für einen Benutzer bestimmte Aufgaben erledigen kann und einen gewissen Grad an Intelligenz besitzt, der ihr gestattet, ihre Aufgaben in Teilen autonom durchzuführen und dazu mit ihrer Umwelt auf sinnvolle Art und Weise zu interagieren.“

## 1.2 Allgemeine Eigenschaften von Agenten

Softwareagenten haben nach [Brad97], [WBWi98] und [Morr98] vier grundlegende Eigenschaften gemein:

- *Reaktiv*: Sie überwachen und reagieren in angemessener Art und Weise sowie in endlicher Zeit auf Umwelteinflüsse oder Informationen durch Sensoren und/oder durch ein internes Umweltmodell (deliberative Agenten).
- *Proaktiv*: Sie ergreifen in bestimmten Situationen selbständig die Initiative.
- *Zielorientiert*: Sie verfügen über wohldefinierte Ziele bzw. über ein komplexes Zielsystem und verfolgen diese bzw. richten ihre Handlungen nach diesem System aus.
- *Autonom*: Agenten arbeiten ohne direkte Intervention.

Desweiteren sind folgende Eigenschaften quantifizierbar:

- *Kommunikativ*: Agenten kommunizieren mit dem Anwender, anderen Agenten und der Umwelt. Zu diesem Zweck fungieren Agentenkommunikationssprachen über definierte Protokolle zum Austausch von Informationen.
- *Mobilität*: Agenten können sich selbst innerhalb elektronischer Kommunikationsnetzwerke bewegen.
- *Intelligenz*:
  - Wissensbasis*: Sie bildet die Grundlage um überwachte Ereignisse zu interpretieren.
  - Schlussfolgerungsfähigkeit*: Die Fähigkeit Schlussfolgerungen aus den Inhalten einer internen Wissensbasis zu ziehen. Dabei wird Rationalität der Schlussfolgerungen in dem Sinne vorausgesetzt, dass die Handlungen die Agenten einen Schritt näher an die Erfüllung eines Gesamt- oder Teilzieles bringen.
  - Lernfähigkeit*: die Agenten können sich den Änderungen der Umwelt anpassen, z.B. durch neuronale Netze oder Evolutionsmodelle.
- *Kooperation*: Die Zusammenarbeit mehrerer Agenten zur Lösung komplexer Aufgaben wird durch Agentenkommunikationssprachen ermöglicht, die den Austausch von Zielvorstellungen und Absichten erlauben.
- *Charaktereigenschaften*: Vertrauenswürdig, ehrlich und zuverlässig. Als virtuelle Person definiert kann ein Agent auch emotionale Zustände annehmen.

Weitere Unterscheidungen der Eigenschaften, wie z.B. interne und externe Eigenschaften, werden hier nicht getroffen. Es sei auch angemerkt, dass bisher keine einheitliche Klassifikation von Agenten existiert.

Ein Beispiel für eine agentenbasierte Anwendung ist der von der Firma FunArts für Microsofts Office 97/2000 entwickelte Assistent „Clippit“ (siehe Abbildung 4). Clippit ist ein agentenbasiertes Programm mit episodischem Gedächtnis ([Wagn97]).



Abbildung 4: Clippit von Office 2000

In Clippit wurden mehrere Konzepte aus der KI-Forschung realisiert: Beim Konzept des *Believable Agent* sollen Agenten durch einen glaubwürdigen audiovisuellen Ausdruck künstlicher Emotionen eine gefühlsbetonte Wirkung auf den Anwender ausüben. Das zweite Konzept ist die Verwendung eines wissensbasierten Benutzermodells, welches durch maschinelles Lernen ständig verfeinert und aktualisiert wird.

Mobile Agenten vereinen die vier Grundeigenschaften von Agenten, wie in Abschnitt 1.2 dargestellt, in sich und besitzen darüber hinaus per Definition die Mobilitätseigenschaft. [PhKa98] vergleicht Mobile-Agenten-Systeme untereinander mit ihren Charakteristika:

- *Sicherheit*: Die Sicherheit in einem Mobilen-Agenten-System muss für seine vier Komponenten getrennt betrachtet werden: Host-Rechner, Ablaufumgebung, Mobiler Agent und das Netzwerk-Subsystem das die Ablaufumgebungen, die auf verschiedenen Host-Rechnern errichtet sind, miteinander verbindet. Auf die Probleme der Sicherheitsaspekte und deren Lösungen wird in Abschnitt 3.2 eingegangen.
- *Portabilität*: Die Portabilität ist durch die Heterogenitätseigenschaft gekennzeichnet, damit ein Mobiles-Agenten-System auf multiplen Plattformen agieren kann. In der Zeit vor Java, basierten die Systeme auf betriebssystemspezifischem Kern und sprachenspezifischen Interpretern (sog. *Core-Systeme*), was Leistungs- und Skalierprobleme nach sich zog. Java-basierte Systeme dagegen bringen Betriebssystemneutralität, Leistungsverbesserungen (aufgrund der Just-in-Time Kompilation) sowie Skalierbarkeitsverbesserungen mit sich.
- *Mobilität*: Auf die verschiedenen Arten von Mobilität wird in Abschnitt 2 eingegangen.
- *Kommunikation*: In Bezug auf Agenten existieren zwei verschiedene Modelle der Kommunikation:

Die Kommunikation der Agenten untereinander

Die Kommunikation zum Benutzer, d.h. die Kontrolle der Agenten betreffend (vgl. [PhKa98]).

Java-basierte Systeme unterstützen meist Ereignis-, Nachrichten- und/oder Remote Procedure Call (vgl. Abschnitt 2), während Core-Systeme nur Client/Server-Arten der Kommunikation unterstützen. Die Sprachenabhängigkeit, die sich durch ein Java-basiertes System ergibt, könnte nach [PhKa98] durch CORBA gelöst werden. Mit CORBA kann man eine den Java-basierten Systemen ähnliche Architektur in sprachen- und plattformunabhängiger Form bereitstellen.

- *Ressourcenmanagement*: Das Ressourcenmanagement der Core-Systeme ist von den Entwicklern nicht genau diskutiert. Bei Java basierten Systemen übernimmt die Java Virtual Machine (JVM) das Ressourcenmanagement.
- *Ressourcen-Verfügbarkeit*: Die Idee, die dieser Charakteristika zugrunde liegt ist die, dass ein Mobiler Agent sich bewusst ist, dass die Seite, zu der er migriert, auch momentan erreichbar ist. Dieser Ansatz wurde jedoch noch in keinem Mobilen-Agenten-System realisiert.
- *Identifikation*: Um Kontrolle, Kommunikation, Kooperation und Koordination von Mobilen Agenten zu ermöglichen, müssen sie einzeln in ihrem Umfeld identifiziert werden können. Die Realisierung in den Mobilen-Agenten-Systemen reicht nach [PhKa98] von Alias-Namen (Voyager), einfachem Tabellen-Nachschlagen (Odyssey) über Directory-Manager (Concodria) bis hin zu einem Agent-Execution-Control-Server Arrangement auf globaler Ebene (Ship-MAI).
- *Kontrolle*: Mobile Agenten zu kontrollieren heisst sie zu kreieren, zu starten, zu duplizieren und sie anzuweisen sich selbst zu terminieren. Diese sehr wichtige Kontroll-Charakteristika wird von allen Systemen bei einer kleinen Anzahl von isolierten Agenten unterstützt. Die Probleme bei einer grösseren Anzahl von Agenten liegen in der Skalierbarkeit der Kontrolle (vgl. [PhKa98] zum Beispiel bei Voyager Space oder Aglet Message).
- *Datenmanagement*: Das Datenmanagement umfasst das Verwalten von Daten, die ein Mobiler Agent mit sich führen kann, um seine Aufgabe zu erledigen, als auch das speichern seiner selbst in einer persistenten Form. Diese Dienste werden von allen aktuellen Mobilen-Agenten-Systemen unterstützt.

### 1.3 Ablaufumgebungen

Alle Agenten können nur innerhalb sogenannter *Agentenplattformen* existieren. Die Agentenplattform (siehe [Abbildung 5]) ermöglicht überhaupt erst die Ausführung der Agenten, das heisst die Agentenplattform bildet den Rahmen, in dem sich Agenten bewegen können. Sie stellt die Orte zur Verfügung, an denen sich die Agenten aufhalten können. Desweiteren ermöglicht sie den Nachrichtenaustausch zwischen den Agenten, stellt den Mechanismus zur Verfügung sowie diverse andere, teilweise individuelle Dienste, die von den Agenten genutzt werden können. Sie regelt und ermöglicht außerdem den Zugriff auf Systemressourcen.

