Mobile Agenten

Friedemann Mattern
Fachbereich Informatik, Technische Universität Darmstadt

Mobile Agenten sind autonome Programme, die in einem Netz von Rechnern umherwandern und dabei im Auftrag eines Nutzers Dienste verrichten. Wir beschreiben das allgemeine Konzept, mögliche Anwendungsgebiete, Anforderungen an Infrastruktur und Sprachen für mobile Agenten sowie Problembereiche wie beispielsweise Sicherheitsaspekte. Schließlich geben wir einen Überblick über aktuelle Prototypen und Forschungsprojekte.

Mobile agents. *Mobile agents are autonomous program entities that travel through a network of heterogeneous machines and act on behalf of a user. We describe the general concept of mobile agents, possible application areas, requirements on agent infrastructures and languages, and problem issues such as security. We also give an overview on current prototypes and research projects.*

1 Das Konzept mobiler Agenten

Der Begriff des mobilen Agenten wurde Anfang der 90er Jahre von der Firma General Magic geprägt, die interessanterweise 1997 ein Patent [17] für die zugrundeliegende Technologie erhielt. In diesem Sinne sind mobile Agenten Programme in Form von autonomen Objekten, die in einem Netz heterogener Rechner – typischerweise Intranets oder Teile des Internets – umherwandern und dabei im Auftrag eines Nutzers Dienste verrichten. Ein Agent entscheidet dabei selbst, aufgrund lokaler Gegebenheiten, ob, wann und wohin er gegebenenfalls migrieren möchte. Bei der Migration wird der dynamische Prozeßzustand des Agenten eingefroren und zusammen mit Kontextinformationen und dem variablen Datenteil in eine zu versendende Nachricht verpackt. Am Zielort wird der Prozeßzustand wieder aufgetaut und der Agent läuft an der unterbrochenen Stelle nahtlos weiter.

Zur Verrichtung seiner Arbeit interagiert ein Agent mit der jeweiligen lokalen Umgebung, also dem Wirtssystem, das ihn gerade beherbergt, dabei kann er auch mit anderen lokal vorhandenen oder entfernten Agenten kooperieren. Ferner sollte ein Agent mit seinem an einem anderen Ort residierendem Auftraggeber kommunizieren können, etwa um Zwischenresultate abzuliefern oder neue Daten und Instruktionen anzufordern. Letzteres wird allerdings ein eher seltenes Ereignis sein, da vom Konzept her Agenten in der Lage sein sollten, weitestgehend autonom zu handeln.

2 Mobile Agenten als Architekturprinzip verteilter Systeme

Das Paradigma mobiler Agenten läßt sich zunächst als ein neues Architektur- und Strukturierungsprinzip zur Realisierung verteilter Anwendungen in einer offenen, heterogenen und sich dynamisch

ändernden Umgebung auffassen. Als solches verallgemeinert und ergänzt es andere Strukturierungsprinzipien wie beispielsweise das klassische Client-Server-Modell mit dem hierfür typischen "Remote Procedure Call" (RPC) als Kommunikationsmuster bzw. die objektorientierte Variante "Remote Method Invocation". Abbildungen 1 und 2 veranschaulichen den wesentlichen Unterschied der beiden Modelle: während im ersten Fall i.w. nur die Daten zu einer beim Server bereits vorhandenen Prozedur transportiert werden und der Client typischerweise während der Datenübermittlung und entfernten Prozedurausführung inaktiv ist, wird im zweiten Fall ein Agent zum Server gesendet, der lokal mit dem Dienst interagiert und dort im Auftrag des Klienten die Daten verarbeitet. Dies geschieht asynchron zu möglichen anderen Aktivitäten des Klienten.

Schon in dieser einfachen Form ist bei manchen Anwendungsklassen das Agentenparadigma – unter gewissen Voraussetzungen hinsichtlich der Migrationskosten – dem Client-Server-Modell in einigen wichtigen Aspekten überlegen:

1. Kommunikationsbandbreite: Besteht die Aufgabe darin, eine größere Menge entfernt gespeicherter Daten zu verarbeiten, so kann es effizienter und billiger sein, die Verarbeitungsmethode zu den Daten zu bringen ("function shipping"), anstatt die Daten vom Server zum Client zu transportieren ("data shipping"). Falls die benötigte Verarbeitungsmethode bereits in parametrisierter Form auf dem Server vorliegt ("stored procedure"), wie dies etwa bei WWW-Suchmaschinen oder vielen Datenbankservern der Fall ist, stellt das Agentenparadigma diesbezüglich sicherlich keinen Vorteil dar. Anders sieht es aber aus, wenn sich die Verarbeitungsprozedur nicht antizipieren läßt. Beispiele dafür stellen etwa die spezifische Auswertung sehr großer Mengen meteorologischer Daten dar, oder auch die Analyse von aktuellen Börsendaten entsprechend einer eigenständigen Bewertungsstrategie. (Bei diesem Szenario kann gewissermaßen jeder seinen persönlichen Broker-Agenten direkt an der Börse sitzen haben...)

2. Reaktionszeiten: Ein Agent als Statthalter vor Ort kann auf Änderungen der lokalen Umgebung unmittelbar reagieren. Dies ist bei Echtzeitanwendungen (z.B. Robotersteuerung oder Ereignisbearbeitung bei der dynamischen Konfiguration von Kommunikationsnetzen) von Vorteil, da so Verzögerungszeiten bei der Meldung von Ereignissen zu einer entfernten Entscheidungsinstanz und der Rückübermittlung der Antwort vermieden werden können.

