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

C.3. Kooperative multimediale Anwendungen: Basis für virtuelle Arbeitsumgebungen

Dipl.-Inf. L. Kirchner
Prof. Dr. K. Meißner
Dipl.-Inf. F. Wehner
Technische Universität Dresden

Abstract:

The article starts describing synchronous, document-based groupware applications as an important basis for virtual workspaces and internet-wide distributed collaborating teams working in areas such as media design, engineering and electronic banking. New communicative and collaborative software solutions are preconditions for such scenarios. The article points out deficits of existing application sharing technology used for collaboration, suggests supplementing this technology by synchronous document-based groupware applications and lists requirements for these applications. It is described how general support for replication, synchronisation and concurrency control a development framework should provide. Furthermore the article discusses, how the parts of an interactive application should be distributed by using the MVC paradigm. An approach based on replication that combines message passing and state passing for synchronisation and a distributed optimistic concurrency control with rollback-able short transactions are outlined. For handling resulting differences in application development, an advanced observer pattern is proposed.

1 Kooperative Anwendungen und virtuelle Arbeitsumgebungen

Fast täglich entnehmen wir den Nachrichten Informationen über den Zusammenschluß, die Übernahme, Allianzen und strategische Kooperationen von Firmen, die häufig mit der Globalisierung der Märkte begründet werden. Hierdurch entstehen räumlich vernetzte, hierarchisch geprägte Organisationen mit z.T. unterschiedlicher Kultur. Zudem begegnet uns zunehmend der Begriff des ‚Virtuellen Unternehmens'. Hierunter versteht man Kooperationsformen unabhängiger, häufig kleiner und mittelständischer Unternehmen, die auf der Grundlage eines gemeinsamen Geschäftsverständnisses eine bestimmte Leistung erbringen und gegenüber Kunden oder Lieferanten wie ein einheitliches Unternehmen erscheinen. Diese neue Organisationsform sprengt die bekannten Unternehmensgrenzen nicht nur im Hinblick auf die Geschäftsprozesse, sondern auch in räumlicher wie rechtlicher Hinsicht. Sie ist geprägt durch eine problem- und kundenbezogene, dynamische Verknüpfung von Ressourcen und Abläufen, ist

häufig in Teilen, aber auch als Ganzes flüchtig und durch dynamische Rekonfiguration in der Lage, sich an hochgradig variable Aufgabenstellungen flexibel anzupassen. Ausgehend von einem gemeinsamen Geschäftsverständnis entstehen mentale und strukturelle Kopplungen innerhalb eines kundenorientierten persönlichen Netzwerkes, das die Kultur der virtuellen Organisation prägt und dessen Voraussetzung neue Informations- und Kommunikationstechnologien sowie kooperative Anwendungen sind. Einen stimulierenden Faktor bei dieser Entwicklung bildet sicherlich das Internet als firmenübergreifendes Netz und das World-Wide-Web als universeller Informationsdienst, der multimediale Präsentationsformen nutzend, sich besonders gut für neue Vertriebsformen eignet.

Jedoch nicht nur im beruflichen, sondern zunehmend auch im privaten Umfeld besteht durch die Entwicklung der Internet-Technologie der Wunsch nach Anwendungen, die Kommunikation und Kooperation mit entfernt agierenden Partnern ermöglichen. Man denke z.B. an das elektronische Einkaufen oder die Erledigung von Bankgeschäften auf elektronischem Wege. In beiden Fällen spielt die fachlich kompetente Beratung und damit die persönliche Bindung des Kunden an den Anbieter eine wichtige Rolle. Im Falle der Bankgeschäfte wird der Kundenberater situationsabhängig Experten, z.B. der Immobilien- und Anlageabteilung, zu dem Gespräch hinzuziehen wollen. Der Kunde sieht auf seinem Bildschirm die Gesprächspartner, wie auch die zur Beratung genutzten Unterlagen, Abbildungen und Anwendungsfenster, z.B. mit Ergebnissen der Finanzierungsberechnungen. Am Ende der Beratung wird er evtl. Aufträge erteilen, d.h. vom Kundenberater vorbereitete Unterlagen elektronisch unterschreiben. Andere private Anwendungen sind virtuelle Lernumgebungen (CSCL, Computer Supported Collaborative Learning) oder interaktive multimediale Unterhaltungsangebote. So entwickeln z.B. interaktive Computerspiele, die ein substantielles Marktsegment mit zweistelligen Wachstumsraten bilden, besondere Attraktivität, wenn nicht gegen den Computer, sondern mit anderen i.a. entfernten Personen gespielt werden kann.