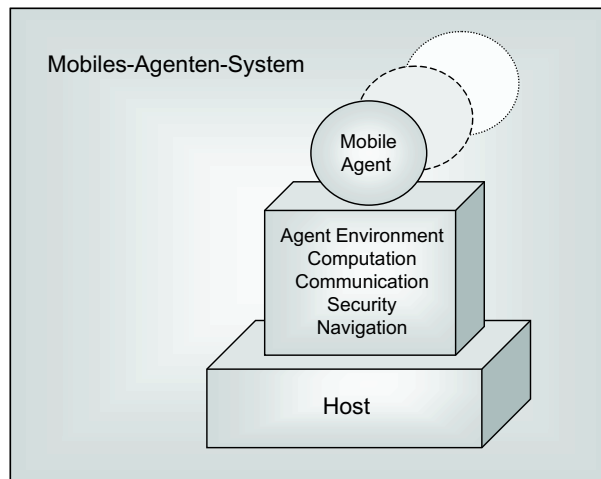


Abbildung 5: Agentenplattform

Die Architektur der neueren Java-basierten Ablaufumgebungen basiert auf *Schichten (layered architecture)*. Die einzelnen Schichten übernehmen dabei spezifische Aufgabenstellungen. Zu ihrer Lösung benötigen sie die Dienste niedrigerer Schichten. Durch eine hierarchische Anordnung der Schichten steht in der obersten Schicht die Gesamtfunktionalität des Systems zur Verfügung (vgl. [KrRe00]). Mobile Agenten Systeme (siehe [Abbildung 6]) machen sich

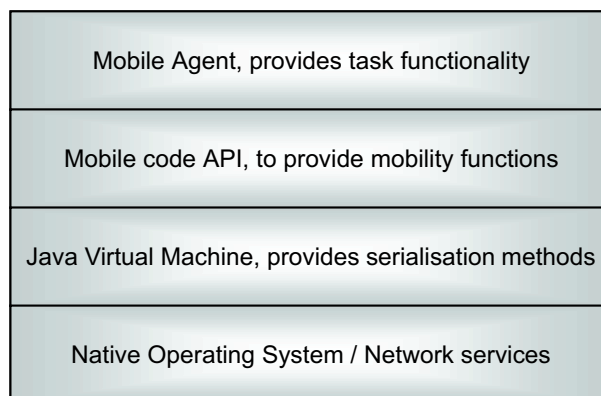


Abbildung 6: Schichtenarchitektur Mobiler-Agenten-Systeme und deren Anwendungen

die untersten Schichtendienste - die Möglichkeiten des Betriebssystems eines Host-Rechners - zu Nutze, indem sie Code und Daten über das Netzwerk versenden. Die nächste Schicht führt Programmcode aus und ist typischerweise eine virtuelle Maschine (virtual machine), die die Betriebssystemunabhängigkeit garantiert. Über dieser stehen die Bibliotheksklassen (class libraries) der Anbieter von Mobilen-Agenten-Systemen, die Mobilitätsfunktionen, wie z.B. bereitstellen. Die Mobilen Agenten letztendlich werden auf Basis die-

ser Mobilitätsfunktionen entworfen, um Management-Funktionen auf hohem Niveau für die Steuerung und das Bearbeiten von Aufgaben bereitzustellen.

Da "Agentenplattformen" das Thema eines anderen Seminarbeitrags darstellen (vgl. ???), werde ich an dieser Stelle nicht weiter darauf eingehen.

## 1.4 Agentenkommunikationssprachen

Agentenkommunikationssprachen sind Sprachen, in denen sich Agenten untereinander unterhalten können (siehe Abbildung 7).

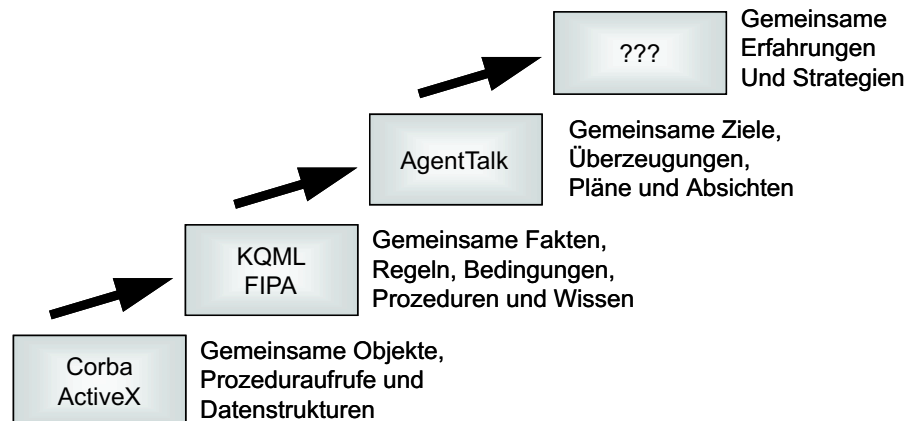


Abbildung 7: Agentenkommunikation (KI)

Die unterste Ebene dieser Kommunikation stellt dabei der Austausch von Objekten, und entfernte Prozedur-/Methodenaufrufe dar. Beispiele von Sprachen aus der darüberliegenden Ebene sind die aus der Forschung zur Künstlichen Intelligenz (KI) stammenden Sprachen KQML (Knowledge Query and Manipulation Language) und FIPA-ACL (Foundation for Intelligent Physical Agents - Agent Communication Language). In diesen Sprachen sind gemeinsame Fakten, Regeln, Bedingungen, Prozeduren und Wissen formulierbar. Der Nachrichtentypen-Katalog von KQML ist Resultat eines Standardisierungsprojektes (Knowledge Sharing Effort) das von US-amerikanischen Universitäten getragen die Kommunikation zwischen wissensbasierten Systemen zum Hauptanliegen hatte (vgl. [Wagn97]). Auf der obersten Ebene ermöglichen Sprachen wie AgentTalk darüber hinaus, gemeinsame Ziele, Überzeugungen, Pläne und Absichten untereinander auszutauschen. Die meisten Mobilen-Agenten-Systeme bieten nur Agentenkommunikation auf der untersten der beschriebenen Ebenen, da bei den Konzepten der Mobilen-Agenten-Systeme meist der Aspekt eine Architektur für Verteilte Systeme zu sein im Vordergrund steht und weniger der „KI-Aspekt“.

## 2 Mobilität

Mobile Agenten traten aus dem Schatten der KI-Forschung und vereinen Telekommunikations-, Software- und Verteilte-Systeme-Technologien um neue automatisierte Lösungen zu bieten (vgl. [Morr98]). Dies wird vor allem durch die Mobilitätseigenschaft der Agenten ermöglicht, aufgrund derer sie von einem Computer zu einem anderen migrieren können. Auf diese Eigenschaft und ihre Entwicklung in der Informatik werde ich in diesem Abschnitt genauer eingehen. Nach [RoHR97] lassen sich vier Arten von Mobilität unterscheiden:

- Remote Execution

- Code on Demand
- Weak Migration
- Strong Migration

Bei *Remote Execution* wird der Code des Agenten auf einen entfernten Zielort transferiert, um dort ausgeführt zu werden (auch *Prozess-Migration* genannt). Der Transfer des Agenten wird von dem Ort initiiert, auf dem sich der Code befindet. Während der gesamten Ausführung verbleibt der Agent dann auf dem Zielort. Eine typische Anwendung dieser Art sind Java-Servlets (vgl. [Sun97]).

*Code on Demand* entspricht im wesentlichen dem Verfahren der Remote Execution. Der Unterschied besteht darin, dass hier die Initiative vom Empfänger des mobilen Codes ausgeht, er „fordert den Code an“. Diese Art der Mobilität wird zum Beispiel von der Anwendung ActiveX (vgl. [AaAa97]) und von Java-Applets benutzt.

Sowohl bei Remote Execution, als auch bei Code on Demand wird nur der Code des Agenten zum Zielort transferiert. Die Ausführung des Agenten wird erst auf dem Ziel-Host-Rechner begonnen. Während seiner gesamten Ausführung ist der Agent stationär an den Host gebunden. Da in beiden Ansätzen nur der Code des Agenten transferiert wird, spricht man hier von *Code Mobility*.

Wird beim Transfer neben dem Code des Agenten noch sein aktueller Zustand mitgeführt, so spricht man von *Agent Mobility*: Zunächst wird hierzu der Zustand des Agenten ermittelt. Danach werden Code und Zustand des Agenten transferiert. Ist der Agent am Zielort angekommen, wird sein Zustand rekonstruiert und seine Ausführung fortgesetzt. Diese Methode bietet dem Agenten die Möglichkeit, beliebig oft während seiner Ausführung den Ort zu wechseln. Auch bei diesem Ansatz unterscheidet man zwei verschiedene Arten von Mobilität.

