Zeitschrift: Comtec : Informations- und Telekommunikationstechnologie = information and telecommunication technology

Herausgeber: Swisscom

Band: 76 (1998)

Heft: 10

Artikel: Neue, ungeahnte Potentiale schaffen

Autor: Gisiger, Hans Peter

DOI: https://doi.org/10.5169/seals-877328

Aus den Explorationsprogrammen von Corporate Technology (6)

Agenten-Technologie

Neue, ungeahnte Potentiale schaffen

Das Explorationsprogramm* EP9706 «Operational Processes & Customer Care» untersucht den Einsatz neuer Technologien auf die Prozesse «Service Order & Delivery» und «Service Assurance» unter den Aspekten der Dienstentwicklung und der Kosteneffizienz.

*Explorationsprogramme werden von Corporate Technology im Auftrag der Konzernleitung durchgeführt und werden regelmässigen Reviews unterzogen. Die Aktivitäten haben einen mittel- bis langfristigen Zeithorizont, 2–7 Jahre, je nach Gebiet.

Seit den 90er Jahren erleben wir eine Inflation von Agenten, Agentenkonzepten und -technologien. Das Aufkommen von Agenten gibt Anlass zu vielen Diskussionen. Neu wird jedes Programm mit «Agent» bezeichnet, und eine echte «Agento-Mania» ist ausgebrochen. Was ist nun wirklich ein Agent? Worin unterscheiden sich Agenten von allgemeinen Programmen? Sind Agenten Objekte mit intelligentem Verhalten? Dieser Artikel befasst sich mit dem Begriff «Agent» und versucht, ihn an den typischen Eigenschaften zu klären und zu definieren. Er soll Chancen der Agententechnologie aufzeigen, Vor- und Nachteile beleuchten und verschiedene Erscheinungsformen von Agenten erklären. Sodann wird versucht, eine Taxonomie, eine Einteilung der Agenten in ein Schema, vorzunehmen. Es sollen heute bestehende und erhältliche Agenten-Frameworks und Templates vorgestellt und deren Nutzen erklärt werden. Im Rahmen des Explorationsprogrammes EP9706 wurden bei Corporate Technology, Swisscom, zwei Projekte zum Thema Agenten gestartet. Die Zielsetzungen und Erwartungen an diese beiden Projekte werden kurz vorgestellt.

Was sind Agenten?

Diese Frage wird immer wieder gestellt. Im Laufe der Zeit wurden verschiedene Versuche unternommen, den Begriff «Agent» mehr oder weniger formal zu definieren. Dazu ein paar Ansätze zu einer Definition:

HANS PETER GISIGER, BERN

Wir beginnen mit einer Definition aus dem *Webster New World Dictionary.*
– *Agent:* «A person or thing that acts or is capable of acting or is empowered to act, for another.»
Diese Definition beschreibt zwei Eigenschaften, die «Intelligente Agenten» auf einem Computer System immer aufweisen:
1. Agenten tun etwas, und
2. Agenten tun etwas im Auftrag von jemandem.

Ein weiterer Begriff «Software Agent» wird in diesem Zusammenhang oft so definiert:
– *Software Agent:* «A computing entity that performs user delegated tasks autonomously.»

Weitere Versuche, Agenten zu definieren, führten zu folgenden Definitionen:
– *The IBM Agent:* «Intelligent agents are software entities that carry out some set of operations on behalf of a user or another program with some degree of independence or autonomy, and in so doing, employ some knowledge or representation of the user's goals or desires.»
– *The Maes Agent:* «Autonomous agents are computational systems that inhabit some complex dynamic environment, sense and act autonomously in this environment, and by doing so realize a set of goals or tasks for which they are designed.»
– The Wooldridge-Jennings Agent: «... a hardware of (more usually) software-based computer system that enjoys the following properities:
Autonomy: agents operate without the direct intervention of humans or others, and have some kind of control over their actions and internal state;
Social ability: agents interact with other agents (and possibly humans) via some kind of agent-communication language;
Reactivity: agents perceive their environment, (which may be the physical world, a user via a graphical user interface, a collection of other agents, the Internet, or perhaps all of these combined), and respond in a timely fashion to changes that occur in it;
Pro-activeness: agents do not simply act in response to their environment, they are able to exhibit goal-oriented behavior by taking the initiative.»
– The Franklin-Graesser Agent: «... is a system situated within and a part of an environment that senses that environment and acts on it, over time, in pursuit of its own agenda and so as to effect what it senses in the future.»

Die aufgeführten Definitionen erwähnen *aus Sicht einer Applikation* eine ganze Anzahl von *Eigenschaften,* die ein Agent erfüllen sollte. Einige dieser Eigenschaften sind *notwendig,* andere sind hilfreiche und interessante *Erweiterungen* des Agentenkonzeptes. Im Zentrum stehen dabei die Eigenschaften *Autonomie, Reaktivität, Pro-Aktivität* und *Kommunikation.*
– *Autonomie:* Ein Agent übt ohne direkte Interventionen von aussen *Kontrolle* über seine eigenen Aktionen aus. Diese Autonomie kann von der Fähigkeit ein «nächtliches Backup» auszulösen bis zur «Aushandlung eines Preises für ein Produkt» im Namen des Benutzers reichen.
– *Reaktivität* (Sensing and Acting): Ein Agent muss in der Lage sein, Ereignisse oder Änderungen in seiner Umgebung zu erfassen (Sensing, Monitoring) und

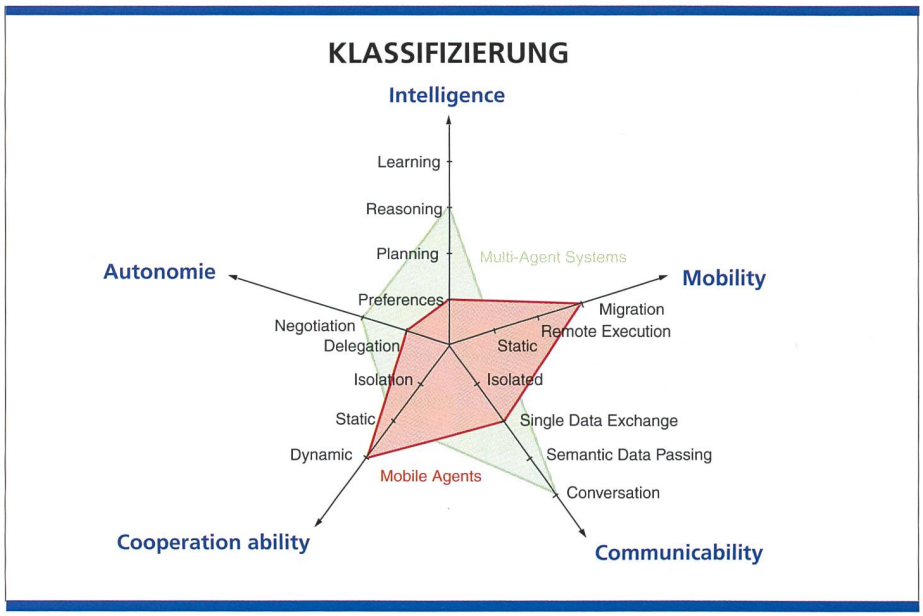

Bild 1. Agenten-Klassifizierungsschema.

zeitgerecht und *zielgerichtet* darauf zu reagieren (Actuation).
- *Pro-Aktivität:* Ein Agent handelt jedoch nicht bloss aufgrund von Ereignissen in seiner Umgebung, sondern ist in der Lage vorausschauend nach einer eigenen Strategie vorzugehen.
- *Kommunikation* (Social Ability): Der Agent muss fähig sein mit dem Benutzer (Menschen), der ihm einen Auftrag erteilt oder den aktuellen Status abfragt, oder mit anderen Agenten zu kommunizieren. Oft wird zur Kommunikation mit anderen Agenten eine eigene Agenten-Kommunikationssprache (z. B. ACL) verwendet.

Fähigkeiten von Agenten

Einen Überblick über die *Fähigkeiten* von Agenten gibt das Klassifikationsschema in Bild 1. Darin wird ein Agent nach folgenden drei Kriterien (Dimensionen) klassifiziert: *Intelligence, Mobility* und *Agency.* Die Agency lässt sich wiederum in die Unterbereiche *Kommunikationsfähigkeit, Kooperationsfähigkeit* und *Autonomie* unterteilen:

Intelligence

Unter dem Begriff *Intelligenz* fasst man die Fähigkeiten oder Techniken von Komponenten zusammen, die *Benutzer-Präferenzen* berücksichtigen, die *Schlussfolgerungen* erlauben, die *Planungs*aktivitäten ermöglichen oder *Lernfähigkeit* unterstützen:
- *Präferenzen* geben dem Agenten ein vom Benutzer gewünschtes Verhaltensmuster vor. Präferenzen können direkt im Code eines Agenten programmiert sein. Flexibler werden sie als Regeln vorgegeben und mit Hilfe eines Interferenz-Mechansimus verarbeitet.
- *Reasoning* beruht auf einem Interferenzmechanismus mit dem Ziel aus vorgegebenen Fakten (Ereignissen), Regeln und Daten *Schlussfolgerungen* zu ziehen, Entscheidungen zu fällen und Aktionen erfolgreich durchzuführen.
- *Planung* ermöglicht einem Agenten Aktionen durchzuführen, die ihn seinem Ziel näherbringen. Dazu unterhält ein Agent ein eigenes internes «Welt-Modell» mit den Bedürfnissen und Anforderungen des Benutzers. Diese Technik unterstützt die *Pro-Aktivität* eines Agenten.
- *Lernfähigkeit* kann das Verhalten eines Agenten unter Berücksichtigung der Anforderungen und Ziele des Benutzers und der Verfügbarkeit von Ressourcen als Resultat von Experimenten oder gemachten Erfahrungen dynamisch verändern. Ein solcher Agent kann beispielsweise neue Beziehungen, Verbindungen und Konzepte unabhängig vom Eingriff des Benutzers erkennen und, ausgehend vom bestehenden Wissen, Entscheidungen fällen.

Mobility

Um spezielle Aufgaben zu bearbeiten, können Agenten durch ein Netzwerk zu *Remote Sites* transportiert werden. Normalerweise wird dazu eine spezielle Run-Time Umgebung (Agenten-Plattform) verwendet. Die Fähigkeit *Mobilität* unterstützt, ergänzt oder ersetzt die Kommunikation eines Agenten. Grundsätzlich kann zwischen zwei Stufen von *Agenten-Mobilität* unterschieden werden:
- *Statische* Agenten können zwar nicht durch ein Netz migrieren, können jedoch auf remote Ressourcen beispielsweise via FTP zugreifen.
- *Remote Execution* (mobility without state) ermöglicht, einen Agenten zu einem Remote System zu übertragen, wo sein Programm-Code mit seinen Daten aktiviert und vollständig ausgeführt wird.
- *Migration* ist die Fähigkeit eines *aktiven Agenten* während der Bearbeitung seines Auftrages seine Aufgabe zu *suspendieren,* sich selber (Programm Code, Daten und Ausführungszustand) an einen anderen Knoten des Netzwerkes zu transportieren und dort die Ausführung der unterbrochenen Aufgabe am Abbruchpunkt wieder aufzunehmen und weiterzuführen.

Kommunikationsfähigkeit

Die Kommunikationsfähigkeit eines Agenten beschreibt den Komplexitätsgrad der Interaktionen eines Agenten mit einer Applikation, einem Service, einer Einrichtung, dem Benutzer oder anderen Agenten.
- *Isolierte* Agenten habe keine Fähigkeiten zu interagieren.
- *Einfacher Datenaustausch* kann mit anderen Entitäten auf der Basis einer minimalen Interaktionssemantik erfolgen. Service-Aufrufe mit geringer Kontrollinformation oder Input-Daten sind charakteristisch dafür. Ein Agent mit dieser Fähigkeit kann Änderungen in seiner Umgebung wahrnehmen und darauf reagieren.
- *Semantische Datenübergabe* unterstützt den Austausch von Daten mit anderen Entitäten. Um die Aufgabe zu erfüllen, muss die Semantik der Daten jedoch interpretiert werden. Dazu wird ein Domain-Vokabular der Applikation verwendet.
- *Konversation* ist die höchste Stufe der Kommunikation zwischen Agenten. Um Konversation zu betreiben, müssen folgende Aufgaben gelöst sein: Heterogenität zwischen Agenten, Austausch von Wissen und Standortkontrolle. Agenten kommunizieren untereinander indem sie Meldungen in einer ausdruckstarken Sprache mit agentenunabhängiger Semantik austauschen.

