Reihe: Telekommunikation und Mediendienste · Band 2

Herausgegeben von Norbert Szyperski, Udo Winand, Dietrich Seibt und Rainer Kuhlen

Martin Engelien/Kai Bender (Hrsg.)

GeNeMe98

Gemeinschaften in Neuen Medien

TU Dresden, 1./2.10.1998

JOSEF EUL VERLAG

Lohmar · Köln

Reihe: Telekommunikation und Mediendienste · Band 2

Herausgegeben von Prof. Dr. Dr. h. c. Norbert Szyperski, Köln, Prof. Dr. Udo Winand, Kassel, Prof. Dr. Dietrich Seibt, Köln, und Prof. Dr. Rainer Kuhlen, Konstanz

Doz. Dr.-Ing. habil. Martin Engelien
Dipl.-Inf. (FH) Kai Bender (Hrsg.)

GeNeMe98

Gemeinschaften in Neuen Medien

TU Dresden, 1./2.10.1998

JOSEF EUL VERLAG
Lohmar · Köln

Die Deutsche Bibliothek – CIP-Einheitsaufnahme

GeNeMe <1998, Dresden>:
GeNeMe 98 : Gemeinschaften in neuen Medien / Technische Universität Dresden, Fakultät Informatik, Institut für Informationssysteme, Dozentur „Entwurfmethoden und Werkzeuge für Anwendugssysteme". Martin Engelien; Kai Bender (Hrsg.). – Lohmar ; Köln : Eul, 1998.
(Reihe: Telekommunikation und Mediendienste ; Bd. 2)
ISBN 3-89012-632-4

Josef Eul Verlag GmbH
Brandsberg 6
53797 Lohmar
Tel.: 0 22 05 / 91 08 91
Fax: 0 22 05 / 91 08 92
e-mail: eul.verlag.gmbh@t-online.de

Printed in Germany
Druck: Rosch-Buch, Scheßlitz

Gedruckt auf säurefreiem und 100% chlorfrei gebleichtem Papier

Technische Universität Dresden
Fakultät Informatik • Institut für Informationssysteme
Dozentur „Entwurfsmethoden und Werkzeuge für Anwendungssysteme"

Doz. Dr.-Ing. habil. Martin Engelien
Dipl.-Inf. (FH) Kai Bender
(Hrsg.)

Dresden, 1./2.10.1998

GeNeMe98

Gemeinschaften in Neuen Medien

Workshop zu Organisation, Kooperation und Kommunikation auf der Basis innovativer Technologien

Forum für den Dialog zwischen Wissenschaft und Praxis zur Inversion der Virtualität (Ubiquitous Computing)

unter der Schirmherrschaft von:

Dr. W. Vehse
Staatssekretär für Wirtschaft
des Landes Sachsen

Prof. Dr. A. Mehlhorn
Rektor der TU Dresden

sowie unter Mitwirkung der
GI-Regionalgruppe Dresden

und mit freundlicher Unterstützung folgender Partner:

IST priv. Institut für angewandte Software-Technologie GmbH, Dresden
eine Ausgründung der TU Dresden auf dem Gebiet der Technologien und Anwendungen in den Neuen Medien

Heyde AG,
Bad Nauheim/ Dresden
Beratung • Software • Integration

B.3. Virtuelle Gemeinschaften - Infrastruktur und Technologie

Dr.-Ing. V. Do
Dipl.-Inf. D. Nguyen
Technische Universität Dresden

Abstract

Als "Virtual Community" wird eine Organisationsform bezeichnet, die auf der Basis eines gemeinsamen Interesses und mittels neuer elektronischer Medien gegründet wird. Dabei soll die Integration von Kommunikationsinhalt und Kommunikationsmittel unterstützt werden. Virtuelle Gemeinschaften erlauben es Unternehmen und anderen Organisationsformen wie z.B. Vereinen, ihre Ressourcen zu bündeln und so gemeinsam eine stärkere Wettbewerbsposition aufzubauen. Durch Informationstransparenz, Kombination der verschiedenen Kernkompetenzen und Verteilung des Risikos werden sie in die Lage versetzt, mit überregional agierenden Wettbewerbern Schritt zu halten bzw. diesen ihre Leistungen anzubieten.

Obwohl derzeit bereits eine Reihe von technischen Voraussetzungen gegeben sind, ist die eigentliche Infrastruktur dazu erst im Entstehen. Dies betrifft einerseits informationstechnische Aspekte (z.B. Interoperabilität unterschiedlicher Verfahren, Datenverschlüsselungssysteme, Qualitätsgarantien), andererseits aber auch organisatorische und rechtliche Rahmenbedingungen. Bei einer Infrastruktur für solche Anwendungen müssen nicht nur Mittel für die Implementierung der Anwendungslogik, sondern auch die Integration der Extrakontext-Logik bereitgestellt werden.

In diesem Artikel sollen die Grundlagen der Infrastruktur für virtuelle Gemeinschaften, besonders die technische und technologische Apsekte behandelt werden. Bei den technischen Grundlagen werden Themen zur Architektur dezentraler Systeme, deren Skalierbarkeit, Robustheit und Effizienz sowie Basismechanismen für Interoperabilität behandelt. Das Paradigma der mobilen Agenten für Client-Server-Interaktion wird vorgestellt und dessen Einsatz für die Infrastruktur der virtuellen Gemeinschaften wird anschließend diskutiert.