3. Offline-Betrieb: Beauftragt ein Nutzer einen Agenten mit der asynchronen Durchführung einer Aufgabe, so ist nur während der kurzen Zeiten des Aussendens bzw. Rückkehrens des Agenten eine Verbindung zum "Rest der Welt" notwendig. Dies ist insbesondere für Szenarien im Bereich des "mobile computing" von Interesse, wo eine ständige Verbindung zum Festnetz, in dem der Agent sich dann vorwiegend bewegt, teuer oder gar aus technischen Gründen nicht machbar ist.

4. Lokale Betriebsmittel: Gewisse Betriebsmittel oder Dienste sind aus unterschiedlichen Gründen oft nur lokal nutzbar und nicht über Dienstschnittstellen von außen ansprechbar. Agenten als lokale Stellvertreter können u.U. ein technisch einfacher Weg sein, diese Betriebsmittel extern verfügbar zu machen.

3 Mobiler Code

Die Idee, Programme zu versenden und an entfernter Stelle auszuführen, ist nicht neu. Eine rudimentäre Form hiervon war bereits mit dem "Remote Job Entry"-Konzept der Batch-Betriebssysteme der 60er-Jahre gegeben. Auch bei der weit verbreiteten Seitenbeschreibungssprache Postscript werden Programme an einen entfernten Prozessor (in einem Drucker) versendet.

Da ein entfernt ausführbares Programm in einer Sprache vorliegen muß, die vom Zielsystem verstanden wird, und meist eine größere heterogene Menge potentieller Zielsysteme ins Auge gefaßt wird, sollte für mobilen Code eine hardwareunabhängige Sprache verwendet werden. In der Praxis kommen überwiegend interpretative Sprachen zum Einsatz; Beispiele hierfür sind Skriptsprachen wie Tcl oder Perl, die für die gängigsten Rechnerplattformen verfügbar sind, in eingeschränkter

Weise auch Betriebssystem-Kommandosprachen wie z.B. Shell-Sprachen unter UNIX.

In technischer Hinsicht ist es einfach, auf einem Server einen Hintergrundprozeß ("Dämon") zu installieren, welcher über Nachrichten eingehenden Programmcode dem lokalen Sprach- oder Kommandointerpreter zur Ausführung übergibt. Bei "Active Mail" [2] wurde dies tatsächlich so realisiert: die Email kann zusätzlich zu normalem Text auch Programmcode enthalten, der beim Empfänger ausgeführt wird. Derartig prototypisch einfach zu realisierende Systeme zeigen jedoch auch unmittelbar einige Defizite und Problembereiche, die für umfassende mobile Agentensysteme relevant sind:

1. Sicherheit: Fremden Programmen, die u.U. sogar unerwartet eintreffen, soll selbstverständlich nicht alles gestattet sein. (Auch Computerviren sind Ausprägungen von mobilem Code!) Die Festlegung eines geeigneten Sicherheitsmodells, das definiert, welche Operationen unter welchen Umständen ausgeführt werden dürfen, sowie die Realisierung der dafür notwendigen Sicherheitsinfrastruktur für die Verwaltung von Rechten, Zertifikaten etc. ist nicht einfach.

2. Kontextinformation: Das Agentenparadigma verlangt in seiner originären Form, daß ein Agent jederzeit (z.B. mittels eines Befehls "go") seine Ausführung auf einem anderen Rechner fortführen kann. Im Code selbst sind jedoch weder der Variablenzustand noch der Ausführungskontext (Befehlszähler, Aufrufverschachtelung, dynamische Speicherbereiche) enthalten, was Voraussetzung dafür ist, die Berechnung an der unterbrochenen Programmstelle fortzusetzen. Ohne weitergehende Maßnahme ist auch die gewünschte "Ortstransparenz" nicht gewährleistet: notwendige Ressourcen wie ausreichend Hauptspeicher oder bestimmte Laufzeitbibliotheken stehen auf dem Zielsystem u.U. nicht oder nur in modifizierter Weise zur Verfügung, externe Referenzen verlieren ihre Bedeutung und Kontrollinformation der Ausführungsumgebung (Laufzeitsystem, Betriebssystem) über das migrierte Programm geht verloren.

3. Infrastruktur: Mobile Agenten sollen u.a. die Möglichkeit besitzen, sich über lokal vorhandene Ressourcen zu informieren, miteinander zu kommunizieren und auf Ausnahmesituationen (z.B. temporäre Unterbrechung einer Kommunikationsverbindung) autonom zu reagieren. Da dies nicht in jeder Anwendung, die das Agentenparadigma nutzt, neu realisiert werden sollte, ist eine vorgegebene Infrastruktur, etwa im Sinne von Middleware-Komponenten, erforderlich.

Ein interessantes und aktuelles Beispiel für mobilen Code stellen Java-Applets dar, die als Bytecode in komprimierter Form versendet werden und auf Empfängerseite durch einen Interpreter (die virtuelle Java-Maschine, die z.B. auch in gängige WWW-Browser integriert ist) ausgeführt werden. Java besitzt ein – wenn auch nicht in jeder Hinsicht perfektes – integriertes Sicherheitskonzept und sorgt durch Maßnahmen wie einheitliche Grundbibliotheken dafür, daß auf dem Zielsystem die erwartete Ausführungsumgebung bereitsteht.