Autonomie
Die Autonomie eines Agenten wird durch sein Verhalten auf Anfragen oder Aufträge reflektiert: Wenn er nur Aufträge ausführt, ist seine Autonomie beschränkt. Folgt er eigenen Zielen, so kann er externe Aufträge akzeptieren oder zurückweisen, aber auch Aufgaben ohne Auftrag ausführen. Dieser Begriff der Autonomie heisst Ausführungsautonomie (Execution Autonomy). Zwei Stufen werden unterschieden:
- Wird die Autonomie *durch die Delegation beschränkt,* führt der Agent Aufgaben im Auftrag des Benutzers durch. Die Aufgabe wird durch den Benutzer oder andere Agenten definiert.
- Wenn der Agent die Fähigkeit hat, auf Anfragen nach eigenem Ermessen zu handeln (Akzeptieren, Zurückweisen Aushandeln), spricht man von *Negotiation.*

Kooperationsfähigkeit
Die Kooperationsfähigkeit eines Agenten beschreibt den Grad an Dynamik der Beziehungen zu anderen Entitäten mit dem Ziel eine Aufgabe gemeinsam zu lösen.
- *Isolation* eines Agenten besteht dann, wenn er seine Aufgabe selber ausführt und keine Möglichkeit hat mit anderen Entitäten zu koperieren
- *Statische Kooperation* existiert dann, wenn ein Agent mit andern Entitäten kooperieren kann, sich seine Umgebung, seine Kooperationspartner sowie die Koopertionsregeln jedoch nicht oder selten ändern.
- *Dynamische Kooperation* besteht dann, wenn ein Agent in eine dynamisch ändernde Umgebung eingebunden ist, und Komponenten sowie Kooperationsregeln laufend wechseln. Agenten sind gezwungen laufend zu evaluieren und Kompromisse zu treffen. Typischerweise können Agenten in einer solchen Umgebung erst nach einer Negotiations-Phase mit einer Kooperation beginnen.

Die *Ursprünge* der Agenten-Technologie, insbesondere jene der Intelligenten Agenten, gehen zurück in die *Computer-Intelligenz* (AI), das *Software Engineering* (SE) und das Gebiet der *Benutzerschnittstellen* (GUIs). Die Computer-Intelligenz für Intelligente Agenten basiert auf *Intentionalen Systemen, Reasoning Theorie* und *Neuronalen Netzen.* Das Software Engineering überdeckt das Gebiet des *on-line Monitoring, high-level*

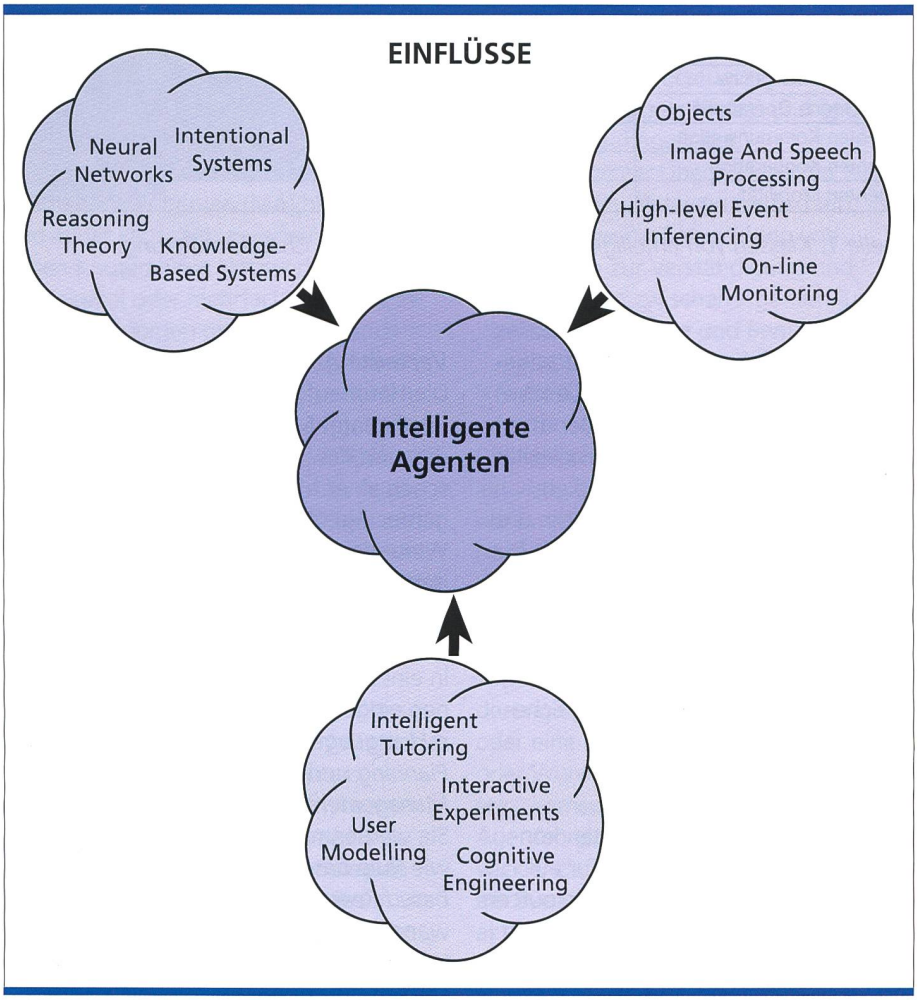

Bild 2. Einflüsse auf Agententechnologie.

Interference, Remote Actuation, Bild- und Sprachverarbeitung und *der verteilten Objekte.* Mensch-Maschinen-Interaktion für Intelligente Agenten kommt aus dem Feld des *Cognitiv-Engineering, der Benutzer-Modellierung,* aus *Mensch-Maschinen-Experimenten, intelligenten Tutoring Systemen* und *Computer Vision* wie in Bild 2 dargestellt.
Nachdem die grundlegenden Charakteristiken von Agenten eingeführt worden sind, können die verschiedenen Agententechnologien besser klassifiziert werden. Neben *Single-Agent*-Systemen können auch *Multi-Agent*-Systeme aufgebaut werden. Single-Agent-Systeme können weiter in sog. *Local Agents* und *Networked Agents* unterteilt werden. Multi-Agent-Systeme umfassen *Distributed Intelligent Agent* Systeme und *Mobile* Agenten.

Übersicht über die vier Klassen von Agenten
In Single-Agent-Systemen führt ein Agent im Auftrag eines Benutzers oder eines Prozesses eine Aufgabe aus und kommuniziert dabei sowohl mit dem Benutzer als auch mit System Ressourcen, jedoch nicht mit anderen Agenten. Agenten in Multi-Agent-Systemen können zur Erreichung ihrer individuellen Ziele intensiv miteinander zusammenarbeiten und sich absprechen, interagieren aber auch mit Benutzern und System Ressourcen.
Tabelle 1 gibt einen Überblick über die vier Klassen von (Intelligenten) Agenten und ihre Charakteristiken.

Lokale Agenten
Lokale Agenten greifen nur auf lokale Ressourcen zu. Sie arbeiten normalerweise als «Advisory Agents» (z. B. Intelligente Help Systeme) oder «Personal Assistants», die den Menschen bei seiner täglichen Arbeit unterstützen. Das Hauptziel dieser Agenten ist die Zusammenarbeit mit dem Menschen, und das Hauptgewicht liegt somit auch auf der Mensch-Agent-Interaktion. Aus diesem Grund bezeichnet man diese Agenten auch als «Intelligente Interfaces» oder «Interface Agenten». Lokale Agenten

Klassen / Attribute	Lokale Agenten	Netzwerk-Agenten	DAI-basierte Agenten	Mobile Agenten
Agenten Intelligence				
Asynchrone Operationen				
Agenten Kommunikation				
Agenten Kooperation				
Agenten Mobilität				

Tabelle 1. Klassen von (Intelligenten) Agenten und deren Charakterisierung.

können den Benutzer in vielen verschiedenen Aufgaben unterstützen: sie können Aufgaben im Auftrag des Benutzers übernehmen und dabei die Komplexität verschiedener Aufgaben verstecken, oder sie können asynchron arbeiten und den Benutzer trainieren oder lehren. Die Menge der möglichen Aufgaben, bei denen ein lokaler Agent den Benutzer unterstützen kann, ist unbegrenzt: lokales Suchen und Filtern von Information, lokale Mail-Verwaltung, Meeting Schedules und ähnliches.

Netzwerk-Agenten

In Gegensatz zu lokalen Agenten können Netzwerk-Agenten nicht nur auf lokale, sondern auch Remote Ressourcen zugreifen und haben somit detailliertes Wissen über die Netzwerk-Infrastruktur und Services im Netz. Im Gegensatz zu Multi-Agenten-Systemen können Netzwerk-Agenten nicht miteinander kooperieren. Viele Netzwerk-Agenten arbeiten als «Personal Assistants», stellen aber nicht nur dem Benutzer ein intelligentes Interface zur Verfügung, sondern benutzen verschieden Dienste über das Netzwerk. Im Gegensatz zu «Smart Mailboxes», welche aufgrund von User-Präferenzen auswählen können, können beispielsweise «Search Engines» auf Auftrag des Benutzers verfügbares Wissen im Netzwerk suchen. Diese Agenten werden unter den Namen «KnowBot» oder «Softbot» gehandelt. Bei der Informationssuche verstecken diese Agenten nicht nur die Komplexität der Netzwerk-Infrastruktur, sondern machen für den Benutzer zusätzlich die Heterogenität der Informationen und Protokolle, verteilt über das Netzwerk, unsichtbar. Ein Benutzer kann seine Informationsbedürfnisse abstrakt spezifizieren, und der Netzwerk Agent sammelt, kombiniert, filtert und präsentiert die gewünschte Information autonom aus mehreren Informationsquellen im Netzwerk. Dieser Agententyp ist speziell im Internet relevant.

Verteilte AI-basierte Agenten

Die Hauptaufgabe verteilter, AI-basierter intelligenter Multi-Agenten ist die Koordination des intelligenten Verhaltens zwischen einer Menge autonomer Intelligenter Agenten, das heisst, die Art und Weise der Koordination ihres Wissens, ihrer Ziele, Skills und Pläne um gemeinsam etwas zu unternehmen oder ein Problem zu lösen. Diese Agenten werden in einer breiten Palette von Applikationen eingesetzt, wie beispielsweise *Natural Language Parsing, Transportation Planning* und im *Telekommunikations-Management.*

Sie werden mit Hilfe von AI Techniken wie *Rule-based* oder *Case-and Example-based Reasoning* entwickelt. Die angewandte Wissensdarstellung und die Interferenz-Mechanismen sind grundsätzlich die selben wie in gewöhnlichen Knowledge-based Systemen. In verteilten AI-basierten Systemen kann ein Agent mit dem Benutzer, mit den System Ressourcen oder mit anderen Agenten interagieren, kommunizieren oder kooperieren.

Viele der vorgeschlagenen inter-agent Verhandlungs- und Zusammenarbeitsprotokolle basieren auf den Konzepten von Kontraktnetzen und der Speech Act Theory. Diese Theorie wird zur Definition der Semantik von Meldungen verwendet. Typische Beispiele derartiger Sprachen sind die *Knowledge Query and Manipulation Language* (KQML), das *Knowledge Interchange Format* (KIF) und die *Agent Communication Language* (ACL).

Mobile Agenten

Mobile Agenten treten in verschiedenen Varianten und mit unterschiedlichen Fähigkeiten auf:

- Ein Agent kann *remote* an ein System übertragen und dort eingesetzt werden. Während der Ausführung seiner Aufgabe verhält er sich wie ein Server oder ein Client, das heisst, eine rege Kommunikation zwischen verschiedenen Systemkomponenten kann stattfinden. Ist der Auftrag ausgeführt, terminiert der Agent normalerweise.
- Meist wird jedoch unter Mobilen Agenten eine Reinform von Mobilität, die sog. *Migration* verstanden. Dabei können Agenten ihre aktuelle Arbeit auf einem Knoten suspendieren, sich selber, das heisst Programm, Code und Ausführungszustand, an einen anderen Knoten transportieren und dort die Ausführung der unterbrochenen Aufgabe am Abbruchpunkt wieder aufnehmen und weiterführen. Wann und wohin ein Agent migriert, ist entweder Bestandteil seines Auftrages (er erhält einen Fahrplan), oder die Entscheidung wird selbständig vom Agenten getroffen. Mobile Agenten können auf ihrer Reise neue Agenten kreieren und starten (launching), um beispielsweise Daten an einen Client auszuliefern oder Subtasks auszuführen. Dieser Ansatz mit Migration unterscheidet sich fundamental vom klassischen Ansatz der Client/Server-Applikationen, der auf RPCs basiert.