1 Einführung

Das Konzept der virtuellen Gemeinschaften (Virtual Communities Abk. VC) wurde entwickelt und eingesetzt, um Chancen in einem rasch sich ändernden, agilen Wettbewerbsumfeld zu nutzen. Eine solche Organisationsform ist nur erfolgreich, wenn sie über eine gut funktionierende technische Infrastruktur verfügt. Ubiquitous

Computing - allgegenwärtige Kommunikation und Information – ist der zentrale, fundamentale und kritische Teil der VC (vgl. [1]). Es ist der Ansatz, der die Durchdringung des Lebens- und Arbeitsumfeldes aller Menschen durch diese IT-Technologien fordert. Die primäre Anforderung an die Infrastruktur für virtuelle Gemeinschaften (Virtual Community Engine Abk. VCE) ist die Bereitstellung der Kommunikation und Information zur Überwindung der Grenze von Raum und Zeit.
Grundlegend für eine solche Infrastruktur ist die Vernetzung der Teilnehmer der virtuellen Gemeinschaft mittels eines Netzwerks. Dieses Netzwerk muß für die Massen preiswert zugänglich sein. Internet ist für diesen Zweck besonders geeignet. Man erhält über Modem bzw. ISDN Zugang zum Internet. Zur Zeit wird ein Verfahren zur Übertragung von Internet-Daten über das Elektrizitätsnetz von zehn internationalen Versorgungsunternehmen, darunter auch die deutsche RWE, erprobt. Die sogenannte Digital PowerLine-Technik könnte vor allem strukturschwächeren Regionen mit unterentwickelter Kommunikations-Infrastruktur, aber funktionierender Elektrizitätsversorgung Zugang zum Internet verschaffen (vgl. [7]).
Das Internet ist praktisch ein Netzwerk, über das an unterschiedlichen Orten untergebrachte Rechner Daten austauschen können. Daher können sich die Teilnehmer einer VC an das Internet anschließen und die Dienste der VCE nutzen, um die von ihnen gewünschten Aktivität auszuführen.
Das folgende Bild zeigt die allgemeine Infrastruktur für eine virtuelle Gemeinschaft.

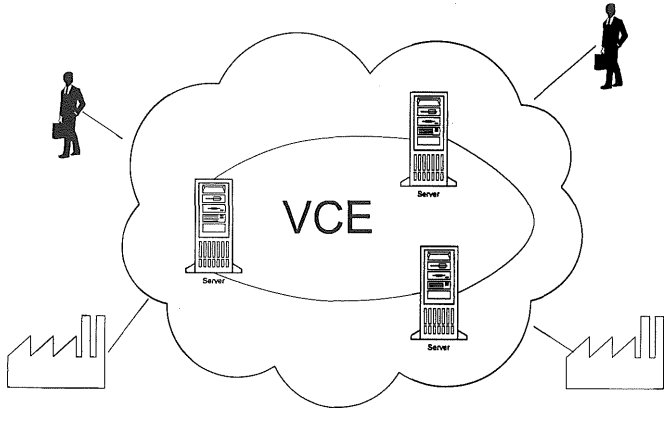

Abbildung 1: Infrastruktur für virtuelle Gemeinschaft als Verbund von Servers und Clients

1.1 Infrastruktur der Virtuellen Gemeinschaft als Crossware: Cross-Everything—Anwendungen

Traditionelle Softwaresysteme wurden für eine spezifische Hardwareplattform und Betriebsystemversion entwickelt, von einem Experten zum Einsatz vorbereitet, der vor einem bestimmten Rechner sitzen mußte, um diese zu installieren, und von Benutzern verwendet, die für diese spezifische Anwendung aufwendig trainiert wurden.
Diese traditionelle Sichtweise auf Software ist für Anwendungen, die die Überwindung der Grenze von Raum und Zeit unterstützen, nicht mehr adäquat. Software, die auf diese Weise entwickelt wurde, wird weder im heterogenen Netzwerk skalierbar noch von Benutzern einsetzbar sein, die an der virtuellen Gemeinschaft teilnehmen.
Um die Zusammenarbeit zwischen den räumlich entfernten Teilnehmern zu unterstützen, muß ein VCE den Charakter der Crossware aufweisen. Crossware beschreibt On-demand-Applikationen, die über die Netzwerk- und Betriebssystemgrenzen hinaus lauffähig sind und ausschließlich auf offenen Internet-Standards wie HTML, Java und JavaScript basieren . Der Begriff On-demand deutet darauf hin, daß die Anwendung erst bei Nutzungsbedarf auf das Rechnersystem des Benutzers übertragen wird. Crossware ist dafür geeignet, Projekte und Prozesse zu unterstützen, die Menschen und Netzwerke umspannen. Der Begriff Crossware wurde von Brian K. Kathman von Systems Resources Consulting geprägt und bildet in der Vision von Netscape einen wichtigen Meilenstein der zukünftigen Software-Entwicklung.
Aufbauend auf den Standard-Internetprotokollen wie HTTP, IIOP, LDAP und SMTP, bietet Crossware folgende Vorteile:

- Nahtlose Interoperabilität: Crossware kann auf die gleiche Weise eingesetzt werden, unabhängig davon, mit welchem Client- oder Server-Betriebssystem man arbeitet, ohne daß Änderungen von Programmcode notwendig wären.
- Crossware-Applikationen können auf die Anwendungslogik und Daten zugreifen, die in Datenbanken, Mainframes und in Geschäftsanwendungen vorliegen.
- Zentrale Verwaltung: Durch eine serverbasierte Anwendungslogik kann Crossware von tausenden oder Millionen von Benutzern eingesetzt und später auf einmal aktualisiert werden.
- Extreme Skalierbarkeit: Crossware wird auf der Basis von modularen Komponenten eingesetzt, die zwischen einer großen Anzahl von Servern partitioniert, repliziert und auf eine praktisch unbegrenzte Anzahl von Benutzern erweitert werden kann.