Allerdings wird auch bei Java kein "laufendes" Programm mit Ausführungskontext transportiert, sondern es wird nur der Code weitergegeben, der auf Empfängerseite wieder von Beginn an neu ausgeführt wird. Bezogen auf die Client-Server-Rollen werden Java-Applets auch nicht im Sinne von mobilen Agenten vom Client zum Server geschickt, um dort wünschenswerte Dienste zu verrichten, sondern Applets wandern in umgekehrter Richtung, typischerweise um beim Client graphische Animationen durchzuführen und so den Versand umfangreicher Pixeldaten einzusparen. Auch Java-Applets stellen also noch keine mobilen Agenten im eigentlichen Sinne dar.

4 Anwendungsszenarien mobiler Agenten

Die gegenwärtige Attraktivität des Paradigmas mobiler Agenten in der Öffentlichkeit beruht zu einem guten Teil auf denkbaren Anwendungsszenarien aus dem Bereich des Electronic Commerce: Agenten treten als Anbieter, Käufer, Verkäufer und Vermittler von Waren und Dienstleistungen auf und bewegen sich dabei gegebenenfalls im Internet von Ort zu Ort. So stellt man sich etwa vor, daß ein mobiler Agent im Auftrag eines Nutzers eine Einkaufstour durchführt, sich über Angebote in-

formiert und auf seiner Rundreise das billigste Angebot ermittelt – idealerweise erfolgt dann auch gleich der Kauf mit elektronischem Geld direkt von Agent zu Agent.

Neben dem elektronischen Markt denkt man aber auch an andere attraktive Anwendungsmöglichkeiten, von denen nachfolgend einige kurz skizziert werden sollen.

1. Informationsbeschaffung: Mobile Agenten, versehen mit dem Profil des Auftraggebers, suchen auf Datenbankservern, WWW-Seiten und anderen im Internet verteilten Datenquellen nach relevanten Informationen. Dabei kommunizieren sie auch mit Informationsvermittlern und Meta-Indizes und tauschen gegebenenfalls Informationen mit anderen Suchagenten, die ein ähnliches Profil besitzen, aus.

2. Unterstützung von Gruppenarbeit: Mobile Agenten in Form aktiver Dokumente bringen (beispielsweise in Workflow-Systemen) die zur Bearbeitung notwendige Funktionalität sowie weiteres Kontextwissen zur Bearbeitungsstelle mit. Auch Aspekte wie Terminvereinbarung und Projektplanung können durch mobile Agenten gut unterstützt werden.

3. Personalisierte Dienste: Ein mobiler Agent kann von einem Dienstanbieter zum Kunden gesendet werden, um dort vor Ort einen generischen Dienst in spezieller Weise zu erweitern. Beispiele wären etwa Funktionserweiterungen bei Telekommunikationsgeräten, spezielle Plug-ins für WWW-Browser oder die Ergänzung von Softwarekomponenten.

4. Fernwartung: Agenten können vor Ort Systemkomponenten (beispielsweise Vermittlungsinstanzen in Kommunikationsnetzen) überwachen und gegebenenfalls auch die Diagnose von Fehlverhalten durchführen sowie Reparaturmaßnahmen treffen.

Damit derartige Vorstellungen realisiert werden können, sind weit mehr Voraussetzungen zu erfüllen als die zuvor angesprochene Codemobilität. Dies betrifft insbesondere Infrastrukturmaßnahmen.

5 Infrastruktur für mobile Agenten

Zu ihrer Ausführung benötigen mobile Agenten auf dem Wirtssystem, auf dem sie zeitweilig zu Gast sind, eine in eine Agentenplattform eingebettete Ablaufumgebung (vgl. Abbildung 3). In Analogie zu einem Hotel muß diese gewisse Dienste und Betriebsmittel bereitstellen (wie beispielsweise Kommunikationsmöglichkeiten, Auskunftdienste, Speicherplatz und Rechenzeit), sie wird aber auch dafür Sorge tragen müssen, daß das Gastrecht nicht durch unsoziales Verhalten (Monopolisieren von Ressourcen, Stören oder Bestehlen anderer Gäste oder des Hotels) oder gar Vandalismus (Zerstören von Teilen der Infrastruktur) mißbraucht wird. Damit dies ordnungsgemäß funktioniert, sind einige Regeln einzuhalten: Agenten checken sich ein und aus und führen gewisse – wenn nicht gar alle – Aktionen nur unter strikter Kontrolle der Ablaufumgebung durch.

Diese Kontrolle ist am einfachsten zu realisieren, wenn der Agentencode nicht direkt vom Prozessor des Wirtssystems ausgeführt wird, sondern die Ablaufumgebung die Aktionen eines Agenten (im Sinne einer virtuellen Maschine) interpretiert: es kann dann auch die Inanspruchnahme von Betriebsmitteln genau nachgehalten werden und es können relativ einfach Maßnahmen gegen eine Monopolisierung getroffen werden. In jedem Fall müssen potentiell “gefährliche” Operationen und Systemfunktionen abgefangen und nach Maßgabe eines Sicherheitsmodells behandelt werden (z.B. entsprechend den Rechten eines Agenten zurückgewiesen oder in modifizierter Form ausgeführt werden).