Nutzen von Agenten

Der Nutzen von Agenten liegt in ihren Fähigkeiten spezifische Aufgaben (Task Skill) zu übernehmen. Der Nutzen lässt sich in verschiedene Kategorien unterteilen:

- Automatisierung: *Repetitives Verhalten, ähnliches Verhalten einer Gruppe* von Benutzern und *repetitives, sequentielles Verhalten* einer Anzahl von Benutzern in einem Workflow können beim Einsatz von Agenten eine *erhöhte Produktivität* bringen.
- Customization: Ein Agent nutzt die Vorteile der Customization, indem er genau die Informationen, die persönliche Informations- und Interaktionspräferenzen des Benutzers betreffen, sammelt, ausfiltert und zur Verfügung stellt. Dadurch wird der Informations-Overload verhindert oder zumindest reduziert.
- Notifikation: Ein Agent, der Notifikation an einen Benutzer unterstützt, befreit den Benutzer vom Beobachten (Monitoring) wichtiger Ereignisse und produziert dadurch einen deutlich geringeren Workload. So kann beispielsweise ein Agent interessante Web Sites beobachten und dem Benutzer Änderungen durch Notifikation mitteilen.
- Learning: Ein Agent mit Lernfähigkeiten kann Aufgaben, die automatisiert werden können, oder Präferenzen, die

zur Customization verwendet werden können, autonom erkennen:
1. Erkennen und anbieten der Automatisierung repetitiver Aufgaben
2. Ähnliche Attribute einer Gruppe von Benutzern erkennen und eine Gruppen-Charakteristik definieren
3. Ähnliches Verhalten einer Gruppe von Benutzern erkennen und Massnahmen identifizieren, um die Pro-Aktivität der ganzen Gruppe zu erhöhen
4. Erkennen wiederkehrender, sequentieller Verhaltensweisen einer Gruppe von Benutzern in einem Workflow, und Anbieten der Automatisierung zur Erleichterung ihrer Arbeit

- Tutoring: Dank der Fähigkeit Ereignisse zu beobachten und davon abzuleiten, welche Informationen der Benutzer benötigt, kann ein Agent mit Tutoring-Fähigkeiten diesen in seinem Kontext coachen. Dadurch wird die Trainingszeit reduziert.
- Messaging: Ein Messaging Agent ermöglicht Benutzern Aufgaben offline von einem Remote Standort auszuführen. Beispiele dazu sind Mobile Agenten, die sich selber von Ort zu Ort transferieren können und zur Lösung ihrer Aufgabe mit anderen Agenten zusammenarbeiten.

Intelligente Agenten

Überblick

Intelligente Agenten (IA) sind Software Entitäten, deren *Intelligence* und *Agency* einen gewissen Grad überschreiten (Bild 1).
Intelligente Agenten (IA) sind in der Lage, sich intelligent, das heisst *reaktiv, pro-aktiv* oder *adaptiv* zu verhalten. Dazu gehört es, Schlussfolgerungen zu ziehen (Reasoning, Reaktivität), Planungaufgaben zu übernehmen (Heuristiken, Pro-Aktivität, Optimierungen) und aus gemachten Erfahrungen zu lernen (Learning, Adaptivität). Mobile Agenten (MA) hingegen sind in der Lage, sich zwischen verschiedenen Knoten eines Netzwerkes zu bewegen (einmal oder mehrfach). Obwohl Intelligente Agenten grundsätzlich auch mobil oder Mobile Agenten auch intelligent sein können, werden diese beiden Aspekte meist klar voneinander getrennt. Für Intelligente Agenten ist Mobilität meist kein Thema und für Mobile Agenten ist eben die Mobilität das Schlüsselattribut. Aus diesem Grund wird in diesem und im folgenden Kapitel, wie auch in der Literatur, getrennt auf diese beiden Typen von Agenten eingegangen.
Bei Intelligenten Agenten sind somit nur die beiden Dimensionen der Klassifikation *Intelligence* und *Agency* nützlich zur Charakterisierung und Unterscheidung verschiedener Arten von Agenten. Diese beiden Dimensionen lassen sich verschiedenen Software-Technologien zuordnen. Diesen können wiederum vier sog. Technologie-Faktoren zugeordnet werden (Bild 3):

- *Machinery* und *Content* sind Faktoren der *Intelligence,*
- *Access* und *Security* sind Faktoren der *Agency.*

Machinery bezieht sich auf *Engines* verschiedener Art, hauptsächlich entwickelt im Umfeld der künstlichen Intelligenz, sie unterstützt damit verschiedene Stufen der Intelligenz. Diese umfassen:

- Verschiedene Formen von *Interferenz*
- Verschiedene Formen von *Lernen*
- Werkzeuge zur Erzeugung und Modifikation von *Regeln* (Rules) und anderer Kenntnisse (knowledge)
- Werkzeuge zur Überprüfung der Regeln (Konsistenz, Widerspruchsfreiheit)
- Werkzeuge zur Entwicklung von Strategien zur Verständigung und Kooperation zwischen Agenten, und zwischen Agenten und Benutzer (z. B. Agent Communication Languages, Onthologies).

Die Machinery wird durch Ereignisse aktiviert. Solche Ereignisse treten ein, wenn Agenten aufgrund neuer Erkenntnisse (Wissen oder Inhalte) eine Änderung der Umgebung signalisieren. Ein Ereignis kann das Erreichen eines festen Zeitpunktes oder das Ablaufen eines Timers, die Ankunft eines Mail, die Änderung eines Wertes in einer Datenbank oder eine neue Vorgabe (goals, beliefs) oder Wunsch (desires) des Benutzers sein. Immer, wenn in der Umgebung des Agenten eine vom Benutzer oder Entwickler definierte Änderung eintritt, wird ein Event getriggert und die Machinery aktiviert. Bei der Aktivierung ist zu entscheiden, ob das eingetretene Ereignis eine entsprechende Aktion bewirkt. Aktionen rufen dabei eine Funktion der Applikation auf, das heisst die Machinery muss die Funktionen der entsprechenden Applikation ansprechen können. Somit muss ein Agent, der Electronic Mail verwaltet, über die Ankunft von Mails durch Ereignisse benachrichtigt werden, und er muss die Möglichkeit haben, die Benutzer-Mailbox direkt zu manipulieren.

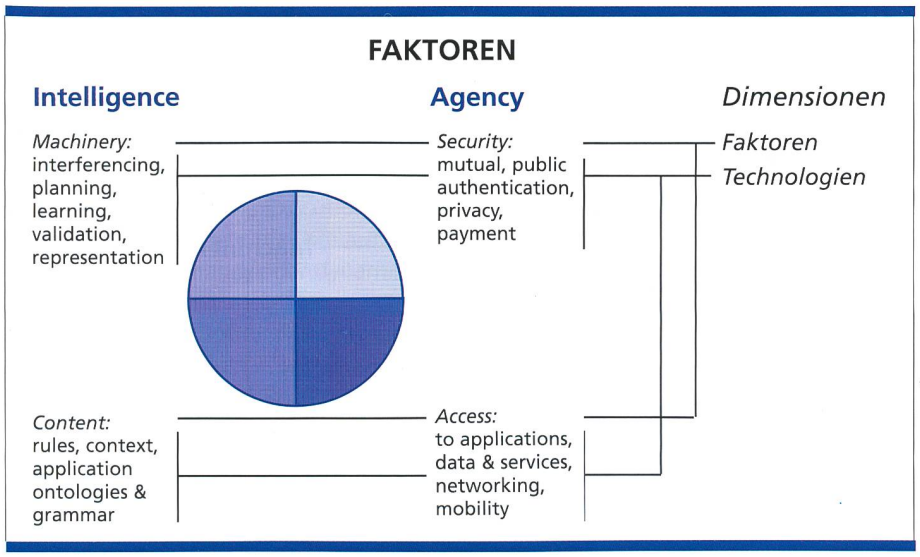

Bild 3. Technologie-Faktoren der Intelligenten Agenten.

Unter *Content* verstehen wir die Daten, welche die Machinery für *Reasoning* und *Learning* verwendet. Der Content besteht zunächst aus Regeln, welche die Präferenzen oder die Policy des Benutzers beschreiben. Content kann aber auch die interpretierbare Darstellung der Welt sein, so dass Agenten über Waren, Produkte und Dienste miteinander kommunizieren können. Zum Content eines Agenten gehört jede Art strukturierten und nicht-strukturierten relevanten Wissens. Man beachte, dass der grösste Teil verfügbarer Information in unstrukturierter Form vorliegt, so dass Agenten mit freien Text-Formaten umgehen können müssen. Somit gehören *Filtering* und *Natural Language Support* zu den grundlegenden Skills eines Intelligenten

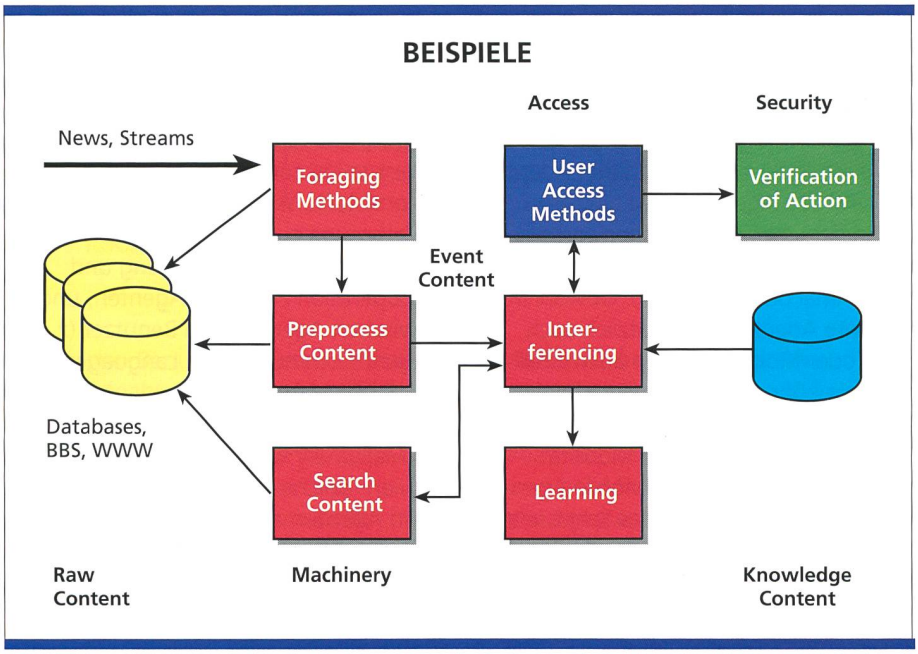

Bild 4. Beispiel eines News Processing Agent.

Agenten. Agenten müssen aber auch in der Lage sein, durch Beobachtung aus dem Verhalten des Benutzers zu lernen oder andere Heuristiken anzuwenden.
Access ermöglicht der Machinery, Content zu erkennen und darauf zuzugreifen und Aktionen als Resultat des Reasoning auszulösen. Access beschreibt den Grad, mit dem ein Agent mit seiner Umwelt interagieren kann. Er muss in seiner Welt Ereignisse erkennen können und entsprechend darauf mit Aktionen reagieren. Vielfach werden Agenten zur Integration von Legacy-Applikationen eingesetzt. Ein grosser Aufwand wird deshalb dafür geleistet, Agenten Interaktionen mit Applikationen zu ermöglichen, die keine geeigneten APIs zu internen Funktionen und Daten besitzen. In lokalen Umgebungen wird der Access durch direkte Anbindung an APIs, durch Shared Memory, Datenbanken oder über das File-System erreicht. Handelt es sich um ein verteiltes System, werden Remote Techniken (RPCs) oder Standard-Protokolle wie beispielsweise HTTP verwendet. Der Access in einem verteilten System kann aber auch mit Hilfe von Mobilen Agenten realisiert werden.
Security ist ein Thema, das weit über den Horizont der Agenten hinausgeht. Der Erfolg von Agenten hängt stark von der Effektivität der Sicherheit ab und stellt auch neue Anforderungen an diese Dienste. Da ein Agent im Auftrag eines Benutzers im öffentlichen Netz (z. B. im elektronischen Handel) aktiv werden kann, muss der Agent auch die rechtliche, zertifizierte Autorität und Verantwortung des Benutzers übernehmen.