- On-demand-Zugriff und leichte Erlernbarkeit: Durch interaktive, dynamische, in HTML, JavaScript oder Java geschriebene Benutzerschnittstellen kann Crossware bei Bedarf geladen und sofort eingesetzt werden.

Wenn man die Crossware-Anwendung näher betrachtet, kann man feststellen, daß die Software-Architektur solcher Anwendungen der Dreischichten-Architektur (Three Tier Architecture) entspricht, während traditionelle Software-Systeme meist der Zweischichten-Architektur (Two Tier Architecture) zugeordnet werden können. Im nächsten Abschnitt werden beide Architekturen vorgestellt und diskutiert.

1.2 Client-Server-Architektur

Eine Zweischichten -Architektur ist eine klassische Client-Server-Architektur. Dienstnachfrager (Client) und Diensterbringer(Server) interagieren, dabei sind Client und Server im allg. auf unterschiedlichen Knoten eines Rechnerverbundes plaziert und kommunizieren über ein Rechnernetz miteinander. Das Client/Server-Modell ist ein Software-Architekturmodell, welches die Rollen der Beteiligten und die zeitliche Abfolge der Interaktionsschritte festlegt. In der Zweischichten-Architektur befindet sich die Darstellungslogik auf der Client-Maschine. Der Server ist ein Datenbank-Server. Jeder Client besitzt eine DB-Verbindung (connection) zu dem DB-Server während der Sitzung, schickt Abfragen zum Server und verarbeitet die zurückgelieferten Ergebnisse. Die Anwendungslogik wird entweder zusammen mit der Darstellungslogik beim Client gelagert (Fat-Client) oder als Triggers oder Stored Procedures beim Server programmiert (Fat-Server). Vorteil einer solchen Architektur ist die einfache Implementation der Anwendungen.
Als Nachteile dieser Architektur können folgende Punkte genannte werden.

- Datenbank-Treiber (ODBC, JDBC) müssen unbedingt auf jedem Client installiert werden. Jede Änderung der Anwendungslogik beim Fat-Client führt zu Update der Anwendungen bei jeder Client-Maschine. Das führt zu großem Administrationsaufwand, besonders wenn die Anzahl der Clients hoch ist.
- Die Anwendung ist nicht skalierbar. Da die Client-Anzahl gleichzeitig die Anzahl der DB-Anwendungen bedeutet, gibt es schon erhebliche Beeinträchtigung der Performance bei mehr als 100 Clients.
- Wenn die Anwendungslogik beim Fat-Client einen hohen Rechenaufwand erfordert, ist sie eine schwere Last für die Client-Maschine.

Eine Anwendung mit Dreischichten-Architektur wird – wie der Name sagt - in drei Schichten eingeteilt: Darstellungsschicht, Anwendungsschicht und Datenbank-Schicht. Jede Schicht führt Funktionen aus, für die sie verantwortlich ist. Zur Realisierung der

Dreischichten -Architektur ist die Unterstützung der Kommunikation zwischen den Objekten unbedingt notwendig (vgl. [15], [16], [17]). Die Nachteile der Zweischichten-Architektur werden bei dieser Architektur aufgehoben. Eine Anwendung mit dieser Architektur erfordert jedoch hohen Entwurfs- und Realisierungsaufwand. Das folgende Bild zeigt die Dreischichten-Architektur und die zugehörige Logik der Schichten.