Damit sich Agenten über die lokal vorhandenen Betriebsmittel und deren Eigenschaften informieren können oder den Ressourcenbedarf mit der Ablaufumgebung aushandeln können, ist eine einheitliche Sprache (zumindest im Sinne einer standardisierten Funktionsschnittstelle) notwendig. Es muß auch festgelegt sein, wie im Fehlerfall (z.B. nicht vorhandenes Betriebsmittel, Fehlverhalten eines Agenten, nicht erreichbares Migrationsziel etc.) verfahren werden soll und welche Rollen dabei dem Agenten bzw. der Ablaufumgebung zukommen.

Über lokale Aspekte hinaus sollten Agentenplattformen zur Unterstützung komplexer Anwendungsszenarien auch globale Dienste erfüllen: Zum einen sollen Agenten fehlerfrei von einer Plattform zu einer anderen migrieren können, was entsprechend vereinheitlichte Kommunikationsprotokolle voraussetzt, zum anderen sollte die globale Kontrolle und die Realisierung globaler Dienste unterstützt werden. So müssen gegebenenfalls verschollene Agenten aufgespürt, Botschaften an abgewanderte Agenten nachgesendet und globale Maßnahmen zur Gewährleistung von Fehlertoleranz und Sicherheit vorgesehen werden. Dies erfordert eine Kooperation der einzelnen Agentenplattformen untereinander, die in einer offenen und verteilten Welt nicht einfach zu realisieren ist.

Dies zeigt auch ein Grundproblem gegenwärtiger Prototypsysteme: die Realisierung dieser Unterstützungsmechanismen ist sehr aufwendig, andererseits läßt sich das Paradigma mobiler Agenten i.a. nur dann mit Gewinn einsetzen, wenn die Infrastruktur vorhanden ist und entsprechende Agentenplattformen genügend weit verbreitet sind. Prototypsysteme mit Anspruch auf Praxistauglichkeit versuchen daher, soweit wie möglich bereits existierende Infrastrukturmechanismen verteilter Systeme zu nutzen bzw. darauf aufzubauen. Zu den entsprechenden Technologien gehören CORBA (u.a. Kommunikationsdienste, Ereignisweiterleitung, Verzeichnis- und Namensdienste), Java (mobiler Code in Form von Applets sowie ein integriertes Sicherheitsmodell) und WWW (als “Heimstätte” für Agentenplattformen und natürliche Schnittstelle zum Internet mit seinen weiteren Ressourcen).

Von einer praxistauglichen verteilten Agentenplattform werden letztendlich nicht nur Anwendungen profitieren, die sich des Paradigmas mobiler Agenten in originärer Weise bedienen – die Rechnerunabhängigkeit und Robustheit gegenüber manchen Fehlersituationen, die bei einer solchen Infrastruktur gegeben sein muß, ist natürlich generell für verteilte Anwendungen von Interesse.

Jenseits der geschilderten Infrastrukturmechanismen besteht hinsichtlich einer höherwertigen Kooperationsfunktionalität allerdings noch ein Unterstützungsbedarf für mobile Agenten, der derzeit erst weniger umfassend erforscht ist und an der Schnittstelle der beiden Gebiete “Verteilte Systeme” und “Künstliche Intelligenz” anzusiedeln ist: Agenten, die in kooperativer Weise eine Aufgabe bearbeiten, müssen ihre Handlungen koordinieren und miteinander interagieren. Unabhängig davon, ob ein mobiler Agent mit einem anderen an einer bestimmten Stelle im Netz zusammentrifft und dort mit diesem lokal kommuniziert, oder ob die Kommunikation entfernter Agenten über Nachrichten oder andere Mechanismen, wie beispielsweise abstrakte gemeinsame Kommunikationssphären, stattfindet, ist hierfür eine einheitliche Sprache als Verständigungsbasis notwendig. Die Verwendung derartiger “Agent Communication Languages” (wie z.B. KQML [5]) für mobile Agenten befindet sich derzeit allerdings noch im Experimentierstadium.

Gleiches gilt auch für eine in einer offenen verteilten Welt letztendlich unabdingbaren Vermittlungsinfrastruktur: mobile Agenten werden nicht in jedem Fall von vornherein wissen, wo momentan im Netz geeignete Ressourcen vorhanden sind. Sie sind dann auf die Dienste von Vermittlungsinstanzen angewiesen, die weiterhelfen können, wobei die Nutzung solcher Vermittler auch wieder eine gemeinsame Verständigungsbasis erfordert. Weiterhin benötigt ein Agent generische Strategien und Metawissen in der Form, daß er Heuristiken zum Auffinden und Bewerten von Ressourcen für einen bestimmten Zweck nutzen oder entwickeln kann, welche ebenfalls in einer systemunabhängigen Sprache formuliert sein sollten.

6 Forschungs- und Problembereiche

Mobile Agentensysteme sind seit Mitte der 90er-Jahre ein weltweit verstärkt bearbeitetes Forschungsgebiet [15, 24]. Neben der Realisierung von Prototypen und ersten Anwendungen werden hierbei auch prinzipielle, derzeit noch nicht vollständig gelöste, Problembereiche angegangen. Zu diesen gehört neben den oben angesprochenen Fragen der Koordination und Kommunikation zwischen Agenten und den erwähnten Aspekten im Zusammenhang mit der Infrastruktur im wesentlichen das Thema Sicherheit.

Tatsächlich scheint der erfolgreiche Einsatz mobiler Agenten, vor allem für den elektronischen Markt, entscheidend davon abzuhängen, inwieweit die Sicherheitsproblematik zufriedenstellend gelöst werden kann. Hierbei gilt es, zwei Aspekte zu betrachten: zum einen den Schutz des Wirtssystems vor böswilligen Agenten, andererseits aber auch den Schutz von Agenten (bzw. damit realisierten Anwendungen) vor böswilligen Wirtssystemen (beispielsweise mit manipulierten Agentenplattformen).