Beispiel

Zur Illustration dieses einfachen Modells eines Intelligenten Agenten betrachten wir das Beispiel eines *News Retrieval and Filtering Agents* in Bild 4. Der Agent verwendet seine Access-Methoden um lokal und remote in Datenbanken nach geeigneten Inhalten (Content) zu stöbern. Dabei können neue Streams aufgesetzt, ein *Retrieval* von Bulletin Boards oder ein *Spider* im Web eingesetzt werden. Die erhaltenen Inhalte sind wahrscheinlich bereits durch gezieltes Suchen gefiltert. Der Agent kann nun durch Einsatz seiner Such- und Sprachverarbeitungs-Maschinerie Schlüsselwörter oder Signaturen aus den Inhalten extrahieren. Mit Hilfe der Reasoning- und Interferenz-Maschinerie kann nun entschieden werden, was mit dem abstrahierten Inhalt (Content) geschehen soll. Dieser Vorgang kombiniert den Event Content mit den Regeln und dem Wissen, das vom Nutzer zur Verfügung gestellt wird. Wird in diesem Arbeitsschritt etwas Wichtiges gefunden, so wird der gesamte Content weiterverarbeitet. Am Schluss wird durch die Maschinerie entschieden, ob als Reaktion auf den neuen Content eine Aktion ausgeführt werden soll – beispielsweise der Benutzer wird informiert. Diese Aktion wird jedoch zuerst durch Security-Funktionen verifiziert. Bestätigt der Benutzer die Wichtigkeit des Ereignisses durch schnelles Reagieren, kann der Agent seine Learning Maschinerie einsetzen um derartige Inhalte und Ereignisse neu zu gewichten.
Machinery, Content, Access und Security sind die Kategorien von Technologie-Faktoren bei der Entwicklung und beim Betrieb von Intelligenten Agenten. Sie stehen jedoch nicht für ein Entwicklungsvorgehen, sondern sind bloss hilfreich zur Unterteilung eines komplizierten Themas.

Konzeptuelles Modell

Das generelle, konzeptuelle Modell

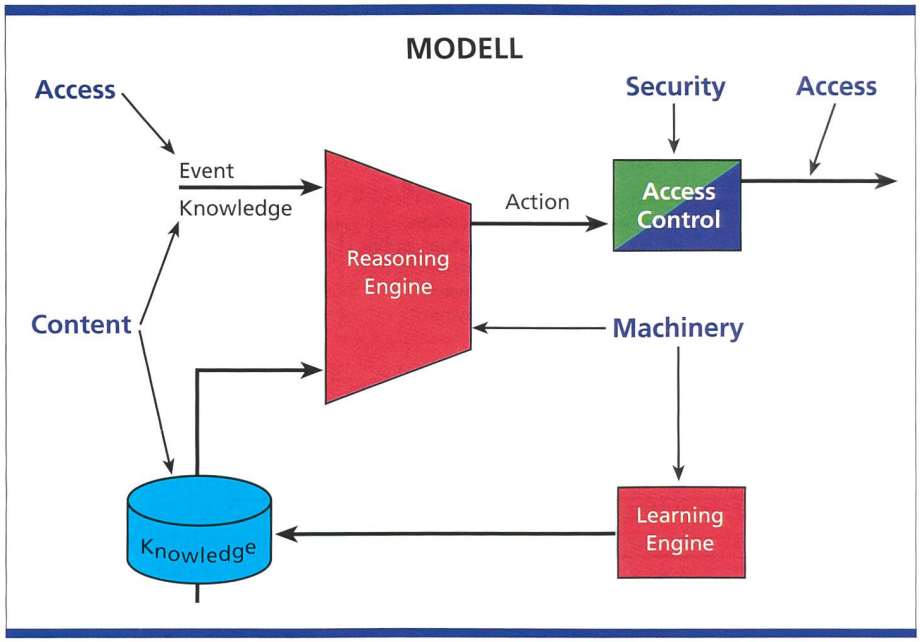

Bild 5. Konzeptuelles Modell eines Intelligenten Agenten.

eines intelligenten Agenten zeigt Bild 5. Wie illustriert, besteht die prinzipielle Aufgabe eines Intelligenten Agenten darin, existierendes Wissen mit dem Erwerb neuen, durch Ereignisse in der Umgebung erzeugten Wissens zu kombinieren, und mit Hilfe der *Reasoning* oder *Learning Maschinerie* wünschbare Aktionen zu bestimmen oder die Wissensbasis entsprechend zu erweitern.
Spezifische Agenten legen unterschiedliches Gewicht auf einzelne Funktionen, andere lassen einzelne Funktionen oder Komponenten gar weg. Grundsätzlich ist ein Intelligenter Agent nicht eine vollständige Applikation, sondern bloss eine Komponente einer grösseren Applikation, eine Middleware Resource, und kann von anderen Applikationen genutzt werden.

Mobile Agenten (MA)

Überblick

Da Mobilität das Potential hat, den traditionellen Weg der Kommunikation zur Realisierung von Services zu beeinflussen, ist sie die Eigenschaft mit der grössten Herausforderung an Agenten. Die Mobilität und Verteilung von Programmen ist nicht grundsätzlich neu. Mit *remote (batch) Job Processing* oder später mit *Function Shipping* oder *Remote Evaluation* wurde bereits früher gearbeitet. Motivationen zu diesen Verfahren waren der Mangel an lokaler Rechenkapazität und der Wunsch, Rechenlast über ein verteiltes System ausgeglichen zu verteilen. Im Gegensatz zu diesen Konzepten für abgeschlossene Umgebungen ist das neue MA Konzept für offene Umgebungen (Internet) konzipiert.
Bei Remote Execution der Agenten wird der ausführbare Code (Code und Daten) vor der Ausführung zum Zielsystem transferiert, wo er auch bleibt, bis die Aufgabe abgeschlossen ist. Die Lebenszeit des Agenten ist dabei abhängig von seiner Aufgabe. Remote Execution wird meist dazu verwendet Software auf Verlangen (on-demand) oder dynamisch zu verteilen.
Die *Migration* von Software Prozessen zum Ausgleich der Last in verteilten Betriebsystemen ist eine Erweiterung des *Remote Execution* Konzeptes: Hier hat ein Agent neben seinem Code und seinen Daten auch einen expliziten Ausführungszustand (Execution state), der es ihm erlaubt die Ausführung an einem bestimmten Punkt zu suspendieren (suspend) und sie nach der Migration wieder aufzunehmen (resume). Er kann dadurch auf verschiedenen Knoten unterschiedliche Aufgaben wahrnehmen. Diese Art von MAs sind auch in der Lage, selber neue Agenten zu erzeugen (fork), ihnen Teile ihrer Aufgaben zu übertragen und die Resultate wieder zu synchronisieren (join). Ein Agent kann somit spezifische Aufgaben kontrolliert auf verschiedene Knoten verteilen. MAs entscheiden autonom oder aufgrund eines mitgegebenen Fahrplans (Schedule), wann und wohin sie gehen, wann und wo sie neue Agenten kreieren und wieder terminieren und welche Resultate sie erzeugen. Typische Applikationen sind verteilte Transaktionen.
Dieses Konzept der Mobilität bedingt jedoch Unterstützung bei der Vergabe von globalen Namen, beim Wissen um den geeigneten Standort und die dort auszuführende Aufgabe, beim Auffinden von Agenten, beim Transport von Agenten usw. Zu diesem Zweck werden *Agenten-Plattformen (Agent System, Execution Environment)* eingesetzt. Ein MA Plattform verwaltet, beherbergt und unterstützt deren Life-Cycle (Create, Transfer, Excecute, Terminate usw.) seiner Agenten. Sie garantiert zudem die *Sicherheit* des Systems vor Agenten (Erweiterung der Sandbox bei Java) und schützt die Agenten vor unberechtigter Manipulation ihrer Daten und Programme. Je intelligenter die Plattform selber ist, um so einfacher wird die Programmierung der Agenten.
Der Grund für das aufkommende Interesse an MAs liegt vor allem in der breiten Akzeptanz von Java. Vor wenigen Jahren waren nur wenige Systeme für MAs vorhanden, und diese basierten auf experimentellen Sprachen wie Tcl, Scheme usw. Das einzige kommerziell erhältliche System war Telescript. Java hat diesen Umstand verändert und es stehen heute über ein Dutzend Java-basierte Agentensysteme zur Verfügung. Die Java Virtual Machine und das Java Class-Loading Modell, gekoppelt mit verschiedenen Java 1.1 Erweiterungen, vor allem die Serialisierung, RMI, Multithreading und Reflections haben die Realisierung von MA Systemen stark vereinfacht.
Die entwickelten MA-Technologie und -Produkte unterscheiden sich in ihren Möglichkeiten, ihrer Architektur und ihrer Implementation. Dadurch wird auch die Zusammenarbeit zwischen dispersen Agenten verhindert. Obwohl viele Ähnlichkeiten festgestellt werden können, existieren keine gemeinsamen Definitionen, keine konzeptuellen Modelle, noch akzeptierte Standards. Aus diesem Grund wurde die Standardisierung für MAs 1996 gestartet (siehe Standardisierung).

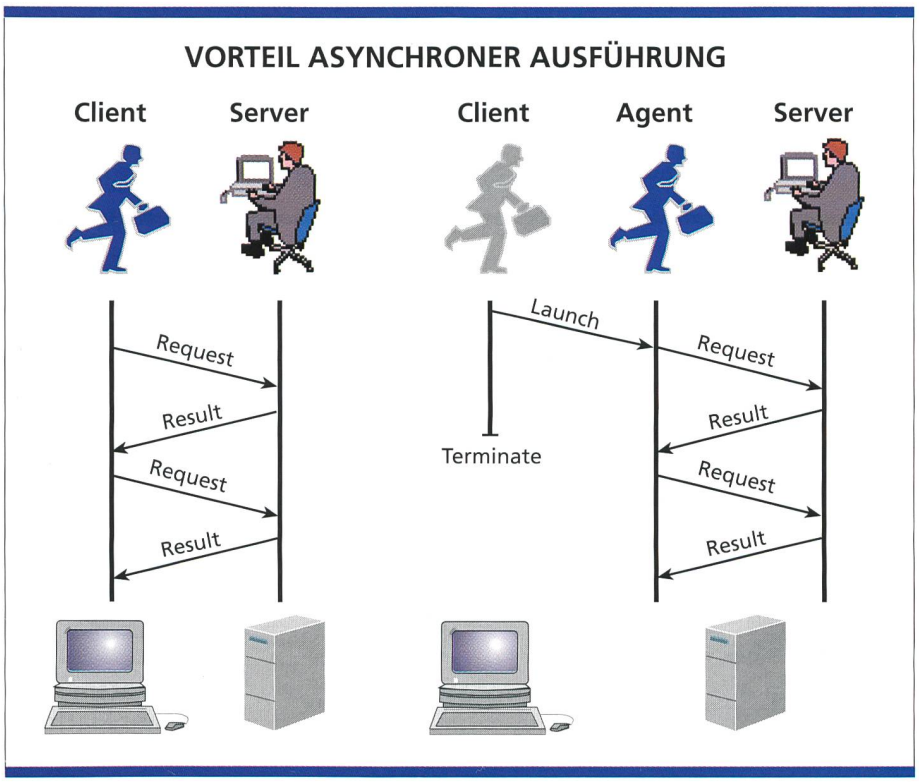

Bild 6. Vorteil asynchroner Ausführung von MAs.

Die treibende Kraft hinter den MAs ist der offene elektronische Service-Markt im Internet. Mit Hilfe von MAs können aber auch die Grenzen der verteilten Objekte (z. B. CORBA) gesprengt werden. MAs treten dabei jedoch nicht als Konkurrenz zu dieser Technologie auf, sondern sind eine sinnvolle Ergänzung.