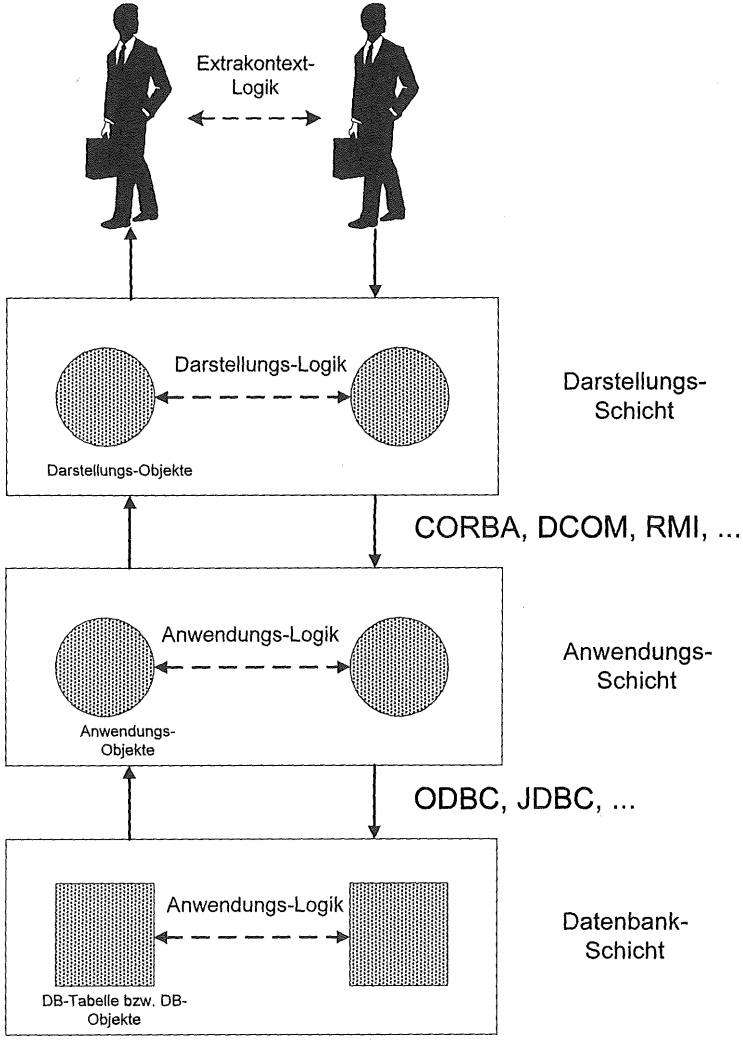

Abbildung 2: Dreischichten-Architektur und zugehörige Logik

Der Benutzer einer solchen Anwendung kommuniziert mit den Darstellungs-Objekten (GUI-Elementen) der Anwendung. Die Kommunikation zwischen diesen Objekten entspricht einer Darstellungslogik. Sie besteht aus Dialogsteuerung und Dialogverarbeitung. Die Dialogsteuerung kontrolliert die Ablauflogik der Nachrichten entsprechend einem Zustandsgraphen. Gegenüber konventionellen Systemen muß eine VCE Dialogführung anbietet, in der die Lösung nicht mehr "menügetrieben" strukturiert wird, wobei das System den Nutzer führen würde, sondern vielmehr "nutzergetrieben", wobei der Nutzer jeweils die *Initiative* ergreift und die Funktionen des Systems in dem von ihm gewünschten Ablauf steuert. Das hängt damit zusammen, daß die Analyse von VCE die Modellierung von Strukturen erfordert, die erst bei der Nutzung des fertigen Softwareproduktes, nicht aber in der Software selbst, zum Tragen kommen. Solche Strukturen werden in [5] als Extrakontext-Logik bezeichnet. Die Anwendungslogik beschreibt die Zusammenhänge zwischen der Anwendung und deren Umgebung, während die Extrakontext-Logik die logischen Zusammenhänge zwischen den Elementen der Umgebung der Anwendung darstellt. Eine Infrastruktur für virtuelle Gemeinschaften muß zur Erhöhung der Flexibilität und darf nicht zur Reduzierung der Flexibilität der Arbeit der Teilnehmer führen. Die Reduzierung der Flexibilität kommt vor, wenn die Extrakontext-Logik mit der Anwendungslogik verwechselt wird und als Anwendungslogik in die Anwendung fest programmiert wird.

Die Anwendungslogik einer Anwendung wird durch die Kommunikationen zwischen Anwendungsobjekten dargestellt. Sie sind individuell erstellte Objekte, die die Anwendungsfunktionalität bereitstellen.

Einige DBMS bieten sogenannte "Trigger" und "Stored Procedures", hinter denen sich Anwendungs-Logik verbirgt, die vom DBMS verwaltet und für den Nutzer unsichtbar abhängig von bestimmten Operationen und Ereignissen aktiviert wird. Diese Technik ist für die Lösung kommerzieller Aufgabenstellungen in einer C/S-Architektur bereits akzeptiert, da sie den Datenverkehr zwischen der DB-Schicht und der Anwendungsschicht reduziert und daneben erlaubt, bestimmte Vorselektionen und Integritätsregeln bereits in der Datenbank zu spezifizieren.

Die Realisierung der Client-Server-Architektur basiert auf einigen Middleware-Technologien, die im folgenden vorgestellt werden.

2 Middleware-Technologien für den Aufbau verteilter Anwendungen

Der Begriff Middleware wird für eine Menge von Diensten verwendet, die Anwendungen und Endnutzern erlauben, Informationen über Netzwerken auszutauschen. Middleware-Technologien werden für die Kommunikation zwischen der

Darstellungs-Schicht und Anwendungs-Schicht bzw. zwischen der Anwendungs-Schicht und Datenbank-Schicht verwendet. Diese Dienste liegen unter der verteilten Anwendung und über dem Betriebssystem bzw. der Netzwerksoftware (vgl. Abbildung 3).

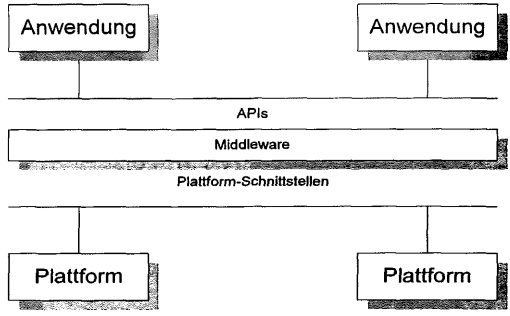

Abbildung 3: Middleware

Middleware-Technologien können in drei Kategorien eingeteilt werden: Kommunikationsdienste, Management- und Unterstützungsdienste (wie z.B. Namen- , Sicherheitsdienste usw.) und anwendungsspezifische Dienste (z.B.Datenbank-Transaktion). Im folgenden werden wir die wichtigsten Kommunikations-Middleware-Technologien betrachten.

2.1 DCOM (*D*istributed *C*omponent *O*bject *M*odel)

DCOM ist die verteilte Erweiterung zu COM (*C*omponent *O*bject *M*odel), das eine ORPC-Schicht (Object remote procedure call layer) auf dem DCE RPC (Distributed Computing Environment Remote Procedure Call) aufbaut, um ferne Objekte zu unterstützen. Ein COM-Server kann Objekt-Instanzen einer mehrfachen Objektklassen kreieren. Ein COM-Objekt kann Mehrfach-Schnittstellen unterstützen, die unterschiedliche Sichten auf und Verhalten des Objekts darstellen. Jede Schnittstelle (Interface) enthält eine Menge von funktionell verwandten Methoden. Ein COM-Client kommuniziert mit einem COM-Objekt durch Erwerbung eines Zeigers auf eine der Objektschnittstellen und durch Aufruf der Methoden mittels des Zeigers, als ob sich das Objekt im Adreßraum des Clients befände.

Zentrales Element dieses Konzepts sind Komponenten. Die Komponenten implementieren die Anwendungslogik und sind an den Anwendungsabläufen beteiligt. Sie können in Visual Basic, C++, Visual J++ oder mit Hilfe eines ActiveX-kompatiblen Entwicklungswerkzeugs erstellt werden.