Der erste Teilaspekt – der Schutz des Wirtssystems samt seiner Ablaufumgebung und der anderen sich dort aufhaltenden Agenten – ist relativ gut verstanden. Neben einem durchdachten Sicherheitsmodell (wer darf was unter welchen Umständen?) und dessen Implementierung durch (u.U. hierarchische) Zugriffsrechte, Sicherheitskontexte etc. gehört dazu die oben erwähnte interpretative Ausführung des Agentencodes bzw. das Abfangen kritischer Befehle. Auch das "Sandkastenmodell" von Java [6], bei dem Agenten sich in einer abgeschotteten und behüteten Welt beliebig tummeln dürfen, ist ein solches Konzept. Damit lassen sich im Prinzip auch Monopolisierungen von Ressourcen durch Agenten und "Denial of Service"-Attacken angehen, selbst wenn hier im Detail noch Forschungsbedarf besteht.

Gegen das Ausspähen oder die Verfälschung von Daten und Programmtext durch Dritte während der Migration von Agenten helfen klassische kryptographische Methoden wie Verschlüsselung, Signierung und Prüfsummen. Die bei einer umfangreicheren Nutzung des Internets notwendige Infrastruktur aus Zertifizierungsinstanzen zum Nachweis der Authentizität von Nutzern und Zertifikaten ist derzeit sowieso im Entstehen.

Schwieriger ist die Problematik des Schutzes von Agenten vor böswilligen Ablaufumgebungen: ein Agent führt nicht selbst Befehle aus, sondern übergibt gewissermaßen jeden seiner Befehle an die lokale Ablaufumgebung zur Ausführung und liefert sich ihr somit völlig aus. Ein Agent kann vor Ort daher nicht überprüfen, ob der Sprachinterpreter unverfälscht ist, ob seine eigenen Befehle korrekt ausgeführt werden und ob er tatsächlich an ein angegebenes Migrationsziel weitergeleitet wird [4]. Da die Ablaufumgebung den Agentencode kennen muß (schließlich soll sie ihn ja ausführen), kann der Agent im Programmtext an sich auch keine Geheimnisse verstecken – allenfalls kann der Erzeuger den Code so verwirrt haben, daß die eigentliche Funktion daraus nicht (einfach) zu rekonstruieren ist [8].

Legendär ist in diesem Zusammenhang das Problem des Agenten, der eine Reihe von Anbietern besucht, um beim billigsten zu kaufen. Er ist dabei mannigfaltigen Fährnissen ausgesetzt: ein Anbieter könnte die mitgeführten Daten der vorher besuchten Konkurrenten verfälschen, er könnte direkt den Programmteil mit "if...then buy(...)" suchen und den "richtigen" Zweig ausführen, er könnte aber auch dem Agenten das elektronische Geld einfach wegnehmen bzw. dessen Kreditkartennummer mißbrauchen. Noch interessanter wird es, wenn zwei oder mehr Anbieter sich verschwören oder wenn Agenten in böswilliger Absicht kloniert werden (Clones sind vom Original nicht zu unterscheiden). Sofern überhaupt Maßnahmen gegen solche Attacken existieren, erfordern diese i.a., daß der Agent sich zeitweise an einen sicheren Ort zurückzieht oder daß er zwischendurch mit anderen Instanzen (Nutzer, Bank etc.) kommuniziert – dies widerspricht aber zumindest der Idee der Autonomie von Agenten; der Nutzer hätte vermutlich einfacher jeden Anbieter direkt abfragen können!

Neben der Sicherheitsproblematik existieren noch weitere gegenwärtig erforschte Problemaspekte im Bereich "mobile Agenten", von denen einige nachfolgend kurz angerissen werden:

1. Fehlertoleranz: In einer offenen, verteilten Welt ist es oft der Fall, daß einiges (zumindest zeitweise) nicht funktioniert. Agenten sollten ein derartiges Fehlverhalten nach Möglichkeit überleben und damit realisierte Anwendungen sollten von diesen Fehlern möglichst nichts mitbekommen. Die analog zum RPC gewünschte "exactly once"-Semantik von Agenten ist allerdings nur unter gewissen Voraussetzungen erreichbar: ob ein Agent während des Transports verloren ging oder nur getrödelt hat, läßt sich prinzipiell nicht unterscheiden. Eine nach einer ausgebliebenen Empfangsbestätigung (die unter Umständen selbst verloren gegangen sein könnte) erneut versendete Kopie

darf aber nicht zu Agenten-Clones führen. Insbesondere möchte man vermeiden, daß eine Agentenplattform nach einem anscheinend erfolglosen Migrationsversuch einen Agenten mit dem Hinweis "Migrationsziel nicht erreichbar" lokal wiederaufleben läßt, obwohl die versendete Kopie noch unterwegs ist oder nur die Ankunftsbestätigung verloren ging. Generell wünscht man sich, insbesondere für Anwendungen im Bereich des Electronic Commerce, weitergehende Kontrollmechanismen, Transaktionsunterstützung sowie eine (zumindest optionale) Persistenzeigenschaft von Agenten.

2. Integration in bestehende Systeme und Infrastrukturen: Existierende größere Anwendungen sollten einfach an Agentensysteme angebunden werden können. Insbesondere die Integration von Agenten in die WWW-Infrastruktur ist diesbezüglich wünschenswert.