Vor- und Nachteile Mobiler Agenten

Soll ein neues Paradigma eingeführt werden, müssen seine Vorteile aufgezeigt werden können. Folgende Vorteile können beim Einsatz von MAs identifiziert werden:

- MAs erlauben die *asynchrone Ausführung einzelner Tasks.* Dabei kontrollieren Agenten die Ausführung der Aufgabe und der Client kann entweder terminieren oder andere Aufgaben übernehmen (Bild 6).
- MAs ermöglichen die *Reduktion der Netzlast* und der benötigten *Rechenleistung beim Client.* Dieser Vorteil ist besonders für Netze mit niedriger Bandbreite und leistungschwachen Clients interessant. Die Interaktionen erfolgen dabei lokal auf den meist leistungsstarken Server-Systemen. Typische Anwendungen sind Information Retrieval und Filterung von Informationen an der Quelle. Dabei werden dann nur noch die Resultate an den Client zurückgegeben (Bild 7).
- Mit MAs kann einen *Verbesserung der Robustheit* durch Reduktion der Abhängigkeit von der Verfügbarkeit des Netzes und des Client/Server Systems erreicht werden. Ausfälle des Netzes können den MA nicht tangieren (Bild 8).
- *Verteilte Aufgaben können automatisiert werden,* indem an verschiedenen Standorten spezifische Tasks ausgeführt werden (Bild. 9)
- Durch *Dezentralisierung* können Aufgaben lokal wahrgenommen werden. Sowohl Leistung, Verfügbarkeit als auch Sicherheit werden verbessert (Bild 10).
- Die zusätzliche Flexibilität ermöglicht *on-demand customised SW Distribution* und *Service Provisioning.* Neue Dienste können unmittelbar vom Provider zum Benutzer hinuntergeladen werden. Ebenso können auf diesem Weg auch Server erweitert oder ersetzt werden (Bild 11).
- *Adaption und Kombination bestehender Dienste* wird einfach möglich: MA werden dazu eingesetzt, mehrere Service Provider zu kontaktieren und spezifische Informationen zu filtern. Dazu ist Mobilität jedoch im Prinzip nicht unbedingt notwendig (Bild 12).

Ebenso muss man sich aber auch der Nachteile und Probleme eines neuen Konzeptes wie Mobilität von Code und Daten bewusst sein:

- Die *Sicherheitsfragen* in Zusammenhang mit Mobilen Agenten sind noch nicht vollständig gelöst. Wie soll unterschieden werden, ob es sich um einen *gutmütigen* oder um einen *destruktiven* Agenten handelt? In Java wird diese Gefahr durch die *Sandbox* der Java Virtual Machine entschärft. Trotzdem muss aus Sicht der Agenten-Plattform ein Mechanismus existieren, der fremde Agenten eindeutig identifiziert, das heisst, diese zu einer *Authentisierung* zwingt und sie ggf. zurückweisen kann. Agenten müssen sich selber aber auch gegen Attacken schützen können. Es besteht die Gefahr, dass beispielsweise ihr Auftrag, Fahrplan usw. an einer Plattform manipuliert werden können. Eine wichtige Aufgabe kommt bei diesen Fragen der Agenten-Plattform zu. Ebenso sind geeignete Standards ein wichtiger Beitrag zur Lösung dieser Probleme.
- Der aktuelle *Standort* und der *Status* eines Agenten sind während seines Einsatzes nicht bekannt. Sein Auftraggeber weiss nicht, wo sich der Agent befindet, ob er noch aktiv ist und wie weit sein Auftrag bereits erfüllt ist. Dieses Problem kann durch regelmässige Rückmeldungen und Berichterstattung entschärft werden, der Verkehr hingegen wird dadurch erhöht.
- *Fehlerbehandlung* ist schwierig zu realisieren. Ein Fehlverhalten eines Mobilen Agenten ist auf Distanz schwierig zu erkennen, unmöglich zu beheben.
- Die *Performance* eines interpretierten Systems ist nicht optimal. Da Mobile Agenten meist auf Script-Sprachen oder eben auf Java basieren, wird ihr Code zur Ausführung interpretiert. Interpretierte Sprachen sind meist nicht die effizientesten.
- Die Heterogenität der Umgebungen tangiert die Agenten-Plattformen. Ohne gemeinsame *Standards* kann keine *Interoperabilität* zwischen MAs erreicht werden. Standards der FIPA (Agenten-Management, Kommunikationssprachen, Agenten/Software Integration) und OMG (MASIF (Mobile Agent System Interoperability Facilities Specification) sind sehr wichtig.

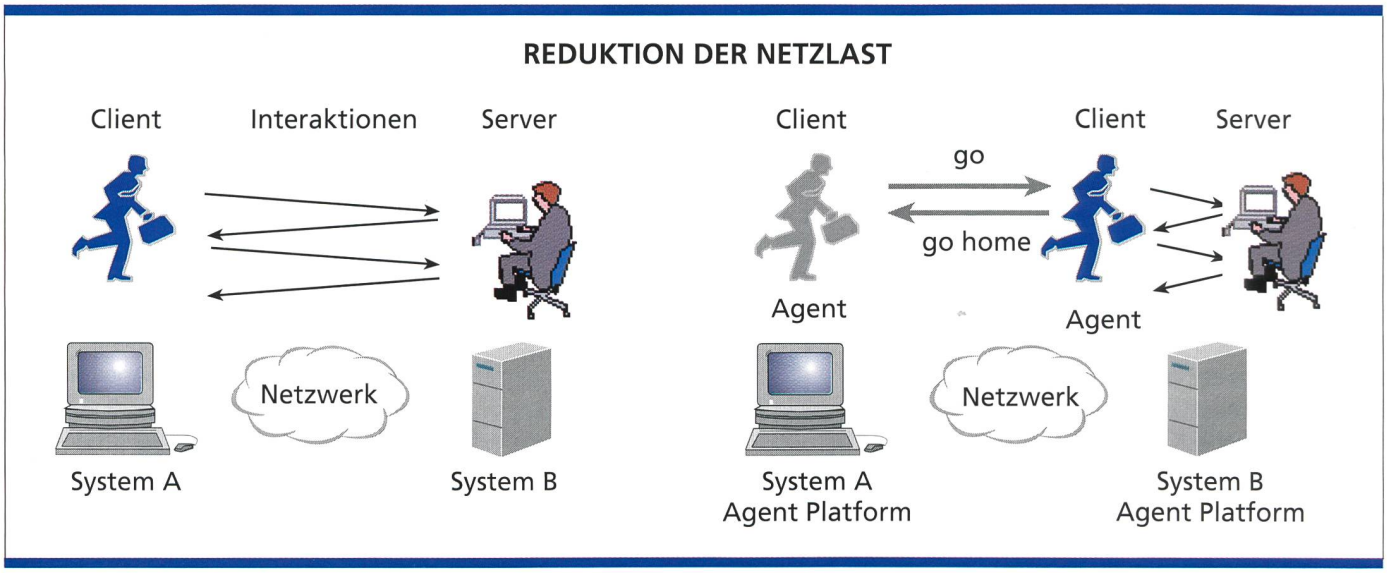

Bild 7. Reduktion der Netzlast und Rechenleistung am Client.

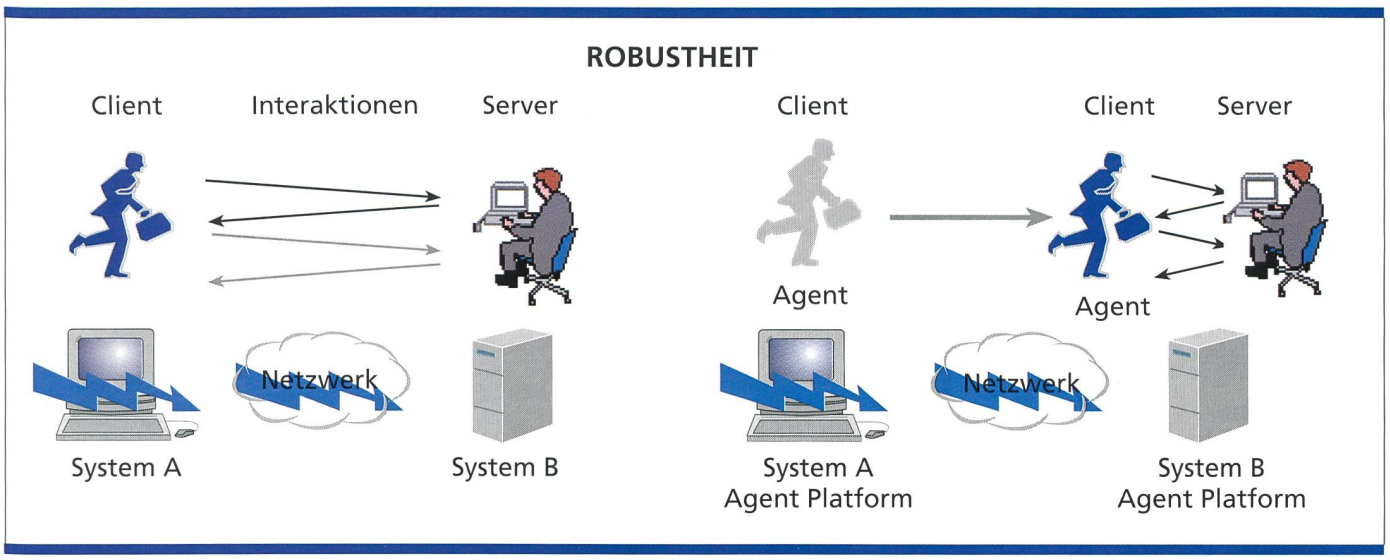

Bild 8. Verbesserung der Robustheit.

- Der *Overhead* beim Transfer von Codes ist im Vergleich zu einfachen Interaktionen gross. Ist die auszuführende Arbeit oder die Verweilzeit auf einem Knoten gering, so lohnt sich der Einsatz von Mobilen Agenten kaum.
- Der Zugriff auf *existierende Services* wie Internet, CORBA usw. muss gewährleistet sein. Da man nicht alle Services von CORBA für Mobile Agenten neu realisieren will, sollten MAs auf existierende CORBA-Services zurückgreifen können.

Einsatz von Mobilen Agenten

Mobile Agenten zeigen vor allem im Telecom-Umfeld ungenutzte Potentiale auf:

- Die Möglichkeit spezielle Aufgaben mit Hilfe von Agenten an einen oder mehrere Knoten zu delegieren, ermöglicht hoch-dynamisches, nebenläufiges, *asynchrones und kooperatives Ausführen von Aufgaben.* Insbesondere kann mit Hilfe von Agenten der Workflow (Prozesse) einfach und äusserst flexibel unterstützt werden.
- *Custimization und Configuration von Services:* In einem elektronischen Markt (Electronic Commerce) können mit Hilfe Mobiler Agenten durch *Customization oder Konfiguration neue Dienste* schnell aus bestehenden Diensten realisiert werden. In diesem Fall werden Agenten zu Dienste-Adaptern und können einfach installiert werden. Agenten können dabei je nach Aufgabe, Rolle und Standort als *Broker, Customer* oder *Provider* auftreten.
- Mobile Agenten übernehmen die Aufgabe von Service Clients, besuchen potentielle Kunden, bieten diesen spontan und unverzüglich die *Nutzung neuer Dienste* an. Diese einfache Verbreitung von Service Clients kann zu aktivem Trading ausgebaut werden.
- Mobile Agenten erlauben, durch *Dezentralisierung* oder Delegation auf disperse Management-Agenten den Druck auf zentrale Management-Systeme und Netzwerk-Bandbreiten zu reduzieren. Die Verteilung kann sowohl *temporal* (Verteilung über die Zeit) als auch *lokal* (auf verschieden Netzwerk-Knoten) erfolgen. Ebenso können mit Hilfe von Mobilen Agenten dem Kunden Dienste speziell *auf Verlangen (on-demand Provisioning)* installiert werden.
- Mit Hilfe von Mobilen Agenten können zukünftige Bedürfnisse nach *intelligenter Kommunikation* unterstützt werden. MAs unterstützen die Konfiguration der Umgebung von Benutzern, wo diese auf eintreffende oder abgehende Kommunikation Kontrolle ausüben wollen (z. B. Kommunikation-Screening, Konversion von Informations-Formaten, Netzzugangsarrangements).
- Mobile Agenten unterstützen mit *Information Retrieval* das effiziente

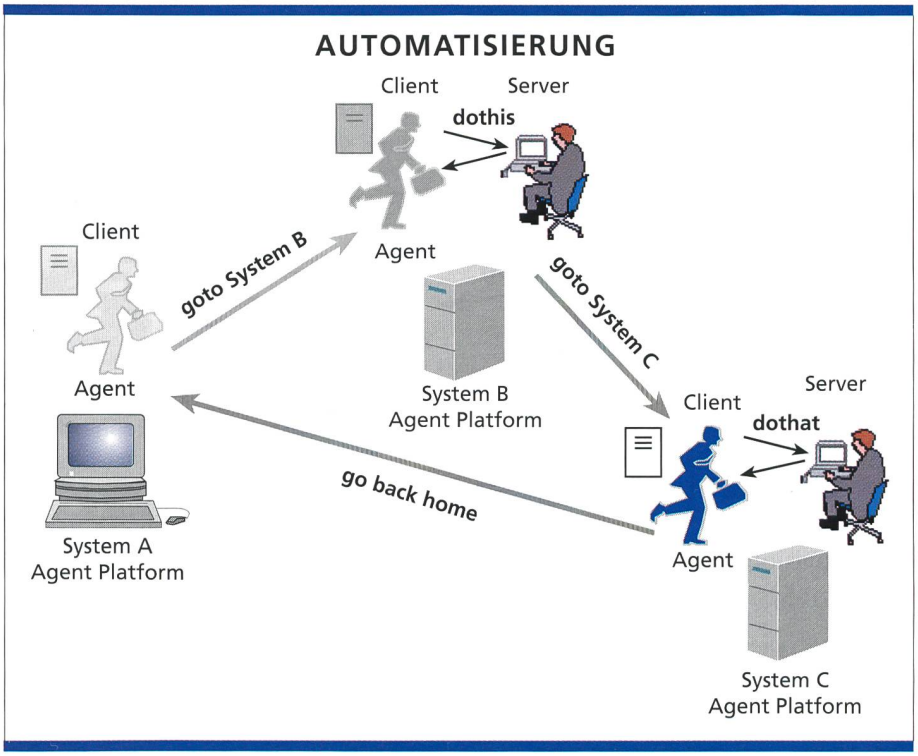

Bild 9. Automatisierung durch Verteilung von Aufgaben.