Die zuvor genannten Anwendungsfelder sind geprägt durch Kommunikation, Kooperation, Koordination, Informationsverteilung und Gruppenbewußtsein, das diesen Merkmalen zuzuordnende interdisziplinäre Forschungsgebiet ist als ‚Computer Supported Cooperative Work' (CSCW) bekannt; die praktische Umsetzung der Erkenntnisse dieses Forschungsgebietes in IuK- (Informations- und Kommunikations-) Systemen wird als Groupware bezeichnet [1]. In diesem Umfeld werden zunehmend multimediale Präsentations- und Kommunikationsmittel wichtiger, nicht zuletzt, weil sie das Handeln in der Gruppe, die Bildung einer gemeinsamen Kultur in einem ‚Virtuellen Unternehmen', usw. fördern.

Überwiegt der Koordinationsanteil bei CSCW-Systemen, d.h. die Modellierung, Simulation, Ausführung und Steuerung von Geschäftsprozessen - also voneinander abhängige Aktivitäten -, so spricht man von ‚Workflow Systemen'. Die Kooperation zwischen Gruppenmitgliedern ist hier stark formalisiert [2].

Zukünftige CSCW-Anwendungen, insbesondere aus den oben dargestellten Anwendungsbereichen, werden stark durch informelle, häufig spontane Kooperation bestimmt sein. Die nachfolgenden Ausführungen beziehen sich auf solche CSCW-Ansätze und -Anwendungsfelder.

2 Existierende Ansätze kooperativer Anwendungen

Für kooperative Arbeit ist Kommunikation Voraussetzung und erfolgt u.a. indem sich die Gesprächspartner treffen. In virtuellen Organisationen mit potentiell global verteilten Teams sind persönliche Treffen aufwendig und kostspielig, spontane Treffen gar nicht möglich. Wünschenswert ist deshalb, solche Abstimmungsprozesse unabhängig von räumlichen Gegebenheiten unter Nutzung z.B. der jeweiligen Arbeitsplatzrechner und geeigneter Kommunikationstechnik, wie Intranet, Internet oder ISDN (Integrated Services Digital Network), durchführen zu können. Prinzipiell kann dies asynchron, über E-Mail oder Bulletin Board, aber auch synchron, unter Verwendung von Chat- oder auch Audio- und Videokonferenztechniken, erfolgen. Insbesondere bei multimedialen Konferenzsystemen sind erhöhte Anforderungen an die Hardware - d.h. den Computer und die Peripherie, wie Kamera, Sound- und Framegrabberkarte - an die Bandbreite und Latenz der Netzverbindung wie auch an die Softwarekomponenten zu stellen. Voraussetzung für die Kommunikation zwischen i.a. heterogenen Systemen ist die Unterstützung internationaler Standards, wie z.B. SMTP (Simple Mail Transfer Protocol) für e-mail oder H.323 für Audio- und Videokonferenz.

Kooperative Arbeit benötigt neben der Kommunikation auch Mechanismen für den gemeinsamen Zugriff und die konsistente Veränderung globaler Daten und Dokumente sowie bei deren synchroner Bearbeitung die Möglichkeit, gemeinsame Sichten auf die jeweiligen Anwendungszustände zu erzeugen. Die Verwaltung der Daten kann z.B. durch eine zentrale Datenbank erfolgen, die dann auch die Operationen der verteilten Anwendungen auf die Daten synchronisiert. Hierdurch wird jedoch z.B. nicht erreicht, daß alle an der kooperativen Arbeit Beteiligten eine gemeinsame Sicht auf den Zustand einer Anwendung erhalten. Häufig sind auch Dokumente nicht zentral gespeichert oder die zu ihrer Bearbeitung genutzten Programme nicht kooperativ. Um solche nur für einzelne Benutzer entworfene Programme, das sind fast alle PC-Anwendungen, in kooperativen Umgebungen gemeinsam im Abstimmungsprozeß nutzen zu können, gibt

es seit längerem das Konzept der zentralen Kommunikationsschale [3], [4], bei dem die Anwendung auf einem an der Kooperation beteiligten Rechner zentral ausgeführt wird. Dieses Konzept wurde für das X-Windows System entwickelt, ist jedoch inzwischen für andere Plattformen, wie Windows 95, verfügbar.