Bei einer *Weak Migration* wird neben dem Code des Agenten lediglich der Zustand seiner Daten transferiert. Es liegt hierbei in der Verantwortung des Programmierers, die Variablen zu selektieren, die transferiert werden sollen und aus denen der Ausführungszustand des Agenten nach der Migration rekonstruiert werden soll. Ferner muss vom Programmierer dafür Sorge getragen werden, dass der Agent nachdem er am Zielort angekommen ist, wieder in seinen Ausführungszustand versetzt wird. In Java-basierten Mobilen-Agenten-Systemen wie Aglets oder Mole wird die Methode Weak Migration verwendet (vgl. [PhKa98]).

Den höchsten Grad an Mobilität kann man durch *Strong Migration* erreichen. Hier wird der komplette Zustand des Agenten transferiert. Die Fixierung des Zustandes, der Transport und die Wiederherstellung des Agenten wird vom zugrundeliegenden System geleistet und ist für den Programmierer transparent. Beispiele für Systeme, die Strong Migration realisiert haben sind die Core-Systeme wie Ara, Telescript oder AgentTcl (vgl. [PhKa98]).

Streng genommen wird *Agent Mobility* nur durch das Konzept der *Strong Migration* erreicht. Die komplette Beschreibung des Zustandes eines Agenten kann jedoch sehr gross werden. Dadurch wird der Vorgang der Migration zu einer sehr zeitaufwendigen und teuren Operation. Dies rechtfertigt die Benutzung des Ansatzes der *Weak Migration*, bei dem die zu transferierende Zustandsinformation limitiert werden kann. Tabelle 1 fasst die verschiedenen Arten der Mobilität nochmals zusammen.

Mobile-Agenten-Systeme unterscheiden sich von Prozess-Migrations-Systemen dadurch, dass sie selbst entscheiden wann sie migrieren, typischerweise durch einen „ “ oder „ “ Befehl (vgl. [KoGr99]). Im Gegensatz zu Prozess-Migrations-Systemen, bei denen das System entscheidet, wann und wohin der aktuell laufende Prozess migriert wird (typischerweise um CPU-Auslastungen zu verteilen).

		Code Mobility		Agent Mobility	
		Remote Execution	Code on Demand	Weak Migration	Strong Migration
Transport of	Code	X	X	X	X
	Data State			X	X
	Execution State				X

Tabelle 1: Arten von Mobilität

Das Mobile-Agenten-Konzept entstand nach [Morr98] aus drei früheren Technologien: *Process Migration (Process Migration)*, *Entfernte Auswertung (Remote Evaluation)* und den *Mobilen Objekten (Mobile Objects)*. Diese Technologien wurden entwickelt um den *Entfernten Prozedur Aufruf (Remote Procedure Calling)* für das *Verteilte Programmieren (Distributed Programming)* zu verbessern (siehe Abbildung 8).

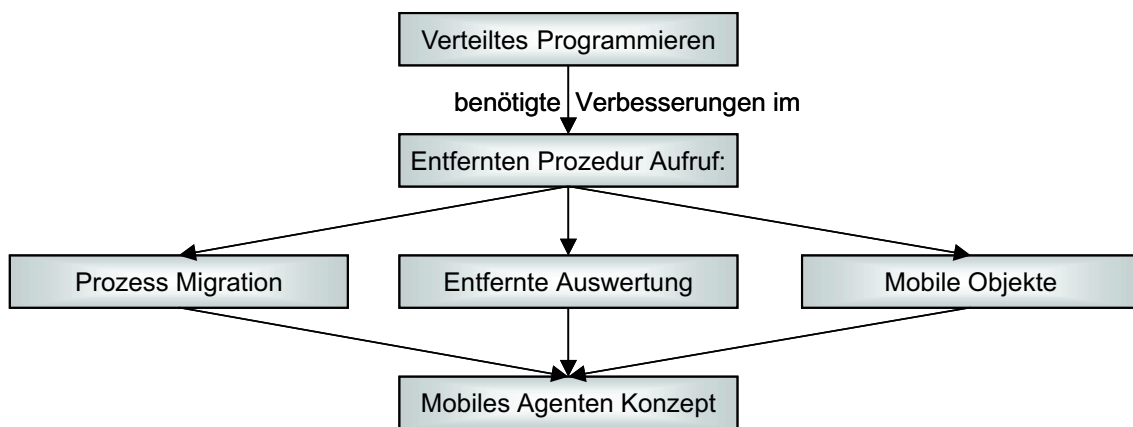


Abbildung 8: Entstehung des Mobilen-Agenten-Konzepts

### 3 Vorteile und Problemstellungen

Nachdem nun die zwei grundlegenden Konzepte Mobiler Agenten, Mobilität und Agent, vorgestellt wurden, werde ich nun die Vorteile und Problemstellungen diskutieren, die sich aus dem Ansatz der Mobilen Agenten ergeben.

#### 3.1 Merkmale, Vorteile und deren Nutzen

Neben den unter Abschnitt 4 genannten Vorteilen fassen D. Lange und M. Oshima in [LaOs99] sieben Hauptgründe für Mobile Agenten zusammen:

- *Mobile Agenten reduzieren die Netzwerklast:* Verteilte Systeme obliegen oft auf Kommunikationsprotokollen, die viele Interaktionen mit sich bringen um eine gegebene Aufgabe zu erfüllen. Das Resultat ist eine große Menge an Netzwerk-Datenverkehr. Mobile Agenten erlauben es ihren Benutzern eine Konversation „einzupacken“ und sie zum Ziel-Host zu schicken um dort lokal die Interaktionen auszuführen, wie in Abbildung 9 dargestellt. Mobile Agenten sind auch nützlich um den Fluss an Daten in einem Netzwerk zu reduzieren. Wenn sehr grosse Mengen an Daten bei entfernten Host-Rechnern gespeichert

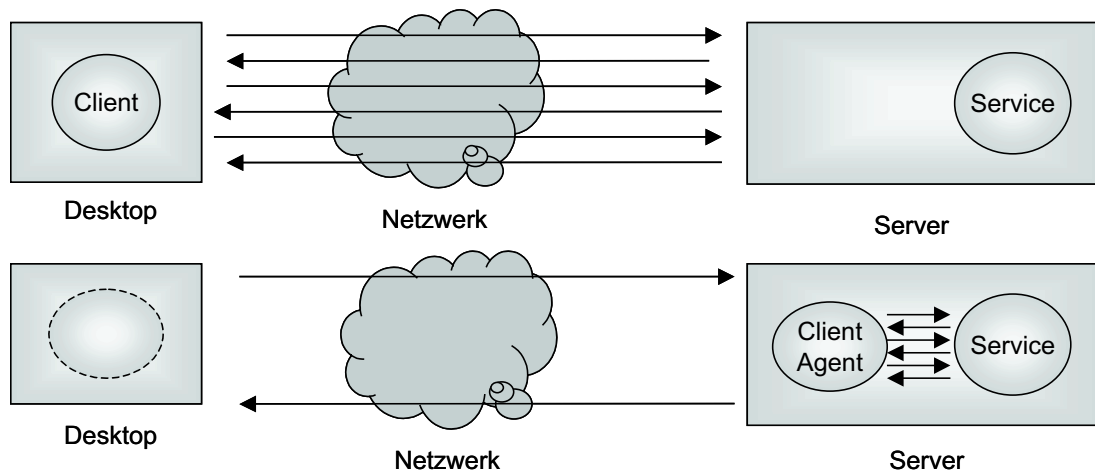


Abbildung 9: Agentenbasierter Ansatz

werden müssen, sollten diese Daten lokal verarbeitet und nicht über das Netzwerk transferiert werden. Das Motto für Agenten-basierte Datenverarbeitung ist einach: Bring das Programm zu den Daten und nicht die Daten zum Programm.

- *Überwindung von Netzwerk-Latenzen:* Kritische Echtzeit-Systeme, wie Robotersteuerungen in Fabriken, müssen in Echtzeit auf Veränderungen ihrer Umgebung reagieren. Um solche Systeme zu kontrollieren, können hohe Netzwerk-Latenzen nicht akzeptiert werden. Mobile Agenten stellen hier eine Lösung dar, indem sie von einem zentralen Steuerzentrum abgeschickt werden um die Befehle des Controllers lokal direkt auszuführen.
- *Einkapseln von Protokollen:* Wenn in einem Verteilten-System Daten ausgetauscht werden, muss auf jedem Host-Rechner ein Programm installiert sein, das die Protokolle umsetzt, die benötigt werden, um ausgehende Daten zu codieren und um eingehende Daten interpretieren zu können. Jedoch müssen Protokolle aufgrund von steigenden Effizienz- und Sicherheitsanforderungen angepasst werden. Da dies oft nur sehr schwer, wenn nicht gar unmöglich ist, stellen Mobile Agenten eine Lösung dar, indem sie vom entfernten Host-Rechner aus einen „Kanal“ auf Basis von bereits vorhandenen Protokollen aufbauen.
- *Programme asynchron und autonom ausführen:* Mobile Endgeräte haben meist nur eine teure oder schwache Netzwerkverbindung zur Verfügung. Anwendungen, die eine kontinuierliche Verbindung zum Anwender benötigen, sind dann möglicherweise nicht ökonomisch oder technisch realisierbar. Um dieses Problem zu lösen kann eine solche Anwendung in einen Mobilen Agenten eingebettet werden, welcher anschliessend ins Netzwerk entsendet wird. Nachdem der Agent entsendet ist, kann er unabhängig vom Prozess, der ihn erschaffen hat, existieren und asynchron und autonom operieren. Das Mobile Endgerät kann seine Verbindung trennen und zu einem späteren Zeitpunkt den Agenten wieder „einsammeln“ (siehe auch Abbildung 9).
- *Dynamische Anpassungsfähigkeit:* Mobile Agenten können ihre Ablaufumgebung beeinflussen und autonom auf Veränderungen reagieren. Multiple Mobile Agenten haben sogar die einzigartige Fähigkeit sich selbst unter Host-Rechnern im Netzwerk zu verteilen um eine optimale Konfiguration zu bieten, damit ein Problem gelöst werden kann.
- *Natürliche Heterogenität:* Netzwerk-Technologie ist fundamental heterogen, sowohl von Hardware- als auch von Softwareperspektive aus. Da Mobile Agenten generell computer- und transportunabhängig sind (abhängig sind sie nur von ihren Ablaufumgebungen), bieten sie die optimalen Bedingungen für eine nahtlose Systemintegration.