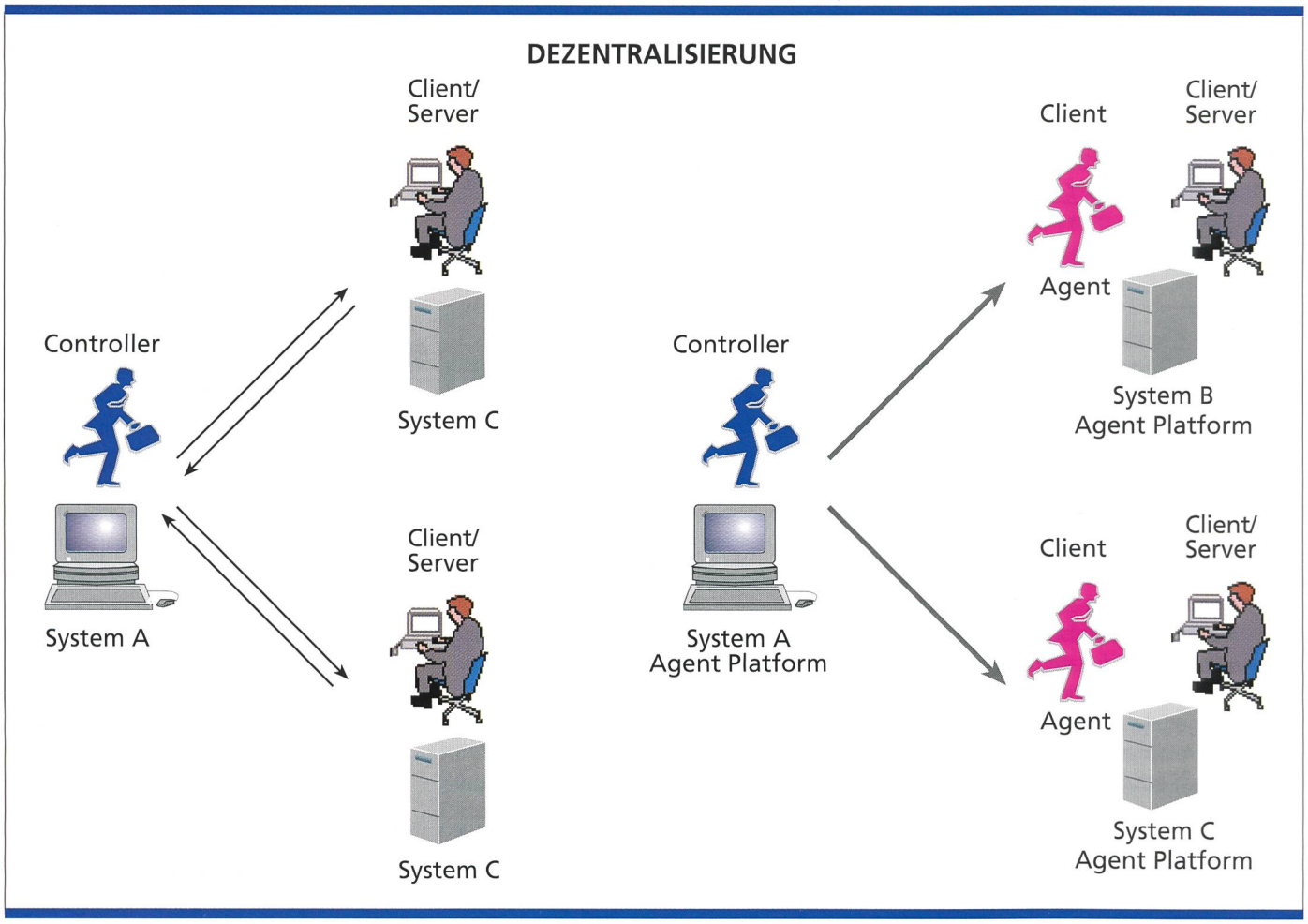

Bild 10. Dezentralisierung.

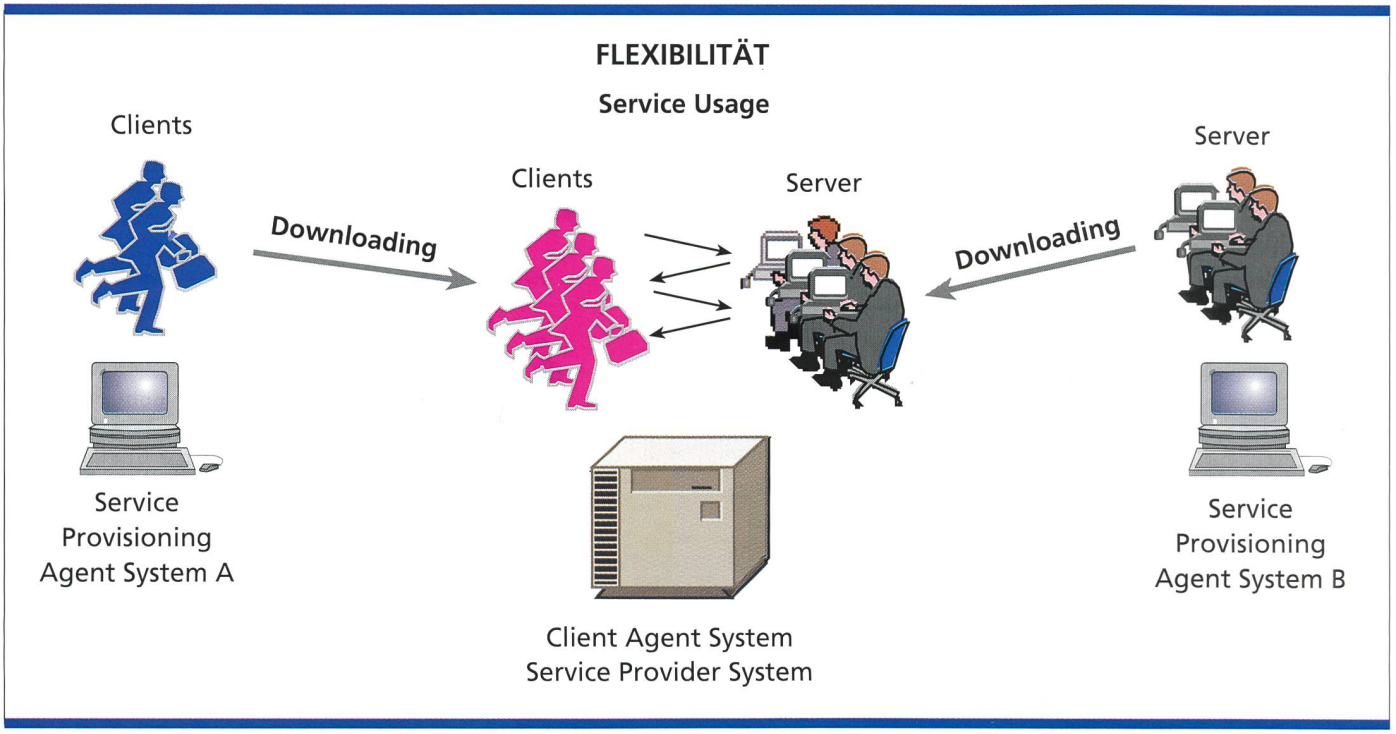

Bild 11. Flexibilität.

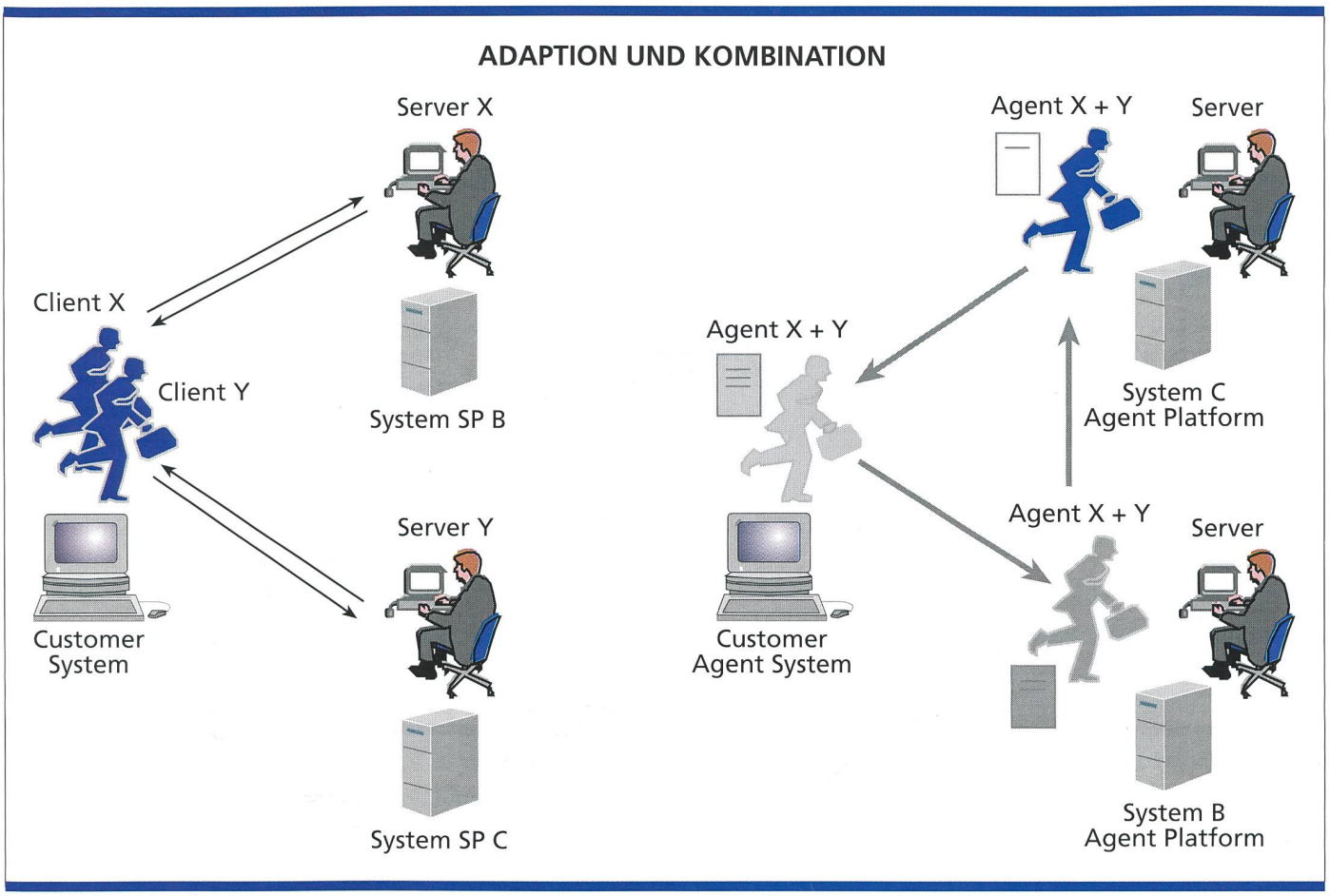

Bild 12. Adaption und Kombination bestehender Dienste.