2.2 CORBA (*C*ommon *O*bject *R*equest *B*roker *A*rchitecture)

CORBA ist eine Abkürzung für Common Object Request Broker Architecture. CORBA wurde von dem Konsortium Object Management Group (OMG) entwickelt, mit dem Ziel, eine offene verteilte Objekt-Architektur zu definieren.
Der Kern der CORBA-Architektur ist der Object Request Broker (ORB), der als Objektbus agiert, über den Objekte mit anderen lokalen und fernen Objekten transparent kommunizieren. Ein CORBA-Objekt wird für die äußere Welt durch eine Schnittstelle (Interface) als eine Menge von Methoden dargestellt. Eine Instanz eines Objekts wird durch eine Objektreferenz identifiziert. Der Client eines CORBA-Objekts bekommt seine Objektreferenz und kann seine Methoden aufrufen, als ob sich das Objekt im Adreßraum des Clients befände. Der ORB ist verantwortlich für alle Mechanismen zum Finden der Objektimplementation, Vorbereiten des Empfangs der Client-Forderung, Transfer der Forderung zum Objekt und Rückbringen der Anwort zu dem Client. CORBA hat mehrere Vorteile wegen der Sprachen-, Hersteller- und Betriebssystemunabhängigkeit gegenüber anderen verteilten Modellen.

2.3 RMI

Remote Method Invocation (RMI) ist eine Leichtgewicht-Java-ORB-Architektur, die einfachen Zugriff auf Objekte auf fernen virtuellen Maschinen ermöglicht. Nachdem man eine Referenz zu einem fernen Objekt bekommen hat, kann man sie wie ein lokales Objekt behandeln. RMI kann nur in einer homogenen Java-Landschaft benutzt werden und ist deshalb kein direkter Konkurrent zu CORBA oder DCOM, die nicht nur eine Programmiersprache unterstützen. RMI bietet auch für ferne Objekte automatische Speicherbereinigung. Objekte können auch by-value übergeben werden. Klassen können auch dynamisch über das Netz geladen werden. RMI kann auch über eine Firewall hinweg genutzt werden („Tunneling"). Als Nachteil von RMI kann das Fehlen der automatischen Aktivierung von Servern, der Kommunikationssicherheit und anderer CORBA-ähnlicher Dienste gewertet werden.

3 Neues Paradigma der Client-Server-Interaktion: Mobile Agenten

3.1 Paradigma der Mobilen Agenten

Die Client-Server-Interaktion kann im allgemein in zwei Paradigmen erfolgen. Das erste ist das Paradigma des synchronen Nachrichtenaustauch, wobei die Objekte verteilt, aber stationär sind. Diese können miteinander durch Nachrichtenaustauch, der auf synchronen Protokollen basiert, kommunizieren. Die Basis-Technologie für dieses Paradigma ist die RPC-Technik, auf der die Konzepte von DCOM, CORBA und RMI entwickelt wurden.

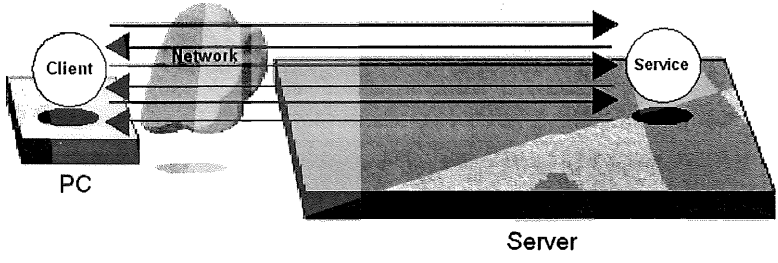

Abbildung 4: : Paradigma des synchronen Nachrichtenaustauschs

Das Paradigma der mobilen Agenten bietet eine andere Kommunikationsmöglichkeiten zwischen verteilten Objekten durch Konzepte wie asynchronen Nachrichtenaustausch, Objektmobilität und aktive Objekte. Zusammen mit der Mobilität wird ein Agent durch folgende Eigenschaften charakterisiert:

- Objektübergabe: Wenn ein mobiler Agent sich bewegt, wird das gesamte Objekt, d.h. sein Code, seine Daten, sein Ausführungszustand und seine Reiseroute, zusammen übergeben.
- Selbständigkeit (Autonomie): Der mobile Agent hat seinen ein eigenen Ausführungspfaden(thread of execution) und kann asynchron ausgeführt werden.
- Lokale Interaktion: Der mobile Agent interagiert lokal mit den anderen mobilen Agenten und stationären Objekten.
- Getrennte Operation: Der mobile Agent kann sein Task bei der offenen oder geschlossenen Netzwerkanbindung ausführen. Wenn die Netzwerkanbindung geschlossen ist und der Agent sich bewegen will, kann er warten bis die Anbindung wieder aufgebaut wird.

- Parallele Ausführung: Mehr als ein mobiler Agent können zu unterschiedlichen Servern geschickt werden, um Aufgaben parallel auszuführen.