- *Robustheit und Fehlertoleranz:* Die Fähigkeit Mobiler Agenten auf unvorhersehbare Situationen und Ereignisse dynamisch zu reagieren, erlaubt es robustere und fehlertolerantere Verteilte-Systeme zu schaffen. Wenn ein Host-Rechner heruntergefahren wird, werden alle Agenten, die auf diesem Host-Rechner operieren gewarnt und ihnen wird Zeit gegeben, ihre Operationen auf einem anderen Rechner fortzuführen.

## 3.2 Problembereiche

Bevor die Vision des Einsatzes Mobiler Agenten im kommerziellen Umfeld stattfinden kann, müssen noch viele Probleme gelöst werden. [KoGr99] unterscheidet in technische und nicht technische Hürden.

Folgende Problembereiche lassen sich als technische Hürden identifizieren:

Welche zusätzlichen Funktionen müssen bereitgestellt werden um Agentenwelten effizient implementieren zu können ohne dabei wieder vom Betriebssystem abhängig zu werden? Zum Beispiel das Bereitstellen von Ressourcen, Abrechnungen, Dienstgütegarantien, ...

Agentenplattformen sind, ähnlich wie Middleware (CORBA, OMA, DCE), Plattformen, die grundlegende Agentenweltfunktionalitäten zur Verfügung stellen. Sie dienen dazu, Agentenwelten einfach erstellen zu können. Damit sind neben universellen auch spezialisierte Agentenwelten implementierbar. Das Hauptproblem heutiger Agentenplattformen ist ihre Uneinheitlichkeit. Agenten unterschiedlicher Mobiler-Agenten-Systeme können nicht auf Host-Rechner eines anderen Mobilen-Agenten-Systems migrieren, geschweige denn miteinander kommunizieren. Hauptforschungsschwerpunkt wird hierbei das Identifizieren von Bausteinen sein, sowie die Festschreibung von Standards. So sind zum Beispiel Bausteine in verschiedenen Ausprägungen von Agentenmigration, Sicherheitsverfahren oder aber Agentenkommunikation denkbar.

Agentenwelten dienen den Agenten als eigentlicher Aufenthaltsort, und müssen folgende Kernfunktionalitäten unterstützen:

- *Heterogenität:* Agenten müssen in einer Sprache formuliert werden, die dann auf allen Rechnern ausgeführt werden kann. Diese Forderung in Verbindung mit Sicherheitsaspekten haben sehr schnell zu der Einsicht geführt, daß Agenten in einer interpretierten Sprache zu erstellen sind. Noch ist keine deutliche Entscheidung für eine der existierenden interpretativen Sprachen gefallen. Zudem wird an neuen Agentensprachen gearbeitet. Verwendung finden u.A. safe-tcl(PD), Java (SUN), und sogar Lisp. Jüngere Systeme basieren jedoch fast ausschliesslich auf Java. Durch den interpretativen Ansatz lassen sich Zugriffe auf sicherheitsrelevante Systemressourcen schon durch den Interpreter reglementieren.
- *Migration* der Agenten: Existierenden Projekte lassen hier im wesentlichen zwei Verfahren erkennen: Migration des Agentencodes auf niedrigem und hohem Niveau. Der Vorteil des ersten Verfahrens besteht darin, dass die Migration für den Agentencode transparent geschieht, die zweite Variante ist einfacher, da der Agent am neuen Ort einfach wieder von Beginn ausgeführt wird.

- *Sicherheit* der Agenten und des Wirtssystemes: Die Forschungen stehen noch am Anfang, wenn es um den Schutz des Agenten vor dem Wirtssystem geht. Kann das Wirtssystem noch durch geeignete Abschottung des verwendeten Interpreters für die Agentensprache gewährleistet werden, und lassen sich durch Public-Key Verfahren Authentizität und Integrität des ins Wirtssystem transferierten Agentencodes überprüfen und gewährleisten, so existieren bisher lediglich nicht verwirklichte Ansätze, welche den Agenten etwa vor „böartigen“ Wirtssystemen schützen. [KaAG98] z.B. haben ein System entwickelt, das es dem Wirtssystem ermöglicht zu erkennen, ob der Agent „befallen“ wurde und das zum anderen auf einem geheimen kleingehackten Kettenprotokoll (secret hash chain protocol) basiert um Attacken abzuwehren.

Das Hauptproblem, das Mobile Agenten mit sich bringen ist, dass sie sich in ihrer Architektur und der Form ihres Programmiercodes nicht wesentlich von Computer Viren unterscheiden, weshalb viele Verwalter von Host-Rechnern den Mobilien Agenten kritisch gegenüber stehen. Folglich sind Lösungen für Mobile Agenten im Bereich der Sicherheit unbedingt erforderlich (vgl. ???).

- *Kommunikation und Kooperation* zwischen Agenten: Agenten sollen untereinander kommunizieren können, um Informationen auszutauschen. Dies soll sich allerdings nicht nur darauf beschränken, dass zum Beispiel zwei Agenten mit einem ähnlichen Auftrag Informationen über die jeweils bekannten Dienste austauschen, sondern gerade im ange-dachten elektronischen Markt von Abbildung 11 sind dynamische Vermittlungsprotokolle zwischen Händler-Agenten und deren „Kunden“ (Benutzer-Agenten) gefordert. Im Bereich der Agentenkooperation sind Kooperationsprotokolle nötig, die auch zwischen Agenten funktionieren, die in verschiedenen Agentensprachen formuliert sind, und somit u.U. verschieden mächtige Kommunikationsmechanismen aufweisen.

Die KI-Ansätze im Bereich „Agenten“ - der sogenannten Intelligenten Agenten - gehen traditionell in eine völlig andere Richtung. Hier sind Problembereiche wie:

- Problemlöseverfahren (auch verteilt)
- Selbstlernende Agenten
- Mensch-Maschine-Unterstützung
- Wissensrepräsentation
- Kooperationsstrategien

angesiedelt. Allerdings deuten neuere Forschungen zum Beispiel im Bereich WWW (Intelligente Informationssuche) darauf hin, dass eine Zusammenarbeit der beiden Lager zukünftig zu erwarten ist.

Nach [KoGr99] existieren zudem noch nicht-technische Probleme:

- Fehlen einer *Killer-Applikation*: Alle Forscher im Gebiet der Mobilien Agenten sind sich einig: Es fehlt die Killer-Applikation. [LaOs99] drücken es noch drastischer aus: „ Es gibt keine Mobile-Agenten-Anwendungen, aber es gibt eine Menge an Anwendungen, die davon profitieren, das Paradigma der Mobilien Agenten zu benutzen“.
- Der natürliche Weg der Software-Evolution muss eingehalten werden. Das direkte Umschalten von existierenden Client/Server Systemen auf volle Mobile-Agenten Technologie würde kein Internet Service Provider mitmachen. Vielmehr müsste der Weg zum Erfolg der Technologie über lokale Netze, dem Intranet und schliesslich zum Internet führen.



- Ein Agent wird vom PDA aus in ein Warenhaus geschickt um Sonderangebote zu finden.
- Ein Arbeiter hat verschiedene Aufgaben in unterschiedlichen Gebäuden einer Fabrik zu erledigen. Sobald er ein Gebäude betritt springt ein Mobiler Agent in seinen PDA und assistiert ihn bei seiner Aufgabe, z.B. kritische Punkte in einem Kraftwerk zu finden und zu untersuchen.