Sammeln, Selektieren und Filtern von Informationen und Diensten innerhalb eines verteilten Systems, sowie *dynamische Informationstypen* innerhalb von E-Mails oder anderen fortgeschrittenen, netzwerkbasierten Informationssystemen wie beispielsweise im Internet.

Agenten Technologien, Templates und Frameworks

Auch hier muss unterschieden werden zwischen Intelligenten und Mobilen Agenten:

Für Intelligente Agenten sind vor allem die Frameworks JAFMAS und JATLite zu erwähnen, welche die Entwicklung von Multi-Agenten-Systemen unterstützen:

- *JATLite* besteht aus einer Menge von Java-Klassen, die verschiedene Layers von Kommunikations-Protokollen implementieren, und einem Router Agenten, der die Kommunikation kontrolliert. Die gesamte Kommunikation zwischen Agenten erfolgt über diesen Router. Der Router Agent besteht vor allem aus einer *Message Queue*. JATLite ist einfach zu installieren, ist Applet-basiert, benötigt keine Anpassungen des Router-Agenten, und Klienten-Agenten können einfach von einer Basisklasse abgeleitet oder überschrieben werden. Durch die Verwendung von Message Queues gehen keine Meldungen verloren und die Kommunikation des gesamten Systems kann einfach kontrolliert werden. Hingegen lässt sich der Ansatz mit Router-Agent schlecht für grosse Systeme skalieren. Zudem wird der Router zu einem Single Point of Failure und unterstützt keine Peer-to-Peer Kommunikation, wodurch das ganze System blockiert werden kann. Bild 13 zeigt die verschiedenen Ebenen, von denen ein Agent abgeleitet werden kann. Das Template ist frei erhältlich.
- *JAFMAS* ist ein Java-basiertes Agenten-Framework zur Entwicklung und Implementierung von Multi-Agenten-Systemen und ist mit Dokumentation frei erhältlich. JAFMAS definiert eine generische Methode, wie Agenten entwickelt werden sollten. JAFMAS unterstützt die wichtigsten Kommunikations-, Interaktions- und Koordinationsmechanismen, indem die Services in verschiedene Layers unterteilt werden (Bild 14). Die untersten zwei Layer enthalten die Hardware und das Betriebssystem. Das lokale Modell

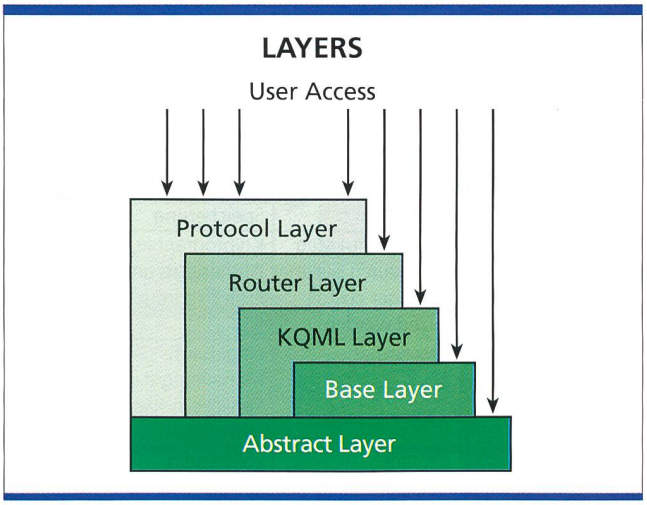

Bild 13. JATLite Layers für Agenten.

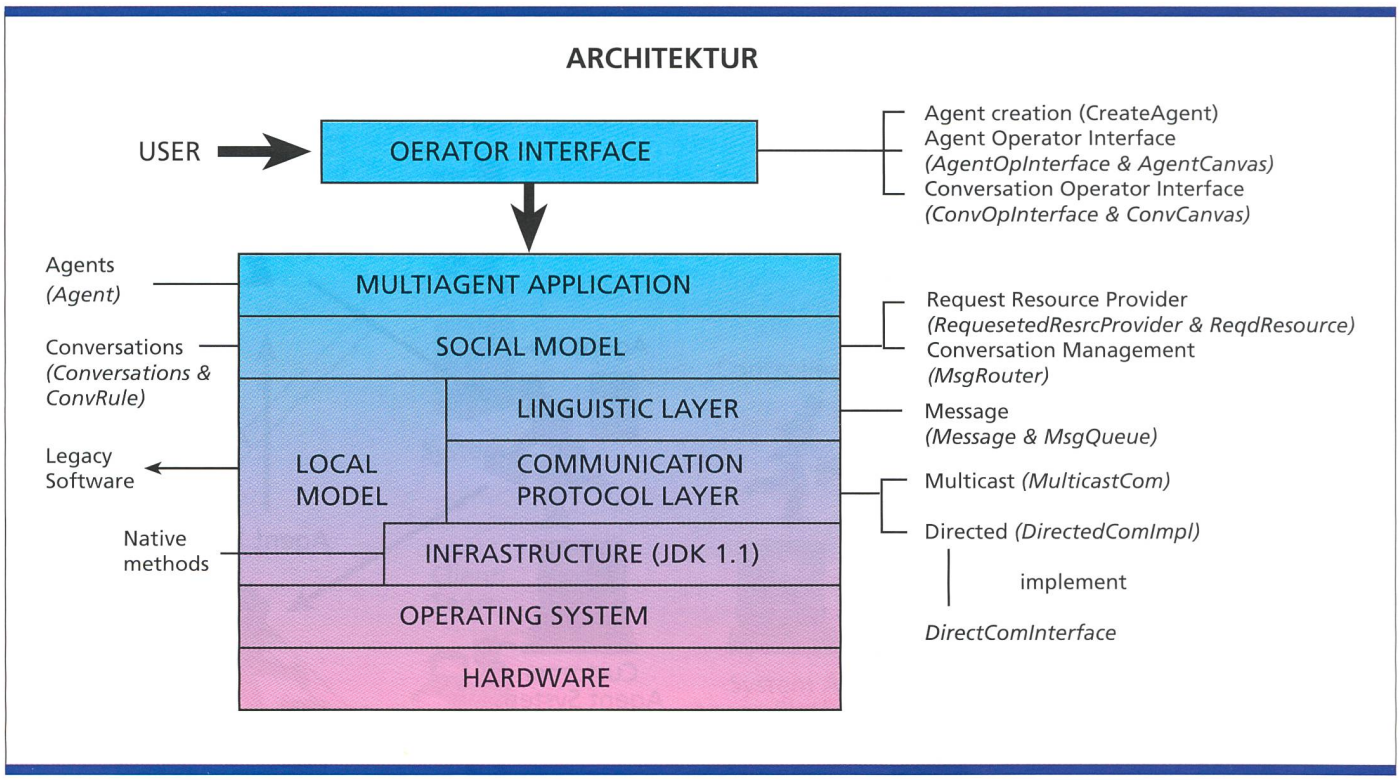

Bild 14. JAFMAS Architektur.

ist anwendungsspezifisch und erlaubt dem Entwickler, Legacy-Software einfach zu integrieren. Der *Linguistic Layer* bietet eine gemeinsame Sprache für Agenten-unabhängige Kommunikation. Der *Kommunikations-Layer* unterstützt sowohl gerichtete als auch Broadcast-Kommunikation mit TCP/IP oder UDP/IP Protokollen. Aufbauend auf der Kommunikation gibt das *soziale Modell* der Agenten vor, wie diese kommunizieren, interagieren und sich untereinander koordinieren, um ein Ziel zu erreichen. Eine *Konversion* ist eine Umsetzung in eine regel-basierte Beschreibung und hält fest, was ein Agent in einer bestimmten Situation tun soll. Über all diesen Layers steht die Multi-Agent-Applikation, welche die JAFMAS Agenten koordiniert und verwaltet.

Im allgemeinen gibt es in JAFMAS wenig Unterstützung des Life-Cyles der Agenten. Ebenso gibt es keine Übereinstimmung mit den Standards von FIPA.

	Aglets	Grasshopper	Odyssey	Voyager
Firmen	IBM	IVK++	General Magic	ObjectSpace
Erhältlichkeit	frei	kaufen	frei	frei
Dokumentation	schwach	gut	schwach	sehr gut
Transport und Kommunikation	ATP (Aglet Transport Protokoll)	CORBA, TCP/IP, RMI	RMI, CORBA, DCOM	Voyager's ORB
Type der Kommunikation	Message Exchange/ Method Invocation	Method Invocation	Nur lokale Method Invocation (Meetings)	Method Invocation
Art der Kommunikation	asynchron, synchron, one-way	asynchron, synchron, dynamisch, one-way (geplant)	synchron (nur lokal)	asynchron, synchron, dynamisch, one-way
Persistence	begrenzt, an Applikation gebunden	unbegrenzt durch Datenbank-Speicherung	keine	unbegrenzt durch Datenbank-Speicherung
Sicherheit	trusted/untrusted kann durch Security Manager customized werden	Kryptographie, Authentisierung, Zugriffskontrolle	Ticket/Places (nur teilweise implementiert)	?
Management von MAs	unterstützt	eingebaute Management-Services in Klassen-bibliothek	keine	keine
MASIF Conformance	geplant	vollständig	keine	keine

Tabelle 2. Vergleich verschiedener Mobile-Agent-Plattformen.

Java-basierte Frameworks zur Entwicklung *Mobiler Agenten* gibt es in der Zwischenzeit eine ganze Reihe. Zu erwähnen sind insbesondere Aglets von IBM, Voyager von ObjectSpace, Odyssey von General Magic und Grasshoppers von IKV++. Diese Plattformen sollen hier nicht im einzelnen vorgestellt, sondern nur kurz in Tabelle 2 einander gegenübergestellt werden. Informationen sind direkt über die URL der einzelnen Plattformen erhältlich.

Standardisierung

Auch die Agenten-Technologie war bereits Ziel verschiedener Standardisierungsbestrebungen. Profiliert haben sich dabei zwei Institutionen, die *FIPA* (Foundation for Intelligent Physical Agents) und die *OMG* (Object Management Group).

Die FIPA, eine non-profit Organisation mit Sitz in Genf, hat das Ziel den Erfolg von Agenten-basierten Applikationen zu fördern. Mit international akzeptierten Spezifikationen soll die Interoperabilität

zwischen agenten-basierten Applikationen erreicht werden.
Die FIPA 97 Spezifikation ist der erste Output der FIPA. Dieser Output spezifiziert die grundlegende Agententechnologie, die zur Entwicklung von komplexen Agentensystemen mit einem hohen Grad an Interoperabilität integriert werden kann. FIPA beschreibt dabei die Interfaces der verschiedenen Komponenten der Umgebung, mit denen ein Agent interagieren kann, das heisst Menschen, andere Agenten, nicht-agenten-basierte Software und die physikalische Welt.
Bild 15 gibt einen Überblick zu diesem Spezifikationspaket.
Dabei wurde zwei Arten von Spezifikationen erzeugt:
- *normative* Spezifikationen, die das externe Verhalten eines Agenten vorgeben und die Interoperabilität mit anderen FIPA-Spezifikationen sichern
- *informative* Spezifikationen, die für Applikationen als *Guide* zur Nutzung der FIPA-Technologie durch die Industrie dienen.