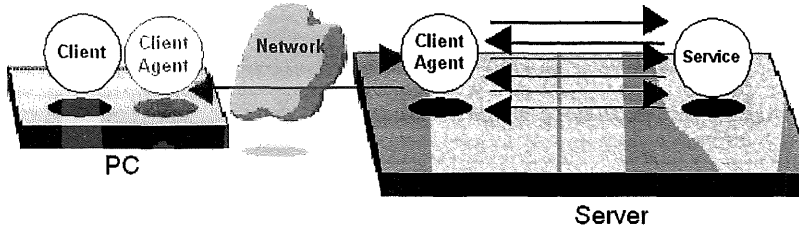

Abbildung 5: Paradigma der mobilen Agenten

3.2 Mobile Agenten für virtuelle Gemeinschaften

Das Paradigma der mobilen Agenten liefert mehrere Vorteile gegenüber dem traditionellen Paradigma, besonders bei der Infrastruktur für virtuelle Gemeinschaften. Agenten bieten bessere Unterstützung für mobile Clients. Die Agenten brauchen nur Low-bandwidth-Verbindungen und vor allem nicht die permanente Netzwerk-Verbindung. Diese Eigenschaft ist besonders hilfreich, weil die Struktur der Teilnehmer einer allgemeinen virtuellen Gemeinschaft recht heterogen ist und die Infrastruktur offen für die Massen sein soll, die zumeist über preisgünstige und langsame Verbindungen verfügen. Der Einsatz von mobilen Agenten führt zu geringeren Belastung des Netzwerkverkehrs, weil ein mobiler Agent gleichzeitig das Suchen und Filtern der Information bei mehreren Servern ausführt und nur relevante Suchergebnisse zurückliefert, während bei der synchronen Kommunikation mehrere Flüsse zwischen dem Client und Server erforderlich sind, um eine einfache Transaktion abzuwickeln. Dieser Vorteil wird dann deutlicher, wenn die Daten auf mehreren Servern verteilt sind. Es ist viel effizienter, wenn das Programm auf die Datenquellen zugeht und Information bzw. Wissen daraus ableitet, als wenn die Daten zu dem Programm zugesendet würden. Die Daten einer VCE werden in unterschiedlichen Quellen gespeichert (Datenbank, Dokumente usw.). Ein Agent kann Dokumente automatisch registrieren und ein Verzeichnis nach definiertem Focus aufbauen und zur Verfügung stellen.

Unter dem Motto "The real members of a virtual community are creators" [Hagel, Armstrong 1997] soll die Infrastruktur für virtuelle Gemeinschaft nicht nur statische Dienste und Informationen zur Verfügung stellen. Mobile Agenten können Informationen und Dienste in einer vom Teilnehmer definierbaren individualisierten

Form verwalten und dem Teilnehmer dynamisch bereitstellen[35]. Diese Softwarelösung ist im Vergleich zu vielen anderen Internet-Anwendungen keine Abruf-Lösung („On-Demand-Anwendung"), sondern „schiebt" die Informationen und Dienste zu dem Nutzer („Push-Anwendung"). Diese Möglichkeit ist für die Anbieter als Teilnehmer der virtuellen Gemeinschaft hoch interessant, da eine genau definierte Zielgruppe mit individuellen Interessensprofilen erreicht werden kann.

Mobile Agenten bieten mehrere Vorteile gegenüber traditionellen Lösungen, besonders im Fall der Infrastruktur für virtuelle Gemeinschaften, wo Clients und Server von unterschiedlichen Leuten kontrolliert werden und unter unterschiedlichen Bedingungen arbeiten müssen. Der Einsatz von mobilen Agenten hat jedoch den Nachteil, daß die Sicherheit für Server Agenten besonders berücksichtigt werden muß und entsprechende Maßnahmen bzw. Implementierungstechniken anzuwenden sind.

Im folgenden wollen wir die Architektur einer agenten-basierten Infrastruktur für virtuelle Gemeinschaften skizzieren. Bild 6 zeigt, daß Software-Agenten zentrale Elemente einer solchen Infrastruktur sind. Mobile Agenten sind nur eine Klasse von Software-Agenten. Man kann Software-Agenten in mobile Agenten, Koordinations-Agenten, Ressource-Agenten, (User Interface) GUI-Agenten und Sicherheits-Agenten einteilen. Ein Software-Agent ist "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" [Franklin, Graesser 1996].

Software-Agenten müssen folgende Merkmale aufweisen:

- *reaktiv* - Sie reagieren in einer angemessenen Zeit auf Änderungen in der Umgebung),
- *autonom* (Sie enthalten selbst die Kontrolle über ihre Aktionen),
- *ziel-orientiert* (Sie reagieren nicht auf jede Antwort der Ungebung) und
- *zeitlich kontinuierlich* (Sie sind ein kontinuierlich laufender Prozeß).

Die Umgebungen, in denen Agenten leben und ausgeführt werden, werden als Agenten-Server bezeichnet.

[35] Ein solches Konzept kann mittels Java Dynamic Management Kit realisiert werden.