Während das hoch-breitbandige „Backbone“-Netzwerk des Internet dramatisch wachsen wird (Bsp.: Nortel, die ein 1,6 Tbps optisches Glasfaserprodukt noch für 2000 ankündigten), sind doch die lokalen Bandbreiten der Endbenutzer sehr viel kleiner. Dies hat verschiedene Gründe. Zum einen sind die Kosten für eine 10 Mbps Verbindung (z.B. via Kabel-Modem) sehr hoch und zum anderen sind die Voraussetzungen für schnelle Internet-Verbindungen durch „normale“ Modems mit Kupferleitungen (z.B. ADSL als momentan schnellstes Produkt) und durch kabellose Netzzugänge nicht gegeben.

Anwendungen werden auch im Dynamischen Netzwerkmanagement gesehen (vgl. [Morr98]): Ein Mobiler Agent würde fähig sein die Netzwerk-Auslastung über einen 24 Stunden Zyklus zu lernen, indem der Datenverkehr wächst, sein Maximum erreicht und wieder fällt. Falls ein Mobiler Agent ein solches Maximum, das auch noch bestehen bleibt, an Datenverkehr im Netzwerk zu einer untypischen Zeit erkennt, könnte er mit anderen Agenten kooperieren um herauszufinden, ob ein Teil des Netzwerkes ausgefallen ist um dann den Datenverkehr umzuleiten und an die Netzwerk-Verantwortlichen zu berichten. Ebenso könnten Mobile Agenten neue Ressourcen im Netzwerk erkennen und für den Netzwerkdatenverkehr optimal auslasten.

[Morr98] sieht für Mobile Agenten auch Anwendungsmöglichkeiten bei sehr grossen Echtzeit-Applikationen, einschließlich der Fabrikation, und bei Distance-Learning Applikationen. Im Multimedia-Bereich könnten Mobile Agenten Lösungen für die Ende-zu-Ende *Dienstgüte* bieten, die die Lieferung von Multimedia-Materialien verbessern würde.

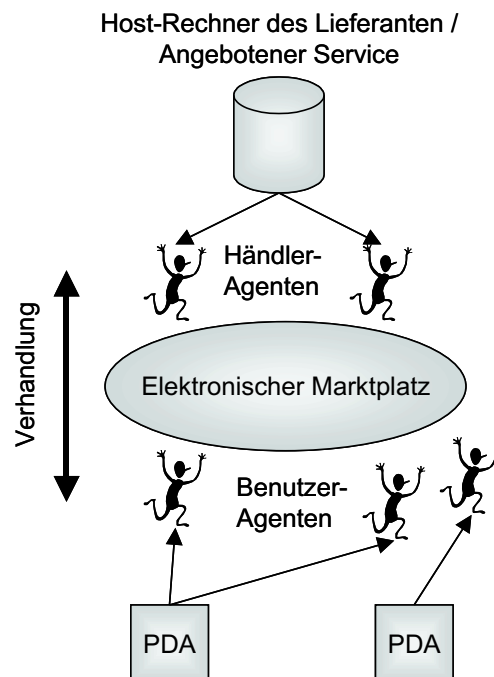


Abbildung 11: Mobile Agenten im elektronischen Marktplatz der Zukunft

In [WoPM99] wird ein mögliches Szenario für Mobile Agenten im Bereich des eCommerce vorgestellt. In einer Agentenwelt sind Spezialagenten, sogenannte *Händler-Agenten*, angesiedelt, die der Vermittlung von Diensten, wie sie durch entsprechende Host-Rechner angeboten

werden und von Dienstnachfragern, den Benutzer-Agenten, in Anspruch genommen werden, dienen. Die Vermittlung der Dienste, also die Verhandlung der Agenten untereinander erfolgt hierbei über die in Abschnitt 1.4 vorgestellten Agentenkommunikationssprachen. In diesem Szenario (vgl. das Anwendungsbeispiel TabiCan in Abschnitt 5) kann man zurecht von der Realisierung eines „elektronischen Marktes“ sprechen (siehe Abbildung 11).

## 5 Anwendungsbeispiele

Es existiert zwar keine „Killerapplikation“ im Bereich der Mobilien Agenten die ihnen zum grossen Durchbruch verhelfen könnte (vgl. [LaOs99]), was nach Meinung der Experten in [MJKo99] jedoch auch nicht zwingend notwendig sei. Alle Lösungen, die man mit Mobilien Agenten entwickelt, könnte man auch mit herkömmlicher Technologie lösen. Die Autoren sehen in Mobilien Agenten daher vielmehr einen einheitlichen Weg, Problemstellungen zu lösen, als eine Technologie, die Neues ermöglicht.

Es existieren jedoch einige Anwendungen, die davon profitieren, Mobile Agenten einzusetzen. Einige von ihnen werde ich hier nun vorstellen. Die Anwendungsbeispiele erstrecken sich über die in Abschnitt 4 vorgestellten Einsatzmöglichkeiten und umfassen Mobile Computing, Netzwerkmanagement, eCommerce und ein Lieferungs- und Managementsystem von verteilten Parallelen Prozessen (*Delivery and Management System for Distributed Parallel Processing*).

### 5.1 Magic Link und PersonaLink (Sony und AT&T) - Telescript von General Magic

Der Einsatz Mobiler Agenten im Mobile Computing war bereits im ersten Mobilien-Agenten-System, dem Telescript-System (1994) eines der anvisierten Anwendungsgebiete. General Magic fand Partnerfirmen für dieses Vorhaben, unter anderem AT&T, Sony und Motorola.

Sony entwickelte mit Magic Link ein tragbares Telescript-fähiges Endgerät (siehe Abbildung 12).



Abbildung 12: Magic Link von Sony

Magic Link hatte eine über einen berührungsfähigen Bildschirm steuerbare grafische Benutzungsoberfläche (Magic Cap). Zubehör wie PCMCIA Pager Card, Headset zum Telefonieren, externe Tastatur und PC-Anbindung rundete die Produktpalette ab. Das Gerät enthält u.a. Anwendungen für Terminplanung, Adressverwaltung und E-Mail. In Japan sollen ca. 1000 dieser Geräte im Einsatz sein.

AT&T lieferte mit den PersonaLink Services die Netzinfrastruktur. Der PersonaLink Service war der erste WAN (Wide Area Network) Service, der intelligente Mobile Agenten im Netzwerkmanagement umsetzte. In diesem sog. „*Smart-Network*“ stellt das Netzwerk nicht mehr nur eine „Röhre“ dar, an deren Enden Applikationen ausgeführt werden, sondern es wird vielmehr zu einem virtuellen Host-Rechner für verteilte Applikationen innerhalb und außerhalb

des Netzes, bzw. Peripherie eines grossflächigen Services. Das Endbenutzer-Gerät, das mit dem Smart-Network verbunden ist, kann beliebig sein: einfach oder state-of-the-art, stationär oder mobil (z.B. Magic Link). Das Netzwerk passt sich automatisch an das Zugangsgerät an und somit sind Benutzer und Anwendungs-Entwickler nicht in das komplexe Netzwerkmanagement verwickelt.

Anstatt, nur auf die Echtzeit-, Verbindungs-orientierten Sitzungen zu bauen, machen Smart-Networks umfangreichen Gebrauch vom Speicher- und Weiterleitungsmechanismus der Agentenplattform. Sie bewirten die Mobilen Agenten, die sich durch das Netzwerk bewegen um Nachrichten zu einem Benutzer zu lotsen oder Informationen und Dienste in Vertretung des Benutzers zu filtern. Das System verwirklicht somit den Vorteil Mobiler Agenten, ihre Benutzer ständig zu vertreten, selbst wenn sie schlafen (vgl. [Rein94]).

PersonaLink konnte sich am Markt nicht durchsetzen. Jedoch hat sich *Oracle* ebenfalls mit der Entwicklung eines Mobilen-Agenten-Systems beschäftigt. Das Produkt, das momentan auf dem Markt ist - *Oracle Mobile Agent Release 3.0* vgl. [Ora] - , erlaubt eine schnelle Implementation eines *Smart-Networks* in einem LAN. Das System erlaubt Zugriff auf gemeinsame Informationsressourcen für eine mobile Arbeitskräfte. Die Hauptaspekte des Systems sind:

- *Mobilität*: Die Fähigkeit überall zu kommunizieren mit dem angemessensten Kommunikationskanal. Möglich ist eine Anbindung von Funknetz-Technologien, schnurgebundene oder schnurlose Einwahl durch ein Modem und LAN-Verbindungen.
- *Client/Agent/Server-Architektur*: Eine Erweiterung der Client/Server-Architektur für Netzwerke, bei denen Bandbreiten begrenzt sind und Kommunikations-Verbindungen begrenzt bzw. unzuverlässig sein können.