Bei dem Konzept der zentralen Kommunikationsschale (Bild 1) wird die grafische Ausgabe der nur auf einem Rechner ausgeführten Anwendung, an alle, an der Kooperation beteiligten Rechner verteilt (Application Sharing). Die Eingaben der einzelnen Nutzer werden an die Kommunikationsschale der Anwendungsinstanz gesendet. Dieser Ansatz wird zur Zeit von der International Telecommunication Union (ITU) als T.128 (T.SHARE) und in existierenden ‚Application Sharing' Produkten, wie

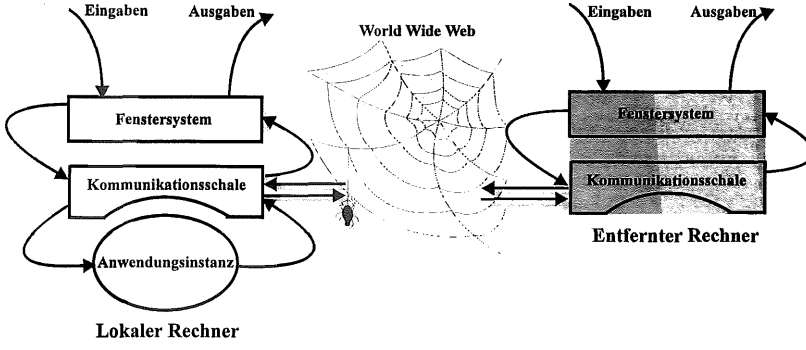

Bild 1: Zentrale Kommunikationsschale

z.B. Microsoft Netmeeting und Intel ProShare, genutzt, wobei diese Systeme auch Audio- und Video-Konferenzfunktionen unterstützen. Der Vorteil dieses Konzeptes ist, daß prinzipiell beliebige Einzelbenutzer-Anwendungen mehreren Benutzern gleichzeitig zur Verfügung gestellt werden können, wobei allerdings zu einem Zeitpunkt nur einer die Kontrolle über die Anwendung hat. Es ist damit für Kooperationsszenarien geeignet, bei denen sich die beteiligten Personen in ihrer Arbeit mit der Anwendung abwechseln, z.B. eine Person etwas präsentiert, was die anderen passiv verfolgen. Die Aktionen der verschiedenen Benutzer müssen durch Weitergabe der Steuerung (eines Tokens) serialisiert werden. Parallele kooperative Arbeit an einem gemeinsamen Dokument ist somit nicht möglich. Das Verfahren impliziert, daß stets alle Benutzer dieselbe Sicht auf das gemeinsame Dokument haben. Bei einer Präsentation ist dies erwünscht, bei kooperativer Arbeit aber möchten die Benutzer zu unterschiedlichen Stellen des Dokuments scrollen können und eigene Sichten auf das gemeinsam und gleichzeitig bearbeitete Hyperdokument haben.

Ein weiterer Nachteil ergibt sich bei dem Verfahren aus der Verteilungsarchitektur. Da alle grafischen Ausgaben der Anwendung über das Netz übertragen werden, stellt dieses Verfahren hohe Anforderungen an die benötigte Bandbreite; die Antwortzeit des Systems verlängert sich um mindestens die doppelte Netzlatenz für entfernte Benutzer. Deshalb ist es für multimediale, stark interaktive Anwendungen mit sich schnell ändernden Bildschirminhalten ungeeignet. Es sind jedoch hybride Ansätze denkbar, die einerseits das ‚Application Sharing' Konzept für entsprechende Einbenutzer-Anwendungen bereitstellen und darüber hinaus allgemeine Groupware-Anwendungen mit folgenden Eigenschaften integrieren. Eine Groupware-Anwendung

- muß Informationen darüber besitzen, welche Benutzer an welcher Stelle der Anwendung wie agieren und dieses Wissen allen anderen Benutzern zur Verfügung stellen („workspace awareness").
- muß parallele Benutzeraktionen zulassen und dafür sorgen, daß Konflikte entweder vermieden oder erkannt und aufgelöst werden (Nebenläufigkeitskontrolle).
- sollte unterschiedliche, flexibel einstellbare Grade der Kopplung zwischen Benutzern zulassen. In eng gekoppelten Sitzungen haben Benutzer identische Sichten auf das gemeinsame Dokument, in lose gekoppelten können sie an verschiedenen Dokumentteilen arbeiten und das Dokument unterschiedlich visualisieren.
- soll genau so schnell auf lokale Benutzereingaben reagieren, wie dies bei einer vergleichbaren Einzelbenutzer-Anwendung der Fall wäre („local feedback").