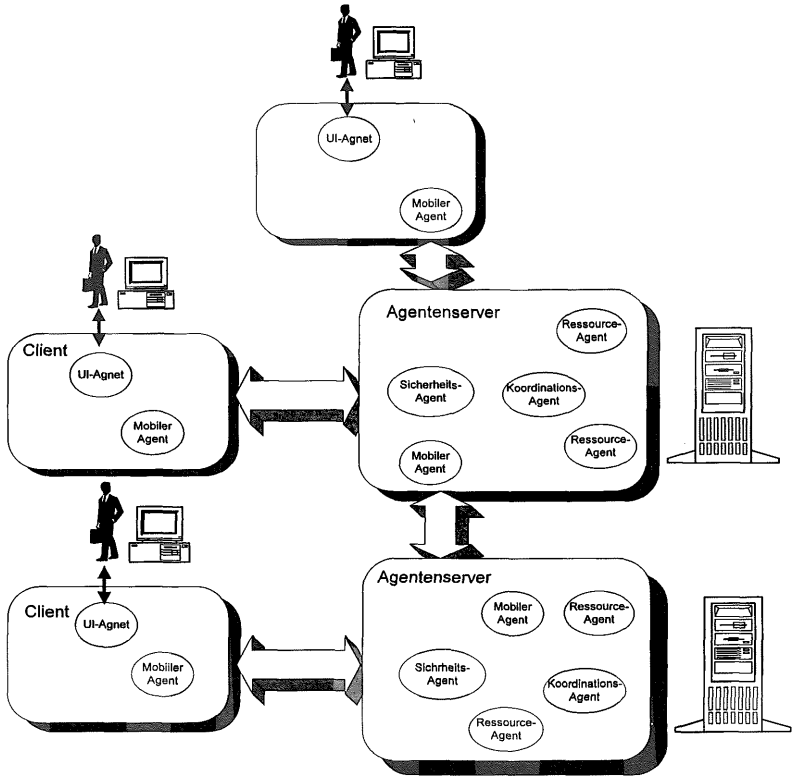

Abbildung 6: Architektur einer agenten-basierten VCE

Sicherheits-Agenten sind statische Agenten, die die Authentifikation, Validation usw. der mobilen Agenten prüfen, die in den Agenten-Server eintreten wollen. Mobile Agenten, die nicht authorisiert sind bzw. die Integrität verletzen, dürfen nicht in den Agenten-Server eintreten.

Ein Koordinations-Agent ist ein statischer Agent, der die Aktivitäten innerhalb eines Agenten-Servers koordiniert. Zu seiner Aufgabe gehört z. B. sicherzustellen, daß der Server nicht mit Agenten „überschwemmt" wird. Er vermittelt die Kontakte zwischen den Agenten, z.B. zwischen mobilen Agenten und Ressourcen-Agenten.

Ressource-Agenten sind statische Agenten, die in einem Agenten-Server existieren und liefert ein Abstraktionskonstrukt zwischen Informationsressourcen und Informationsforderungen der mobilen Agenten. Der Zweck eines Ressource-Agenten ist die Vermittlung des Informationszugriffs für einen mobilen Agenten. Er kann beispielsweise auf Forderung eines mobilen Agenten die Datenbank-Anbindung

aufbauen und dem mobilen Agenten Daten zurückliefern oder die Datenquelle eines Bulletin Boards durchsuchen und Ergebnisse bereitstellen.

Ein UI-Agent kann ein statischer oder ein mobiler Agent sein. Er übernimmt die Aufgabe der Kommunikation mit dem Nutzer. Der wesentliche Unterschied im Vergleich zu stationären UI-Objekten besteht in der Personalisierungsfähigkeit der Agenten. Der UI-Agent soll den Browser des Anwenders erkennen und die Seiten entsprechend aufbauen, weil die Frage nach der Java-Fähigkeit des Browsers oder der Sprache vom Anwender als unangenehm empfunden werden kann.

Selbständige Personalisierung ist für den Erfolg einer virtuellen Gemeinschaft bedeutend. Dieser Erfolg setzt das Erreichen der kritischen Masse der Teilnehmer, der Nutzungsprofile, Transaktionsprofile und letztlich der Transaktionen voraus [Hagel, Armstrong 1997]. Anders als bei einer Organisation im engeren Sinne besitzen die Mitglieder einer virtuellen Gemeinschaft zunächst keine solchen Informationen. Es ist aber auch nicht ratsam, den potentiellen Teilnehmer bereits beim ersten Besuch mit Fragen bzgl. seines persönlichen Profils zu konfrontieren. Eine Untersuchung der Gartner Group hat ergeben, daß 50 Prozent der Besucher eine Site sofort wieder verlassen, wenn sie eingangs aufgefordert werden, derartige Angaben zu machen. Die Personalisierung könnte von UI-Agenten übernommen werden. Sie können die Bewegung der Anwender auf der Website, das Suchverhalten, Kommunikationsverhalten usw. beobachten und Informationen über diese sammeln. Diese Information kann dann verwendet werden, um Kontakte mit dem Besucher herzustellen (z.B. durch regelmäßiges Senden von News-Lettern), bis eine Art von „Vertrauensverhältnis" zwischen Anbieter und Nachfrager entstanden ist. Erst dann ist der Kunde in der Regel bereit, Information für eine Nutzerprofile-Datenbank zu liefern. Der UI-Agent kann danach weiterhin das Nutzungsverhalten der Teilnehmer observieren, um beispielsweise Informationen über die von ihnen durchgeführten Transaktionen zu gewinnen.

Beim Einsatz von mobilen Agenten muß eine Reihe von Anforderungen erfüllt werden: Portabilität (Mobile Agenten müssen selbst portierbar sein), Ubiquität (Mobile Agenten müssen zu verschiedenen Rechnerressourcen (Agentenservern) zugänglich sein), Netzwerkkommunikation (Mobile Agenten müssen innerhalb eines Netzwerks leben können und transferiert werden können), Serversicherheit (Agentenserver müssen vor feindlichen Agenten geschützt werden), Agentensicherheit (Kann ein Agent dem Server vertrauen, bei dem er auszuführen ist ?). Im folgenden wird Java als Basis-Technologie für mobile Agenten vorgestellt.

3.3 Java als Technologie-Basis für Mobile Agenten

Mit dem Konzept der virtuellen Maschinen wird die Anforderung bezüglich der *Portabilität* von Java erfüllt. Java-Programme werden als Bytecode transportiert und von einer virtuellen Maschine intepretiert. Da die meisten Web-Browser eine virtuelle Maschine für Java enthalten, können Java-Programme (als Applets) praktisch überall vom Internet geladen und ausgeführt werden. Die *Ubiquität* wird dadurch gewährleistet. Eine der Stärke von Java ist die *Netzwerkkommunikation*. Die Datenübertragung auf dem Socket-Level, der Methodenaufruf der fernen Objekte (Remote Invocation Method RMI) gehören zur Java-Standardbibliothek. Die Integration von Java-Anwendungen mit dem sprachneutralen verteilten Objektmodell CORBA wurde von mehreren Anbietern angeboten (vgl. [15], [17] usw.). Die Integration mit DCOM wird in Zukunft auch zur Verfügung gestellt (vgl. [13])

Zur Unterstützung der *Serversicherheit* bietet Java das Konzept der Sercurity Manager, durch das definiert wird, auf welche Ressourcen ein Java-Programm zugreifen darf. Für die *Sicherheit der Agenten* liefert Java keine Mechanismen. Um das zu gewährleisten, müßten Schutzmechanismen in den Agenten bzw. Agentenservern implementiert werden.