## 5.2 TabiCan - Aglets von IBM

Um die Effizienz und die Vorteile von Aglet - einem Java-basierten Mobilen-Agenten-System von IBM - zu demonstrieren, entwickelt das *Tokyo Research Laboratory* von *IBM* eCommerce-Applikationen. Im August 1998 startete IBM Japan einen elektronischen Marktplatz, auf dem verschiedene Verkaufsagenten für Reisebüros zur Verfügung gestellt wurden, die Reisen online verkaufen: *TabiCan* - [Tab]. [Fisc99] berichtet, dass der Name „TabiCan“ zum einen aus dem Japanischen „tabi“ (=reisen) und zum anderen aus dem Englischen „can“ von dem Ausdruck „can do“ kommt.

TabiCan kann man sich heute als eine virtuelle Gemeinschaft vorstellen (Wolke in Abbildung 13), die elektronische Marktplätze offeriert, z.B. einen Marktplatz für Fluglinien, einen für Hotels, einen für Mietwagenfirmen, einen für Restaurants und einen für Reiseagenturen (e-Marketplace Server in Abbildung 13). Firmen können in jedem Marktplatz ihren eigenen *Verkäuferagenten (Shop Agent)* kreieren, der ihre Produkte anbietet (rote und blaue Biene in Abbildung 13). Betritt ein Konsument die Plattform von TabiCan, so wird, nach Eingabe der benötigten Informationen, wie zum Beispiel Reiseziel, Reisedatum, Preiskategorie usw., ein Mobiler Agent, der *Verbraucheragent (Customer Agent)*, erzeugt (gelbe Biene in Abbildung 13). Dieser Agent interagiert und verhandelt mit den verschiedenen Verkäuferagenten (vgl. auch Abbildung 11) und überwacht das schwarze Brett („Ägypten-Angebot“ in Abbildung 13). Er filtert Informationenn aus, die nicht auf die Kriterien zutreffen und priorisiert sein Ergebnis. Am Schluss schickt der Agent eine eMail zum Konsumenten - aufgrund einer verbindlichen Deadline - und terminiert. So kann letztlich ein Angebot ausgewählt werden, welches noch 48 Stunden nach Eingang beim Konsumenten für ihn reserviert ist.

Nachdem TabiCan sich als Prototyp eines elektronischen Marktplatzes erfolgreich entwickelte, gingen die Entwickler von IBM Japan einen Schritt weiter (vgl. [Miya00]). Sie entwickelten ein

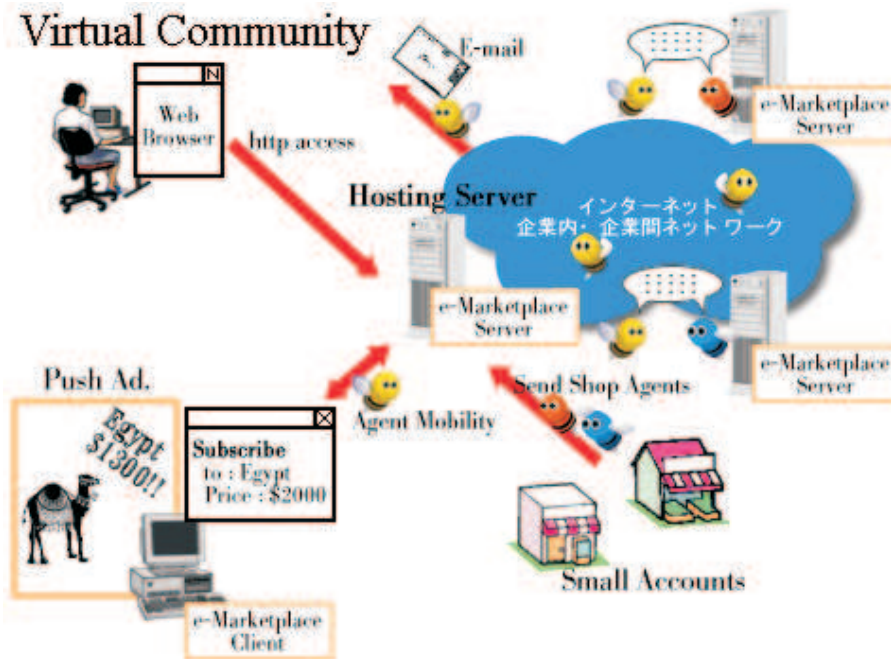


Abbildung 13: Der e-Marketplace von IBM Japan

Middleware-Software-Paket namens *e-Marketplace*, das auf der Technik von TabiCan basiert und für beliebige elektronische Marktplätze genutzt werden kann (siehe Abbildung 13).

Der e-Marketplace hat im wesentlichen zwei Vorzüge vor herkömmlichen elektronischen Marktplätzen:

- Agenten dienen als Beobachter für benötigte Informationen des Konsumenten. Sie stellen Informationen bereit, die man mit schlichten Datenbankabfragen so effektiv nicht finden würde.
- Jede Firma kann ihr Geschäft so gestalten, wie sie es will, d.h. jeder Verkäuferagent kann die Produkte seines Anbieters individuell repräsentieren. Zusätzlich ist die Ausstattung eines *Werbeagenten* (*Advertise Agent*) möglich, der für die Firma bei Konsumenten wirbt, die Produkte, ähnlich dem verkauften, vertreibt (z.B. Fahrzeuge und Fahrzeugversicherungen).

### 5.3 MATS (British Telecom) - Voyager von ObjectSpace Inc.

Das Projekt *Mobile Agent Team System* (MATS) der British Telecom hat unter anderem zum Ziel, die Vorteile Mobiler Agenten bei der Implementierung von groß-angelegten parallelen Applikationen (*large-scale parallel processing*) aufzuzeigen (vgl. [GH99]). MATS beruht auf der Mobilen Agentenplattform *Voyager* der ObjectSpace Inc.. Voyager wurde aufgrund seiner stabilen und gut dokumentierten Java-Bibliothek (mobile agent class library) ausgewählt (vgl. Abschnitt 1.3).

Beim Design von MATS wurde das Hauptaugenmerk darauf gerichtet, die Mobilen Agenten möglichst „leicht“ zu machen, das heißt ihren Umfang zu minimieren. Da MATS entworfen wurde, um rechenintensive Probleme parallel zu lösen, würden Mobile Agenten mit grossem Umfang beim Transport signifikante Mengen an System-Ressourcen verschlingen. Das Team der British Telecom löste dieses Problem, indem sie die komplexesten Funktionen in einen fest verankerten (nicht mobilen) Agenten lokalisierten, der mit den weniger spezialisierten

Mobilen Agenten über Nachrichten-Austausch kommuniziert. Dadurch wurden die Agenten in Abhängigkeit ihrer Rollen in drei verschiedene Typen eingeteilt. Eine Analogie wurde hierbei mit sozialen Insekten gezogen, die oft verschiedene Formen innerhalb einer Spezies entwickelt haben, von denen jede eine unterstützende Rolle in der Gesellschaft ausübt. MATS besteht aus den folgenden Agenten:

- *Bienenstock (Hive)*-Agenten: Sie unterstützen die Benutzerschnittstelle und koordinieren die Teams der Mobilen Agenten.
- *Königinnen (Queen)*-Agenten: Sie sind Teamleiter, beaufsichtigen die Host-Rechner und starten die Arbeitsanweisungen (worker threads).
- *Späher (Scout)*-Agenten: Sie bewegen sich durch das Netzwerk und analysieren die Computer-Ressourcen.

Die hierarchischen Beziehungen zwischen den Komponenten von MATS sind in Abbildung 14 dargestellt:

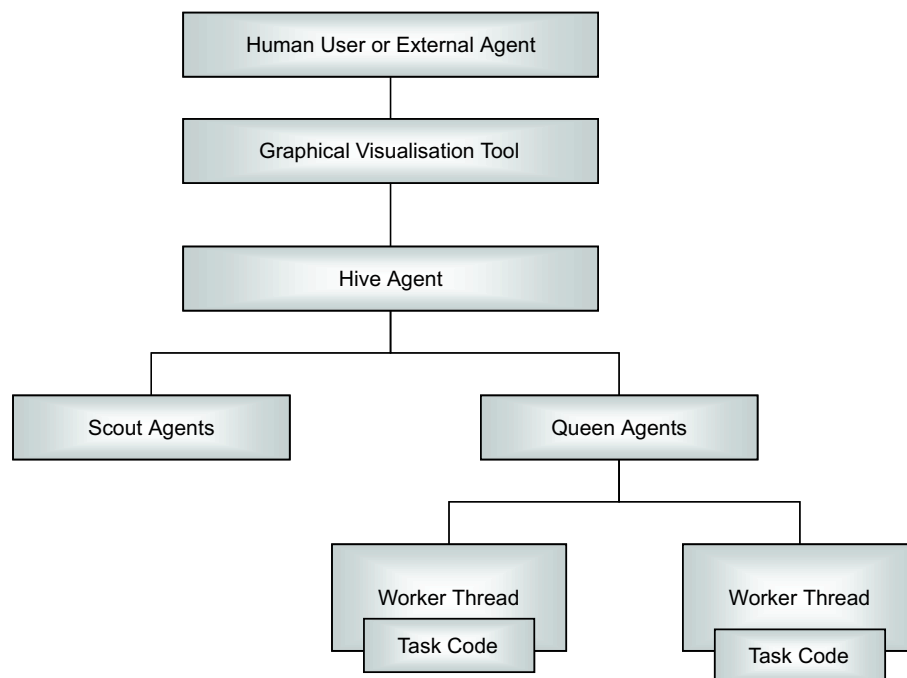


Abbildung 14: Der hierarchische Aufbau der Komponenten von MATS

Um eine verteilte Berechnung mittels der grafischen Benutzeroberfläche (Graphical Visualisation Tool) eines Problems durchzuführen, sendet der Benutzer (Human User) den Code und die spezifizierten Parameter an den *Bienenstock*-Agenten (siehe Abbildung 14). Dieser zerlegt die Berechnung in ein optimales Maß an Feinheit. Da dies eine sehr schwierige Aufgabe ist, ist der Benutzer angehalten, das Problem so zu strukturieren, dass es leicht in eine Menge an kooperierenden Aufgaben „heruntergebrochen“ werden kann. Jede Prozesskomponente wird anschliessend „eingepackt“ (*task wrapping*), d.h. mit dem nötigen Programmcode für Mobilität, Nachrichtenaustausch und Verteilung im Netzwerk versehen. Die resultierenden Befehlsketten werden *Königinnen*-Agent genannt.

Der *Bienenstock*-Agent muss ein exaktes Abbild seines lokalen Netzwerkes unterhalten, da er die *Königinnen*-Agenten auf die Server für die Mobilen Agenten verteilt. Jedoch ist es extrem schwierig für einen fest verankerten Agenten, den Status eines entfernten Rechners

genau einzuschätzen. Folglich haben die Entwickler von MATS aufgrund des Spezialisierungs-Prinzips die *Späher* entworfen, die genau diese Aufgabe übernehmen.

*Späher* sind Mobile Agenten, die periodisch vom *Bienenstock*-Agenten erzeugt und an die Rechner im Netzwerk verteilt werden. Nachdem ein *Späher*-Agent angekommen ist, führt er Tests der lokalen Ressourcen durch und analysiert zunächst welche Hard- und Software präsent ist. Dies ist ein gutes Beispiel für ein Programmkonstrukt, das davon profitiert mobil zu sein. Denn nur so kann der *Späher* einen Rechner bewerten - entfernt wäre er dazu nicht in der Lage. Zum Beispiel überwacht er lokale Ereignisse, um zu überprüfen ob die Tastatur betätigt wurde und zeichnet auf, ob die Menge an freiem Speicher konstant bleibt, fällt oder steigt. Anschließend verlässt der *Späher* den Rechner und migriert zum nächsten Knoten oder terminiert. Um die *Späher* mit möglichst wenig Daten zu beschweren, werden die gesammelten Daten gleich zum *Bienenstock*-Agenten geschickt, bewertet und benutzt um das Netzwerk-Abbild abzugleichen. Als Ergebnis erhält der *Bienenstock*-Agent den Status jeder IP-Adresse in seinem LAN, ob z.B. ein Mobiler-Agenten-Server läuft, ob der Rechner genutzt wird und das Potential der Ressourcen (freier Speicher). Daraus wird eine Ressourcen-Liste mit allen lokalen Rechnern in der Reihenfolge ihrer potentiellen Nutzbarkeit erzeugt (*Ressource Assessment and Modelling*).

Da alle Instrumente eines *Bienenstock*-Agenten Mobile Agenten sind (siehe Abbildung 14), kann er einen Host-Rechner nur dann nutzen, wenn ein Mobiler-Agenten-Server auf ihm läuft. Der *Bienenstock*-Agent versucht also, eine Infrastruktur für Mobile Agenten in seinem LAN aufzubauen. Dies erreicht er, indem er sich auf Host-Rechnern, auf denen noch kein Mobiler-Agenten-Server läuft, einwählt (via Telnet) und einen Mobilen-Agenten-Server als Hintergrundprozess startet (*Infrastructure Maintenance*).

MATS unterstützt über seine grafische Benutzeroberfläche drei Ansichten der Mobilen Agenten:

- Die *Floorplan-Ansicht*: Das physikalische Layout des LANs, in dem die Lokalisierung eines jeden Agenten dargestellt ist (siehe Abbildung 15).
- Die *geografische Ansicht*: Für den Fall, dass ein Agent auf einen Host-Rechner außerhalb des LANs migriert wird er in dieser Ansicht repräsentiert.
- Die *netzwerkorientierte Ansicht*: Hier werden die relativen Abstände zwischen *Bienenstock*-Agent und den Mobilen Agenten Servern basierend auf Ping-Zeiten angegeben.

MATS wurde beim Management und der Erzeugung eines verteilten parallelen Genetischen Programmierungsproblems (GPP) getestet (symbolische Regression einer Kurve). Kopien eines Teils des GPP wurden durch den *Bienenstock*-Agenten an verschiedene *Königinnen*-Agenten verteilt, die sich über die freien Netzwerk-Ressourcen verteilten (einschliesslich einiger UltraSparcs und Dual-Prozessor PCs unter Windows NT).

Die Forscher der BT Laboratories sehen in Mobilen Agenten eine Möglichkeit für einen Sprung hin zum *Intelligenten Netzwerk*, das automatisch seine Ressourcen anpassen kann, um als eine kohärente Einheit für groß-angelegte und verbesserte Computer-Berechnungen zu dienen. Als solches bieten die Mobilen Agenten ein vielversprechendes Anwendungsgebiet, das viele Anwender in heutigen Applikationen vermissen.

## 6 Expertenmeinungen

Zum Thema der Zukunft von Mobilen Agenten äußert sich Milojcic in [MJKo99]:

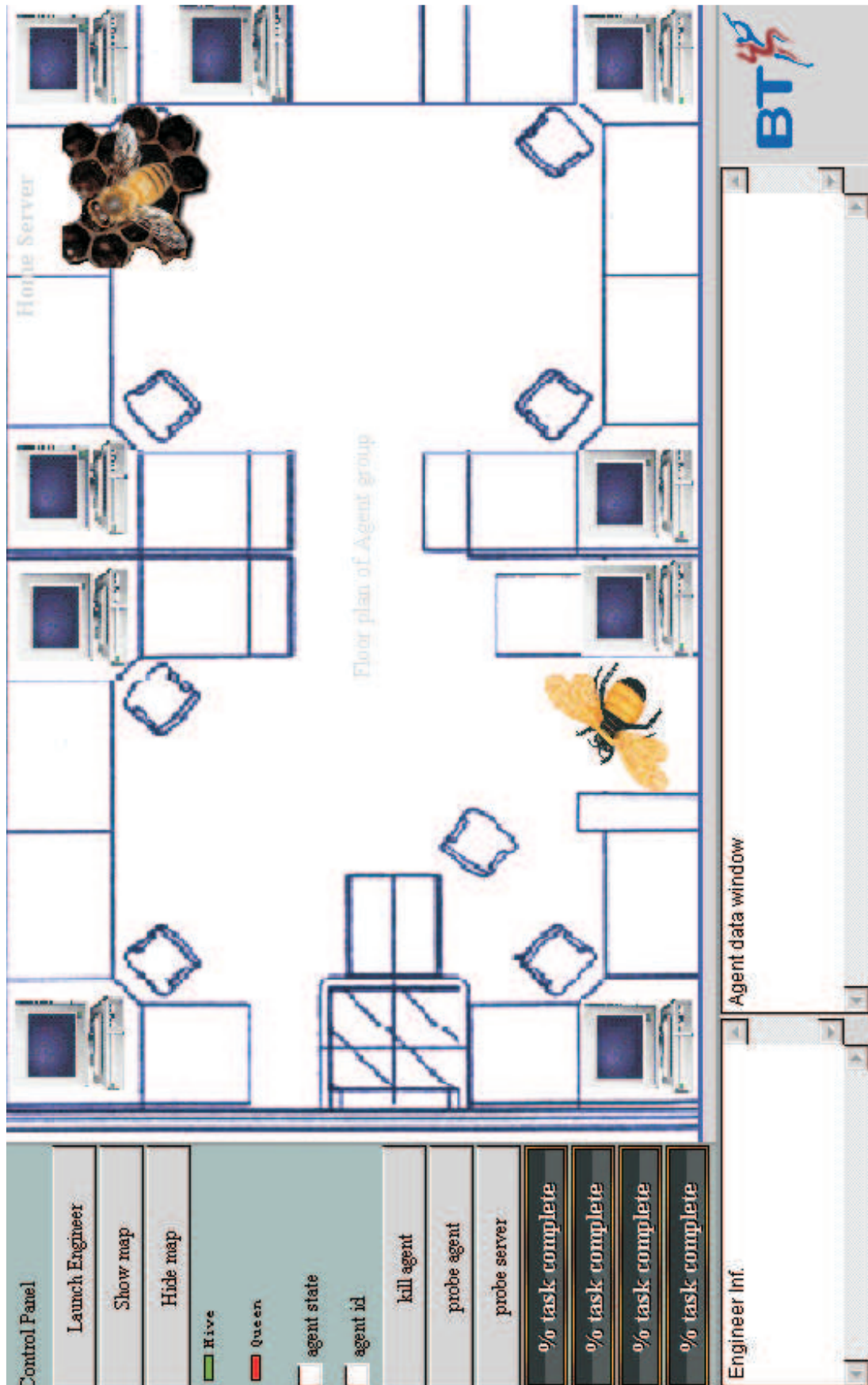


Abbildung 15: Die „Floorplan“ Ansicht von MATS

„Mobile Agenten schlagen den gleichen Weg ein, wie schon die Prozess-Migration: Sie galt als interessante Forschungstätigkeit, wurde jedoch nie kommerziell um- und eingesetzt.“