Multimediale Anwendungen basieren insofern auf komplexen Datenstrukturen, als diese zeitvariant, von ihrem Volumen her sehr groß und die Algorithmen zur Dekompression i.a. sehr komplex sind, z.T. bestehen Synchronisationsbeziehungen zwischen den Datentypen (Medienströmen). Gefordert ist häufig, z.B. bei WWW- (World Wide Web-) Anwendungen, ein großer Verbreitungsgrad, so daß diese Anwendungen auf unterschiedlichen, sehr heterogenen Systemen ausgeführt werden müssen. Dies wird durch einen dokumentenbasierten Ansatz erreicht, in dem die Anwendungssemantik mit einer Dokumentensprache, z.B. einer Weiterentwicklung von HTML (HyperText Markup Language) oder MHEG (Multimedia and Hypermedia information coding Expert Group), formuliert, in einem Dokumentenformat distribuiert und durch ein Runtime-System, häufig auch als Browser bezeichnet, visualisiert wird. Handelt es sich um kooperative Anwendungen, die von mehreren Benutzern synchron genutzt werden, so bezeichnet man das erforderliche Runtime-System mit den von diesem verwendeten Mechanismen des System zur Wahrung der Konsistenz (Framework) als ‚synchrone, dokument-basierte Groupware' (SDBG). Das folgende Kapitel beschreibt wesentliche

Mechanismen, z.B. Replikation und Nebenläufigkeitskontrolle, deren Kapselung in einer transparenten Systemschicht und die sich daraus für den Entwickler einer SDBG-Umgebung ergebenden Konsequenzen.

3 Konzepte und Mechanismen eines Frameworks für SDBG

Aus der Sicht eines Anwendungsentwicklers ergeben sich bei synchroner, dokumentbasierter Groupware gegenüber Einzelbenutzer-Anwendungen erhöhte Anforderungen aus zwei Gründen:

- Für den Mehrbenutzereinsatz muß insgesamt ein Mehr an Funktionen (Funktionen zur Kontaktaufnahme, Abstimmung, Kommunikation etc.) geschaffen und auch ein Mehr an Informationen (Informationen über Benutzer, deren Aktionen, Sitzungen etc.) verwaltet und angezeigt werden.
- SDBG ist verteilte Software, teilweise mit komplexen Verteilungsarchitekturen. Durch diese Verteilung ergeben sich Belange des Abgleichs und der Nebenläufigkeitskontrolle.

Der erste Aspekt soll hier nicht näher untersucht werden; es wird lediglich darauf hingewiesen, daß allein wegen dieses Punktes die Konzeption und Implementierung von Groupware sicherlich komplexer ist als die von Einzelbenutzersoftware.

Im folgenden soll es um die Probleme des zweiten Punktes gehen und es sollen allgemeine Lösungsansätze sowie deren Konsequenzen für die Anwendungsentwicklung aufgezeigt werden.

Dabei wird zunächst die Verteilungs-, dann die Abgleichsproblematik, die Nebenläufigkeitskontrolle und schließlich die notwendigen Konsequenzen für die Anwendungsentwicklung behandelt.

3.1 Verteilung

Die oben genannte Forderung nach ‚local feedback' macht es i.a. notwendig, Anteile von SDBG-Anwendungen auf die beteiligten Rechner zu verteilen. Zur Beurteilung, welche Teile diese sind, bietet es sich als grobes Modell vom Aufbau der SDBG-Anwendungen das MVC- (Model View Controller-) Muster [5], [6] an.

Nach diesem Muster kann man sich eine grafisch-interaktive Anwendung (auch eine Mehrbenutzer-Anwendung [7]) stets vorstellen als aufgebaut aus Objekten dreier Arten:

- Modell-Objekten, die die grundlegenden Daten der Anwendung und deren Beziehungen modellieren (im Fall dokumentbasierter Anwendungen modellieren die Modell-Objekte das Dokument).