3. Interoperabilität verschiedener Agentensysteme: Gegenwärtige Prototypen basieren auf unterschiedlichen Konzepten und verwenden oft unterschiedliche Sprachen für die Agenten (neben Java beispielsweise auch Tcl, Perl, Python und Oblique). Damit Agenten universell verwendbar sind, müssen jedoch gewisse Standards eingehalten werden und sollten andere Middleware-Konzepte (wie sie beispielsweise in CORBA definiert sind) weitestgehend integriert werden. Gegenwärtig gibt es u.a. Bemühungen der Object Management Group (OMG), eine Menge von Grundfunktionalitäten im Rahmen der "Mobile Agent System Interoperability Facility" (MASIF) [26] festzulegen.

7 Prototyp-Systeme

Von wenigen Ausnahmen abgesehen handelt es sich bei den bisher realisierten mobilen Agentensystemen um Prototypen, die sich in der Implementierung meist auf Teilbereiche des generellen Konzeptes konzentrieren. Zu diesen Ausnahmen gehört Telescript, welches bereits früh das Paradigma mobiler Agenten nachhaltig geprägt hat. Wir beschreiben nachfolgend kurz einige ausgewählte Agentensysteme; bezüglich Details verweisen wir allerdings auf die entsprechenden Veröffentlichungen. Einen Überblick zum generellen Forschungs- und Entwicklungsstand liefern auch Tagungsberichte [15] und einschlägige WWW-Seiten [24, 25].

Telescript [18] wurde ab 1991 von der Firma General Magic für den elektronischen Dienstleistungsmarkt konzipiert; die Entwicklung wurde dabei maßgeblich von verschiedenen größeren Firmen aus der Informations- und Telekommunikationsbranche unterstützt. Telescript stellt sowohl eine objektorientierte Agentenprogrammiersprache als auch eine Ausführungsumgebung dar. Agenten können mittels eines "go"-Befehls die Migration veranlassen und behalten dabei ihren Ausführungszustand. Verschiedene Architekturkonzepte (wie beispielsweise verschachtelt organisierbare "Plätze", an denen sich Agenten aufhalten und treffen können) und ein durchdachtes Sicherheitskonzept (u.a. Authorities, Permits, Tickets und Ressourcenbeschränkungen) sollen die Realisierung von Anwendungen unterstützen. Allerdings hat sich Telescript – vor allem wohl wegen des hohen Ressourcenbedarfs und der fehlenden Integration in die Internet-Infrastruktur sowie der kaum vorhandenen Unterstützung klassischer Sprachen und Datenbankschnittstellen – letztendlich bei Anwendern nicht durchsetzen können, so daß die Weiterentwicklung 1996 eingestellt wurde.

Odyssey [22] ist das Nachfolgesystem zu Telescript. Es hat ähnliche Konzepte, ist aber vollständig Java-basiert und nutzt Quasistandards wie DCOM, CORBA-IIOP und Java-RMI als Kommunikations- und Transportmechanismen. Die Weiterentwicklung durch die Firma General Magic ist allerdings auch bei diesem Produkt fraglich.

Aglets [10] von IBM ist ebenfalls Java-basiert; die zugehörige "Aglets Workbench" stellt neben einer Reihe von Entwicklungs- und Unterstützungswerkzeugen auch einen WWW-Zugang sowie Schnittstellen (APIs) zur Umgebung bereit. Das Aglets-System orientiert sich somit stärker an der Pragmatik gängiger Internet-Technologien. Für Agenten können Reisepläne (zu besuchende Ziele mit Anweisungen sowie Maßnahmen für einen Fehlerfall) spezifiziert werden, allerdings verlieren die Agenten ihren Ausführungskontext während der Migration. Die Kooperation zwischen Agenten kann über einen "white board"-Mechanismus geschehen, ansonsten stehen Nachrichten zur Kommu-

nikation zur Verfügung. Die Aglets-Entwicklungsgruppe ist auch hinsichtlich der Weiterentwicklung des Sicherheitskonzeptes stark engagiert.

Agent-Tcl [9] des Dartmouth College hat von vornherein nicht den Anspruch wie Telescript oder Aglets, kommerziell anwendbar zu sein. Es ist ein einfacher konzipiertes System, das die (relativ "langsame") Skriptsprache Tcl nutzt. Die Kommunikation geschieht über einen RPC-Mechanismus, der unmittelbar auf dem TCP/IP-Transportdienst aufsetzt. Die Migration ist aus Anwendersicht bequem: beim "agent-jump"-Befehl wird der gesamte Ausführungskontext zusammen mit dem Agenten migriert. Ebenfalls auf Tcl beruht *TKQML* [3] der University of Maryland, wobei KQML [5] zur Agentenkommunikation eingesetzt wird.

Generell ist die Sprache Java bei neueren Systemen sehr beliebt. Dies hängt nicht nur damit zusammen, daß in jüngster Zeit Java als "Internet-Programmiersprache" populär wurde, sondern daß Java mit dem Applet-Konzept, dem "Remote Method Invocation"-Prinzip, der Objektserialisierung und dem Sicherheitskonzept bereits wesentliche Elemente einer Agentenplattform enthält. Zu diesen neueren Systemen gehören beispielsweise *Voyager* [23] von ObjectSpace (Object-Request-Broker analog zu CORBA; Verzeichnisdienst; persistente Agenten; Kommunikation über Einzelnachrichten oder Broadcast), *Concordia* [19] von Mitsubishi (Integration von Datenbankanwendungen; im Voraus erstellbare Reisepläne für Agenten), *MOA* [13] von der Open Group (Nutzung von Java-Beans als Komponentenarchitektur) sowie *Kafka* [21] von Fujitsu.