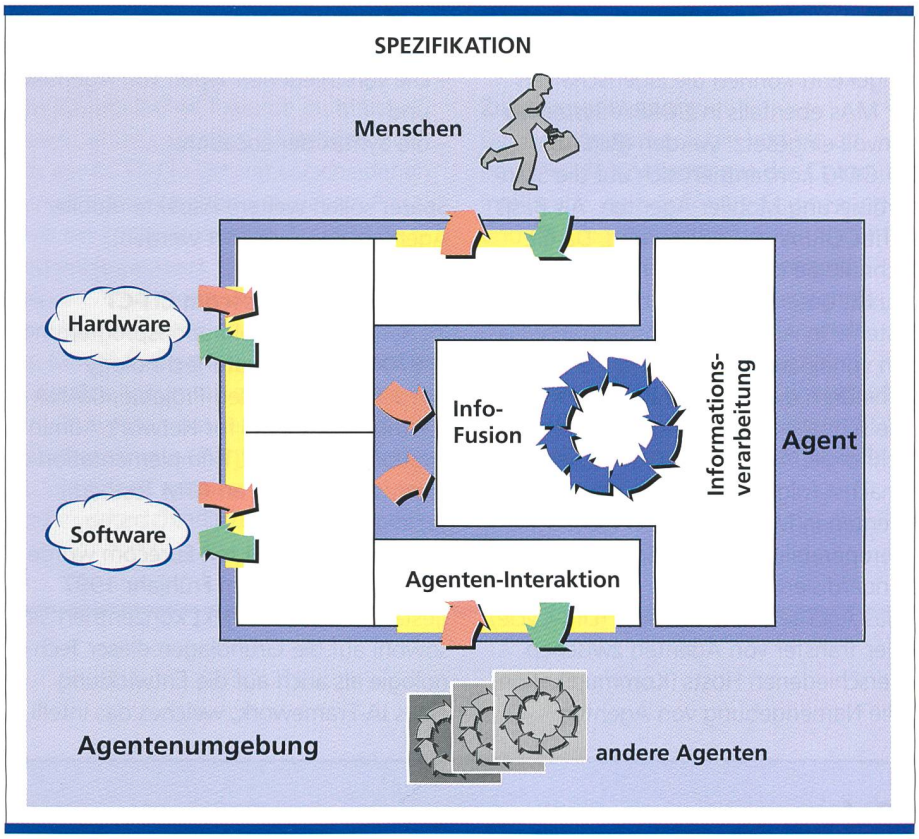

Bild 15. Spezifikations-Framework aus FIPA 97.

Das erste Paket von Spezifikationen – FIPA 97 – hat sieben Teile und umfasst:
- drei normative Teile, die die Grundlagen der Agententechnologie betreffen: Agenten-Management, Agenten-Kommunikationssprache, Agenten-Software-Integration.
- vier informative Teile, die Anwendungsbeschreibungen und Beispiele enthalten, wie normative Spezifikationen eingesetzt werden können: *Personal Travel Assistance, Personal Assistant, Audio-visuale Unternehmen* und *Broadcasting/Network Management/Provisioning.*

Die Spezifikationen sollen folgende Aufgaben unterstützen:
- Die Konstruktion und das Management von Agentensystemen, bestehend aus verschiedenen Agenten und aufgebaut von verschiedenen Entwicklern.
- Die Kommunikation und Interaktion verschiedener Agenten, wobei diese Agenten gemeinsame oder auch individuelle Ziele verfolgen können.
- Legacy-Software oder non-agent Software soll durch Agenten verwendet oder angesprochen werden können.

Verteilte Systeme basieren heute vielfach auf Standards wie CORBA oder DCOM und werden als Schlüsseltechnologien betrachtet. Diese Technologien unterstützen die Realisierung komponenten-basierter Dienste sowie deren Wiederverwendbarkeit, die beliebige Verteilung der Komponenten, eine transparente Kommunikation unabhängig vom Standort (location transparency) und die Interoperabilität zwischen verschiedenen Systemen.
Die Technologie der Mobilen Agenten zusammen mit Java RMI hingegen gewinnt laufend an Bedeutung, da damit Software dynamisch verteilt, die Netzwerklast reduziert und die Robustheit eines Systems – durch Reduktion der Abhängigkeit von der Verfügbarkeit des Netzes, der Server und der Klienten – verbessert werden kann.
Es scheint sinnvoll, diese beiden *orthogonalen* Technologien zu integrieren.
Es wird angestrebt, MA Plattformen auf

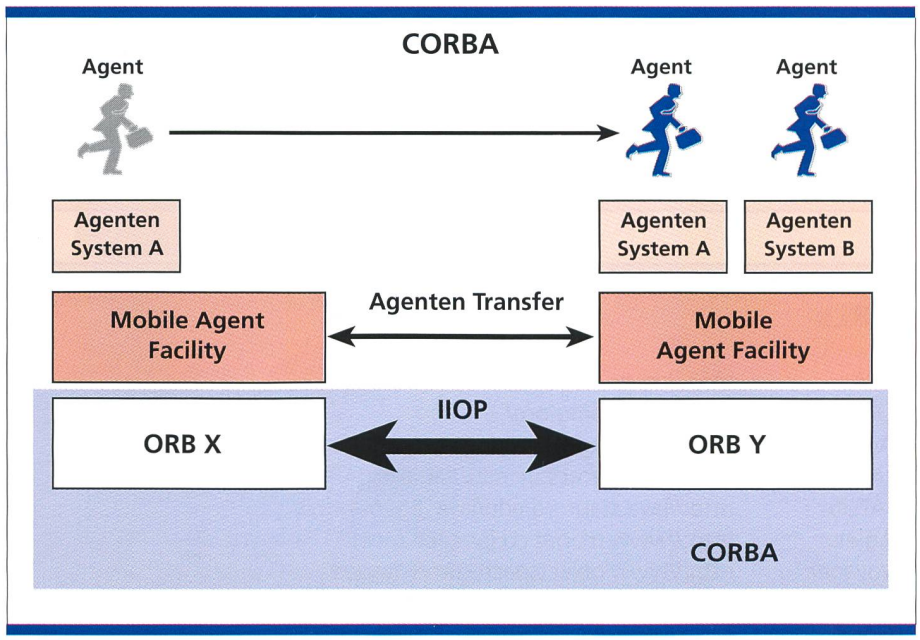

Bild 16. CORBA als Basis Mobiler Agenten.

CORBA Services wie Naming, Trading, Lifecycle, Security und RPC abzustützen. Umgekehrt können die Eigenschaften der MAs ebenfalls in CORBA-Systemen sinnvoll eingesetzt werden (Bild 16).
Die *OMG* konzentriert sich auf die Standardisierung Mobiler Agenten. Als Basis steht CORBA im Vordergrund. Da die Technologie der Mobilen Agenten relativ neu ist, unterscheiden sich die meisten Systeme in Architektur und Implementation voneinander. Diese Unterschiede verhindern die Interoperabilität sowie eine schnelle Verbreitung der Agenten-Technolgie. Aus diesem Grund werden zunächst folgende Aspekte der Mobilen Agenten in MASIF (Mobile Agent System Interoperability Facilities Specification) standardisiert:
- das Agenten-Management (Life-Cycle)
- der Transfer von Agenten zwischen verschiedenen Hosts (Kommunikation)
- die Namengebung von Agenten und Agentensystemen (Syntax und Semantik)
- Die verschiedenen Typen von Agenten, und
- die Syntax der Location.

Später sollen weitere Aspekte mobiler Agenten standardisiert werden.

Agenten bei Swisscom CIT-CT
Im Rahmen des Explorationsprogrammes EP9706 bei Corporate Technology werden zwei Agenten-Projekte, DIANA (Distributed Agents for Network Administration) und IMPACT (Implementation of Agents for CAC in an ATM Testbed), verfolgt:
In Zusammenarbeit mit Eurecom wurde das Projekt DIANA im Frühjahr 1997 gestartet. Dieses Projekt konzentriert sich sowohl auf die Grundlagen dieser Technologie als auch auf die Entwicklung eines IA-Framework, welches das intelli-

Hans Peter Gisiger, *Dr. sc. techn., Dipl. El.-Ing. und Informatik-Ing. ETH, diplomierte 1982 zum El.-Ing. und 1986 zum Informatik-Ingenieur an der ETH Zürich. Anschliessend arbeitete er mehrere Jahre am Institut für Technische Informatik und Kommunikationsnetze (TIK) an der ETH, wo er 1992 auch zum Dr. sc. techn. promovierte. Seit 1995 arbeitet Hans Peter Gisiger bei Swisscom Corporate Technology (früher Forschung und Entwicklung) als Projektleiter, wo er sich als Experte intensiv mit Objekt-orientierten Technologien, New LAN Technologies und Agententechnologie befasst.*

Referenzen

Wooldridge, M., Jennings, N.R.: *Intelligent Agents: Theory and Practice,* Knowledge Engineering Review, January 1995.
Petrie, Ch.J.: *Agent-based Engineering, the Web, and Intelligence,* IEEE Expert, 1996.
Magedanz, T., Rothermel, K., Krause, S.: *Intelligent Agents: An Emerging Technology for Next Generation Telecommunications?,* INFOCOM'96, 1996.
Magedanz, T., Eckardt, T.: *Mobile Software Agents: A new Paradigma for Telecommunication Management,* IEEE/IFIP NOMS96, Kyoto, Japan, 1996.
Mendes, M., Loyolla, W., Magedanz, T., Assis Silva, F.M., Krause, S.: *Agents skills and their roles in mobile computing and personal communications,* IFIP World Conf. On Mobile Communications, Australia, 1996.
Franklin, S., Graesser, A.: Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents, 3rd Inernat. Workshop on Agent Theories, Archs, and Languages, 1996.
Caglayan, A., Harrison, C.: Agent Sourcebook: A Complete Guide to Desktop, Internet, and Intranet Agents, John Wiley & Sons, 1997.
Kiniry, J., Zimmermann, D.: A Hands-on Look at Java Mobile Agents, IEEE Internet Computing, Juli 1997.

URLs

FIPA:	http://drogo.cselt.stet.it/fipa/
OMG:	http://www.omg.org/
Agent Society:	http://www.agent.org/
JAFMAS:	http://www.ececs.uc.edu/~abaker/JAFMAS
JATLite:	http://java.stanford.edu/java_agent/html/
Aglets:	http://www.trl.ibm.co.jp/aglets/
Voyager:	http://www.objectspace.com/voyager/
Odyssey:	http://www.genmagic.com/technology/odyssey.html
Grasshopper:	http://www.ikv.de/products/grasshopper/

Summary

Agent technology

During the 90s we have experienced an inflation of agents, agent concepts and technologies. The emergence of agents has given rise to many discussions. Nowadays almost every program is referred to as an "agent," and a real "agent mania" has broken out. What is an agent really? How does an agent differ from general programs? Are agents objects with intelligent behaviour? This article deals with the term "Agent" and based on its typical characteristics tries to clarify and define it. The opportunities of the agent technology will be shown, the advantages and disadvantages are highlighted, and various manifestations of agents are explained. An attempt is made to provide a taxonomy, a classification into a diagram of agents. The currently existing and available agent frameworks and templates will be presented and their usefulness explained. Within the framework of the exploration program EP9706 two projects dealing with the topic of agents were started at Corporate Technology, Swisscom. The objectives and expectations of these two projects are briefly introduced.

gente Management im LAN-Bereich effizient unterstützen soll. Um die Komplexität der Aufgabe zu meistern, muss die Intelligenz auf verschiedene Stufen des Netzwerk-Managements verteilt werden können. Dazu wird versucht, sehr flexible, adaptive Agenten zu entwickeln. Diese Agenten sollen auf der *BDI-Theorie* beruhen, das heisst, die Hauptkomponenten basieren auf sog. Beliefs, Desires und Intentions. Die Intelligenz eines Agenten wird in verschiedene Layers unterteilt, die untereinander Informationen austauschen. Besondere Beachtung wird dabei auch der Spezifikation derartiger Systeme geschenkt. Um diesen Ansatz in der Praxis zu verifizieren, werden mehrere Case Studies durchgeführt. Erste Resultate sollen Ende dieses Jahres vorliegen.

Das Projekt IMPACT wurde im März dieses Jahres gestartet. Es handelt sich um ein ACTS-Projekt mit verschiedenen europäischen Partnern. In diesem Projekt wird ein Demonstrator in ein ATM Testbed implementiert. Der Demonstrator soll eine CAC-Kontrollstrategie mit Hilfe von Agenten realisieren. Das Projekt soll die Möglichkeiten Intelligenter Agenten in der Telekommunikation aufzeigen. Im Projekt werden das reaktive und das selbstbestimmte (Planungs-)Verhalten konzeptionell voneinander getrennt und und in unterschiedlichen Komponenten implementiert. Ein erster Prototyp soll Anfang 1999 bereitstehen.

Zusammenfassung

Die Agententechnologie kombiniert bestehende Konzepte aus der Computer-Intelligenz, dem Software-Engineering und den Benutzerschnittstellen. Agenten weisen dadurch Eigenschaften und Fähigkeiten auf, die neue, ungeahnte Potentiale schaffen. Diese Potentiale sinnvoll und gewinnbringend auszuschöpfen, wird die Aufgabe der aktuellen und zukünftigen Forschung und Entwicklung im Umfeld der Agententechnologie sein. 11