- View-Objekten, die dafür verantwortlich sind, den von den Modell-Objekten modellierten Inhalt der Anwendung auf verschiedene Weisen anzuzeigen.
- Controller-Objekten, die dafür verantwortlich sind, Nutzereingaben und Interaktionen bezogen auf die View-Objekte zu möglichen Veränderungen der Modell-Objekte zu verarbeiten.

Betrachtet man in einer typischen Anwendung den Informationsfluß zwischen jeweils der Gesamtheit der View-, Controller- und Modell-Objekte sowie zum Windows-System bezogen auf das Informationsvolumen, so stellt man signifikante Unterschiede fest.

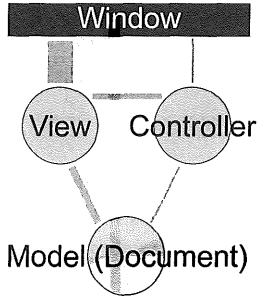

Bild 2: Datenverkehrsaufkommen in MVC

In Bild 2 wird das typische Verkehrsaufkommen zwischen den Objektsorten dargestellt. Dabei wurde von einer typischen Dokumenteditor-Anwendung ausgegangen. Die View-Objekte verwenden den meisten Speicher. Die von den Views gespeicherten Werte enthalten typischerweise redundante Information, die von den Views aus den Modell-Objekten berechnet wird und die mit dem Ziel einer besseren Effizienz der Anzeigeroutinen zur mehrfachen Verwendung gespeichert wird.

View-Objekte gehören, genauso wie Controller-Objekte, stets zu genau einem offenen Fenster. Aus Überlegungen zur Verkehrsminimierung und auch zur Lastverteilung wird einsichtig, daß View-Objekte in einem verteilten System in der Regel stets auf dem Rechner angeordnet werden sollten, zu dem auch ihr Fenster gehört. Wegen sehr häufiger und kurzer Nachrichtenanfragen zwischen Controller- und View-Objekten, sind auch Controller-Objekte verteilt anzuordnen.

Modell-Objekte enthalten in SDBG-Anwendungen das von mehreren Benutzern gleichzeitig veränderte Dokument. Da folglich von mehreren Rechnern aus auf diese Objekte zugegriffen werden muß, bietet sich zunächst als einfache Lösung deren Plazierung auf einem zentralen Server an (Bild 3). Immer, wenn Controller- oder View-

Objekte auf Modell-Objekte zugreifen, werden dann entsprechende Nachrichten über das Netz verschickt, etwa über RMI (Remote Method Invocation).

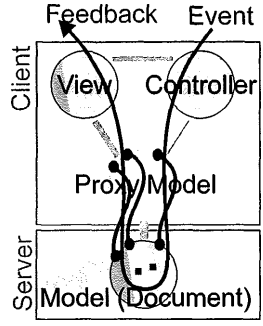

Bild 3: Entfernte Plazierung der Model-Objekte auf Server

Als allgemeine Lösung hat diese Client-Server-Architektur aber noch entscheidende Nachteile. Bei jedem neuen Zugriff durch View-Objekte auf die Modell-Objekte wird Information erneut übertragen, der typischerweise interaktive Charakter des Zugriffsverhaltens der View-Objekte führt zu Wartezeiten eines Vielfachen der Netzlatenz, Veränderungen des Dokumentes durch einen Benutzer werden auch lokal frühestens nach der doppelten Netzlatenz angezeigt. Dies hat insgesammt zur Folge, daß Anwendungen dieser Architektur z.T. relativ träge reagieren.

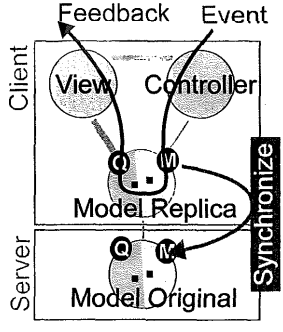

Bild 4: Replikation von Model-Objekten

Die beste, aber auch die komplizierteste allgemeine Lösung ist die Replikation der Modell-Objekte, das heißt systemüberwacht und automatisch abgeglichene Kopien der Modell-Objekte auf allen Rechnern (Bild 4). Vorteil ist, daß so im laufenden Betrieb nur die Veränderungen an den Modell-Objekten jeweils übertragen werden müssen und

daß alle Arbeitsplätze auch bei zwischenzeitlichen Unterbrechungen des Netzwerkes voll lauffähig bleiben.