Mit dieser Meinung vertritt Milojevic allerdings nicht die Meinung aller Forscher auf dem Gebiet der Mobilien Agenten. Die Anwendungen in denen die Mobilien Agenten sich als Zukunftstechnologie durchsetzen könnten, werden vor allem in folgenden Bereichen gesehen:

- Datenintensive Anwendungen.
- Anwendungen, bei denen Agenten von einem Gerät (z.B. Mobiles Endgerät, PDA) aus gestartet werden, das nicht permanent mit einem Netzwerk verbunden ist (disconnected computing).
- Information Retrieval für erweiterbare Server, auf denen der Agent einen personalisierten Vertreter des Benutzers darstellt.
- Dynamischer Software-Vertrieb bzw. -Installation in einem grossen Unternehmen.
- komplexes Netzwerkmanagement.

Einige Experten sehen dabei nicht unbedingt die Notwendigkeit einer Killer-Applikation.

Beim Verhalten zur Umgebung eines Mobilien Agenten sind sich die Autoren von [MJKo99] einig darüber, dass man von den Arbeiten auf dem Gebiet der Prozessmigration - aus der die mobile Agententechnologie hervorging (vgl. Abbildung 8) - lernen kann (z.B. eine Migration erst nach der Beendigung eines Prozesses zu starten). Mobile Agenten sind nach Meinung von Kotz in [MJKo99] aufgrund ihrer Heterogenität eher zu Tätigkeiten auf höherem (Schichten-) Niveau zu gebrauchen (z.B. Information Retrieval auf dem Host-Rechner) als für aktive Netzwerke. Er ist der Meinung, dass Mobilien Agenten im Gebiet des *Mobile Computing* eine große Zukunft bevorsteht. Dem schließt sich allerdings Lange in [MJKo99] nicht an.

Die größten Probleme Mobiler Agenten werden von allen Forschern in der Sicherheit Mobiler Agenten-Systeme gesehen. Problemfelder stehen nach Meinung von Johansen in [MJKo99] auch im Bereich der Diensteverweigerung eines Host-Rechners (vgl. Abschnitt 2) und in der Agentenunversehrtheit an. Petrie sieht in [MJKo99] das Problem, dass es noch zu kompliziert ist Mobile Agenten aufgrund der existierenden Ablaufumgebungen zu entwerfen. Kotz in [MJKo99] schliesst sich dieser Meinung an und sieht in Zukunft ein *Smart Phone* vor sich, mit dem man entweder vordefinierte oder selbst entworfene Agenten ins Internet schicken kann. Die Autoren von [MJKo99] sind sich allerdings in einer Sache einig: Mobile Agenten stellen eine elegante und vielversprechende Lösung für *Large Scale Processing* (vgl. MATS in Abschnitt 5.3) und Netzwerkmanagement auf höheren Schichten dar.

## 7 Fazit

Obwohl bisher nur sehr wenige Mobile Agenten implementiert wurden, steht ihnen meiner Meinung nach in den vier Bereichen von Abschnitt 6 eine grosse Zukunft bevor. Da der Absatz an Mobilien Endgeräten stetig wächst und *Mobile Computing* in Zukunft zum Standard wird, wird sich ein positiver Druck nach Lösungen für das *Information Retrieval* ohne ständige Verbindung mit einem Netzwerk aufbauen. Dadurch wird das Mobile-Agenten-Paradigma nochmals frischen Wind bekommen und letztendlich zu dieser Lösung beitragen.

Ich sehe die Mobile Agenten Technologie auch nicht als Komplettlösung für irgendein Problem, sondern als Werkzeug innerhalb einer Lösung auf den Gebieten, die unter Abschnitt 6 vorgestellt wurden.

Man kann auch die Mobile-Agenten-Technologie als Ergebnis einer Grundlagenforschung sehen, als ein neues Werkzeug, das wie eine neu erfundene chemische Substanz oder ein neu entdecktes Material noch auf seine ideale Verwendung wartet. Daher erscheint der Einsatz in neuen noch nicht erdachten Systemen (welche sich dieser Basistechnologie unmittelbar bedienen) noch erfolgversprechender als der Einsatz der Mobile-Agenten-Technologie in bestehenden Anwendungssystemen.

## Literatur

- [AaAa97] B. Aaron und A. Aaron. *ActiveX Technical Reference*. Prima Pub. 1997.
- [Brad97] J. Bradshaw (Hrsg.). *Software Agents*. MIT Press. 1997.
- [Broc84] Brockhaus (Hrsg.). *Der Neue Brockhaus: Lexikon und Wörterbuch*, Band 7. Brockhaus. 1984.
- [Fisc99] Mark Fischetti. Tireless Travel Agent. *IBM Research Magazine, Special Report: The Rise of e-Business / Wheeling and Dealing* Band 1, 1999.
- [FrGr96] Stan Franklin und Art Graesser. Is it an Agent, or just a Program? An Taxonomy for Autonomous Agents. In *Proceedings of the 3<sup>rd</sup> International Workshop on Agent Theories, Architectures and Languages*. Springer, 1996.
- [GHCN99] R. Ghanea-Hercock, J.C. Collis und D.T. Ndumu. Co-operating Mobile Agents for Distributed Parallel Processing. In *Proceedings of the 3<sup>rd</sup> Int. Conference on Autonomous Agents*, Mai 1999.
- [KaAG98] G. Karjoth, N. Asokan und C. Gülcü. Protecting the Computation Results of Free-Roaming Agents. In *Proceedings of the 2<sup>nd</sup> Int. Workshop on Mobile Agents*, 1998, S. 195–207.
- [KoGr99] D. Kotz und R. S. Gray. Mobile Agents and the Future of the Internet. *ACM Operating Systems Review* 33(3), August 1999, S. 7–13.
- [KrRe00] G. Krüger und D. Reschke. *Telematik, Lehr- und Übungsbuch*. Fachbuchverlag Leipzig. 2000.
- [LaOs99] D. Lange und M. Oshima. Seven Good Reasons for Mobile Agents. *Communications of the ACM* 42(3), März 1999, S. 88–89.
- [Miya00] Shichirou Miyashita. *Aglets Software Development Kit*. IBM Japan, 2000.
- [MJKo99] D. Milojevic, D. Johansen, D. Kotz und andere. Trend Wars: Mobile Agent Applications. *IEEE Concurrency*, Juli 1999, S. 80–90.
- [Morr98] P. Morreale. Agents on the Move. *IEEE Spectrum*, April 1998, S. 34–41.
- [Ora] [www.oracle.com/mobile](http://www.oracle.com/mobile). Internet.
- [PhKa98] V. Pham und A. Karmouch. Mobile Software Agents: An Overview. *IEEE Communications Magazine*, Juli 1998, S. 26–37.
- [Rein94] A. Reinhardt. New Agent-based WANs Presage the Future of Connected Computing. *BYTE.COM - CMP's TechWeb, The IT Network*, Oktober 1994.
- [RoHR97] K. Rothermel, F. Hohl und N. Radouniklis. Mobile Agent Systems: What is missing? In H. König, K. Geihs und T. Preuß (Hrsg.), *Distributed Applications and Interoperable Systems (DAIS'97)*, 1997, S. 111–124.
- [Sun97] Microsystems Sun. The java servlet api. *white paper*, 1997.
- [Tab] [www.tabican.ne.jp/](http://www.tabican.ne.jp/) (auf japanisch). Internet.
- [Wagn97] G. Wagner. Software mit Managerqualitäten. Agenten - Programme mit Überzeugungen und Absichten. *c't Magazin für Computertechnik* Band 15, 1997, S. 234ff.

- [WBWi98] R. Zarnekow W. Brenner und H. Wittig. *Intelligente Softwareagenten*. Springer, Berlin. 1998.
- [WoPM99] D. Wong, N. Paciorek und D. Moore. Java-based Mobile Agents. *Communications of the ACM* 42(3), Marz 1999, S. 92–102.