In Deutschland existiert eine Reihe meist weit fortgeschrittener Projekte an Forschungsinstitutionen und Universitäten [25], so u.a. *Mole* [1] in Stuttgart (Java-basiert mit Telescript-ähnlicher Funktionalität und einer breiten Palette von Kommunikations- und Interaktionsmechanismen), *Ara* [14] in Kaiserslautern (transparente Migration einschließlich des Ausführungszustandes durch Modifikation des Sprachinterpreters von Tcl und Java; Unterstützung von C++), *ffMAIN* [12] in Frankfurt a.M. (Nutzung des HTTP-Protokolls zur Migration von Tcl- oder Perl-Agenten), *AMETAS* [27] ebenfalls in Frankfurt a.M. (Java-basiert; Anwendungszweck Netzmanagement), *WASP* [7] in Darmstadt (transparente Migration in Java; Nutzung von WWW-Servern als Agentenbasen), *MAGNA* [20] in Berlin, *ACE* [16] in Aachen sowie ein Forschungsprojekt zu mobilen Agenten in elektronischen Märkten in Hamburg [11].

8 Ausblick

Mobile Agenten stellen eine relativ neue Technologie dar; erst mit der allgemeinen Akzeptanz von Java und dem sich konkret abzeichnenden elektronischen Dienstemarkt im Internet entstand ab ca. 1996 eine größere Zahl von Prototypsystemen. Noch ist unklar, welche Bedeutung das an sich attraktive Paradigma in der Praxis erlangen wird (John Ousterhout bezeichnete mobile Agenten einmal als "solution in search of a problem") und welche Funktionalität (etwa: reine Codemobilität im Gegensatz zu Übermittlung des Ausführungskontextes) tatsächlich benötigt wird. Klar ist jedenfalls, daß für den praktischen Einsatz eine Integration in die Internet-Infrastruktur, Interoperabilität mit bestehenden Systemen sowie eine Lösung der wichtigsten Sicherheitsaspekte unabdingbar sind.

In jüngster Zeit zeichnet sich nicht nur ein verstärktes Forschungsinteresse ab, sondern es entsteht eine Reihe kommerzieller Systeme, die sich mit ihren Konzepten allerdings erst noch bewähren müssen. Sollte den mobilen Agenten aber trotzdem keine Zukunft als eigenständige Technologie bestimmt sein, so wird das Konzept sicherlich zusammen mit klassischen Prinzipien wie RPC, Botschaftenaustausch oder Ereignisdienst in Weiterentwicklungen von objektorientierten Middleware-Frameworks aufgehen (wie sich dies hinsichtlich CORBA bereits bei Voyager [23] abzeichnet) und dadurch der Anwendungsentwicklung alternativ zur Verfügung stehen.

Noch kaum behandelt wurden im Rahmen der Forschungen zu mobilen Agenten bisher Aspekte von höheren Koordinations- und Kooperationsprotokollen sowie Fragen der systemunabhängigen Repräsentation von Fakten, Wissen und Strategien. Dies ist dann von Bedeutung, wenn mobile Agen-

ten Aufgaben erledigen sollen, die nicht auf rein schematische Art ablaufen und sie dazu mit anderen Agenten in flexibler Weise interagieren müssen. Derartige Fragestellungen werden schon seit längerem von der Künstlichen Intelligenz, insbesondere vom Teilbereich "Distributed AI", im Rahmen von Multiagentensystemen und intelligenten Agenten untersucht. Zweifellos werden "mobile intelligente Agenten" für anspruchsvolle verteilte Anwendungen Aspekte beider Forschungsbereiche integrieren müssen.

9 Literatur

[1] Baumann, J.; Hohl, F.; Rothermel, K.; Straßer, M.:
Mole - Concepts of a Mobile Agent System. WWW Journal, Special issue on Applications and Techniques of Web Agents, 1998.

[2] Borenstein, N.:
Email With a Mind of Its Own: The Safe-Tcl Language for Enabled Mail.
In: Proc. IFIP Conference on Upper Layer Protocols, Architectures and Applications (ULPAA' 94), IFIP Transactions, 1994.

[3] Cost, R.S.; Soboroff, I.; Lakhani, J.; Finin, T.; Miller, E.; Nicholas, C.:
TKQML: A Scripting Tool for Building Agents. In: Singh, M.P.; Rao, A.S.; Wooldridge, M.J. (Hsg.): Intelligent Agents IV: Agent Theories, Architectures, and Languages (Proc. 4th Int. Workshop ATAL'97).
Springer-Verlag, LNCS 1365, 1998.

[4] Farmer, W.M.; Guttmann, J.D.; Swarup, V.:
Security for Mobile Agents: Issues and Requirements. Proc. 19th National Information Systems Security Conf. (NISSC96), 1996, S. 591-597.

[5] Finin, F; Labrou, Y.; Mayfield, J.:
KQML As an Agent Communication Language. In: Bradshaw, J. (Hsg.):
Software Agents, MIT Press, 1997.