Zur Zeit werden mehrere Systeme für mobile Agenten auf dem Markt angeboten. Aglets Workbench von IBM, Java Dynamic Management Kit von Sun, Voyager von Objectspace und Odyssey von General Magic sind einige Agenten-Plattformen, die die Entwicklung und den Einsatz der mobilen Agenten unterstützen.

Java ist nicht die einzige Technologie zur Unterstützung von mobilen Agenten. Alternativen sind ActiveX von Microsoft, TeleScript von General Magic usw. Diese alternativen Plattformen haben jedoch einige Nachteile. Der Nachteil von ActiveX ist der Mangel der Sicherheits- und Schutzmaßnahmen: ActiveX Controls sind reiner Code ohne Zugriffs-Restriktionen. TeleScript ist eine der ältesten Plattform der mobilen Agenten und wurde von General Magic entwickelt, einem Unternehmen, welches das Konzept der mobilen Agenten mitentwickelt hat. Diese Technologie wurde aber wegen der raschen Entwicklung und Verbreitung von Java überholt und von General Magic selbst durch die Java-orientierte Plattform Odyssey ersetzt. Dies zeigt, daß Java-Technologie die zukunftssicherste Plattform für mobile Agenten zu sein scheint.

4 Zusammenfassung und Ausblick

In diesem Artikel wurden die Anforderungen an die Infrastruktur einer virtuellen Gemeinschaft analysiert und eine entsprechende Architektur vorgeschlagen. Die Middleware-Technologien zum Aufbau solcher Anwendungen wurden vorgestellt. Ein neues Paradigma, das Paradigma der mobilen Agenten, wurde als Alternative zur Realisierung von VCE skizziert . Eine agenten-basierte Architektur wurde anschließend diskutiert.

Obwohl Systeme für mobile Agenten reichlich am Markt zu finden sind, ist ihr Einsatz in der Industrie nur als Einzelfall zu beobachten. Das Sicherheitsproblem ist das wichtigste Hindernis für die Verbreitung von Agenten-Anwendungen als Infrastruktur für virtuelle Gemeinschaften. Mit dem raschen Fortschritt auf dem Gebiet der Kryptographie und der ständigen Verbesserung der java-basierten Technologien ist aber zu erwarten, daß das Konzept der mobilen Agenten - zusammen mit intelligenten Agenten - zur zukünftigen Technologie für den Aufbau von virtuellen Gemeinschaften gehören wird.

Literaturverzeichnis:

[1] Goldman, S.; Nagel, R.; Preiss K.;Wernecke, : Agil im Wettbewerb. Springer-Verlag 1996

[2] Hagel, J; Armstrong, A.G.: net gain – expanding markets through virtual communities. Harvard Business School Press 1997

[3] Picot, A.; Reichwald, R.; Wigand R.T.: Die grenzenlose Unternehmung. 3. Auflage Wiesbaden Gabler 1998

[4] Page′, P.: Objektorientierte Software in der kommerziellen Anwendung. Springer Verlag 1994

[5] Bender, K.; Homann, J.: Extrakontext- und Applikationslogik in Anwendungssystemen zur Unterstützung virtueller Gemeinschaften. Dieser Tagungsband, Abschnitt B.1

[6] Fragen vergraulen den Web-Besucher. Information Week 6. August 1998. S. 48.

[7] Nortel: Joint Venture Formed to Market Digital PowerLine Technology Worldwide. (http://www.nortel.com/home/press/1998a/3_25_9898127_NorWebNA.html)

[8] Graesser, Art; Franklin, Stan: Is it an Agent, or just a Program? A Taxonomy of Internet Agents. Procedings of the Third International Workshop on Agent

Theories, Architechtures and Languages, Springer-Verlag 1996 (http://www.msci.memphis.edu/~franklin/AgentProg.html)

[9] Harrison, C.; Chess, D.M.; Kershenbaum, A.: Mobile Agents: Are they a good idea? IBM Reasearch Report 1995

[10] Wooldridge, M.; Nwana, H.: Software Agent Technologies. (http://www.labs.bt.com/projects/agents/publish/paper/sat_report.html)

[11] Hamilton, G.: JDBC™ Database Access with Java. 1997

[12] Hranitzky, N: Java und RMI. OOP'97 München.

[13] ObjectSpace: Voyager – The Agent ORB for Java. (http://www.objectspace.com)

[14] ORACLE: Acces to Relational Data from Java: JDBC and SQLJ. (http://www.oracle.com/nca/java_nca/html/sqlj_jdbc_wp.html)

[15] OHSWG: Java Beans, Java RMI & ObjectSpace Voyager Evaluation. (http://www.daimi.aau.dk/%7Ebouvin/otw/)

[16] OHSWG: DCOM Evaluation. (http://diana.ecs.soton.ac.uk/~dem97r/dcom/dcom.html)

[17] OHSWG: CORBA Evaluation. (http://www.ics.uci.edu/pub/kanderso/CORBA/)

[18] General Magic: Odyssey Information (http://www.genmagic.com/technology/odyssey.html)