[6] Fritzinger, J.S.; Mueller, M.:
Java Security. <http://java.sun.com/security/whitepaper.ps>

[7] Fünfrocken, S.:
How to Integrate Mobile Agents into Web Servers. Proc. Workshop on Collaborative Agents in Distributed Web Applications (WETICE'97), S. 94-99, 1997.

[8] Hohl, F.:
An Approach to Solve the Problem of Malicious Hosts. Fakultätsbericht 1997/03, Univ. Stuttgart, Fakultät Informatik, 1997.

[9] Kotz, D.; Gray, R.; Nog, S.; Rus, D.; Chawla, S.; Cybenko, G.:
Agent-Tcl: Targeting the Needs of Mobile Computers. IEEE Internet Computing 1 (1997), H. 4.

[10] Lange D.B.; Chang D.T.:
IBM Aglets Workbench - Programming Mobile Agents. In: Java, White Paper. IBM Corporation, Japan, 1996.
<http://www.trl.ibm.co.jp/aglets/whitepaper.htm>

[11] Liberman, B.; Griffel, M.; Merz, M.; Lamersdorf, W.:
Mobile Agents - How to Persist, Migrate, and Interact in Electronic Service Markets. In [15], S. 27-38.

[12] Lingnau A.; Drobnik O.; Dömel P.:
An HTTP-based Infrastructure for Mobile Agents. Proc. 4th Int. WWW Conference, Boston, 1995.

[13] Milojicic, D.; LaForge, W .; Chauhan, D.:
Mobile Objects and Agents (MOA). Proc. USENIX COOTS'98, Santa Fe, 1998. <http://www.camb.opengroup.org/RI/java/moa/coots2.fr.ps>

[14] Peine, H.; Stolpmann, T.:
The Architecture of the Ara Platform for Mobile Agents. In [15], S. 50-61.

[15] Rothermel, K.; Popescu-Zeletin, R. (Hsg.):
Mobile Agents (Proc. 1st Int. Workshop). Springer-Verlag, LNCS 1219, 1997.

[16] Sang-Bum Park, A.; Leuker, S.:
A Multi-Agent Architecture Supporting Services Access. In [15] S. 62-73.

[17] US patent no. 5603031:
System and Method for Distributed Computation Based upon the Movement, Execution, and Interaction of Processes in a Network.

[18] White, J.E.:
Telescript Technology: An Introduction to the Language. General Magic Inc., White Paper, 1995.

[19] Wong, D.; Paciorek, N.; Walsh, T.:
Concordia: An Infrastructure for Collaborating Mobile Agents. In [15] S. 86-97.

[20] http://www.fokus.gmd.de/oks/research/magna_e.html

[21] http://www.fujitsu.co.jp/hypertext/free/kafka/

[22] http://www.genmagic.com/agents/odyssey.html ;
http://www.genmagic.com/agents/

[23] http://www.objectspace.com/voyager/

[24] http://www.informatik.tu-darmstadt.de/VS/DagAgs.html

[25] http://www.informatik.tu-darmstadt.de/~fuenf/work/agenten/agenten.html

[26] http://www.camb.opengroup.org/RI/MAF/maf_pres/mafprese.htm ;
http://www.camb.opengroup.org/~dejan/maf/draft10/draft10.book.ps

[27] http://www.vsb.cs.uni-frankfurt.de/~zapf/projekt.html

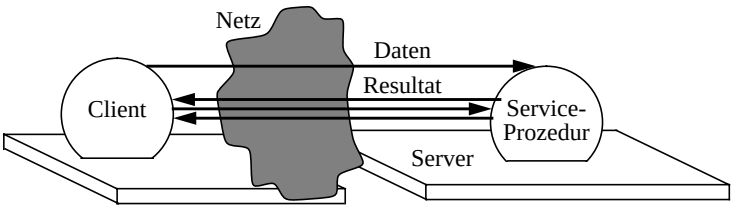

Abb. 1: Client-Server-Struktur mit Remote-Procedure-Call

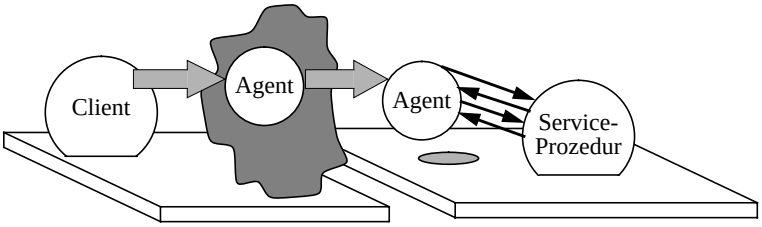

Abb. 2:Versenden eines Agenten

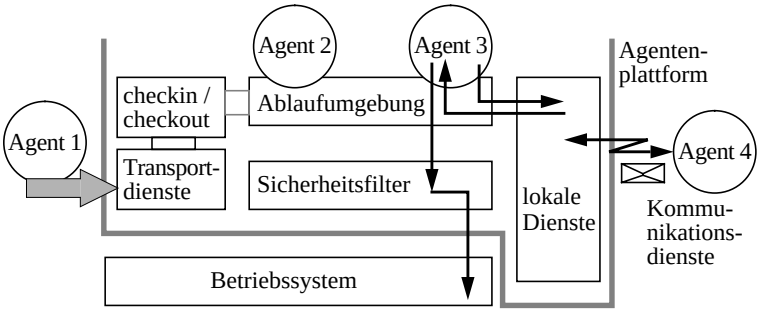

Abb. 3: Wirtssystem ("host") mit Betriebssystem und Agentenplattform