3.2 Abgleichsverfahren

Um einen automatischen Abgleich der Replikate zu erreichen, ist es unter anderem notwendig, daß das System Änderungen an den replizierten Modell-Objekten protokolliert. Es gibt zu diesem Zweck zwei gängige Verfahren:

- Message-Passing: Das System protokolliert Nachrichten von Controller-Objekten an Modell-Objekte. Diese Nachrichten werden dann nicht nur an das Original, sondern auch an alle seine Replikate versendet, wodurch bei allen Replikaten der gleiche Effekt erzielt werden soll.
- State-Passing: Das System protokolliert bei den Modell-Objekten alle Änderungen an den Objektvariablen und nimmt an allen Replikaten direkt analoge Änderungen vor.

Message-Passing ist bezüglich des resultierenden Netzverkehrs in der Regel effizienter und hat den Vorteil, daß auch bei allen Replikaten zum Herbeiführen der Veränderung der Objektcode ausgeführt wird, so daß das Konzept der Objektkapselung eingehalten wird. Bei State-Passing dagegen ist dies nicht der Fall, so daß replizierten Objekten die Veränderung nicht in der gewohnten Weise „bewußt" wird. State-Passing hat allerdings den entscheidenden Vorteil absoluter Sicherheit und funktioniert auch im Fall von Nebeneffekten in teilreplizierten Objektstrukturen.

Um die Vorteile beider Verfahren optimal zu kombinieren, sind Kompromißansätze geeignet, bei denen der Anwendungsprogrammierer seine Objektklassen von vordefinierten abstrakten Framework-Klassen ableitet. Von diesen Klassen abgeleitete Objekte haben eine innere Schale mit vordefinierten Methoden, die frei von Seiteneffekten und deterministisch sind, die den Zugang zu den Objektvariablen kapseln und die vom Anwendungsprogrammierer nicht verändert werden dürfen. Bei einem solchen Ansatz werden dann die Aufrufe dieser vordefinierten Methoden protokolliert und an die Replikate weitergereicht.

3.3 Nebenläufigkeitskontrolle

Nebenläufigkeit, das heißt die scheinbar oder tatsächlich gleichzeitige Ausführung mehrerer Prozesse tritt in SDBG dadurch auf, daß in der Regel über mehrere Fenster von mehreren Benutzern gleichzeitig Veränderungen vorgenommen werden. Ein Anwendungsprogrammierer ist damit überfordert, alle bei Nebenläufigkeit vorkommenden Situationen zu überschauen und zu behandeln. Es müssen daher Konstrukte verwendet werden, die den Umgang mit Nebenläufigkeit vereinfachen [8]. Ein Konstrukt, das für den Fall von SDBG geeignet ist, sind Transaktionen, eine Art Klammerung zusammengehöriger Operationen zu Paketen. Innerhalb von Transaktionen darf der Anwendungsprogrammierer annehmen, daß keine Nebenläufigkeit existiert und das System garantiert, daß die Operationspakete ganz oder gar nicht ausgeführt werden (Nebenläufig ausgeführte Transaktionen stehen potentiell im Konflikt, entstandene Konflikte müssen aufgelöst werden).

In einem replizierenden System kann bei der Nebenläufigkeitskontrolle pessimistisch oder optimistisch vorgegangen werden. Bei einer pessimistischen Vorgehensweise wird vor dem Ausführen einer Transaktion sichergestellt, daß sie auf allen Replikaten ausgeführt werden kann. Bei einem optimistischen Verfahren wird eine Transaktion lokal sofort ausgeführt. Dabei wird riskiert, daß mit Verzögerung festgestellt wird, daß die lokal bereits ausgeführte Transaktion auf einem anderen Rechner nicht ausgeführt werden konnte und deshalb auch lokal zurückgesetzt werden muß.

Die Nebenläufigkeitskontrolle hat einen Effekt auf die Benutzungsschnittstelle einer SDBG-Anwendung [9]. In der Regel werden zumindest für manche Benutzungsinteraktionen optimistische Verfahren benötigt, um eine hinreichend schnelle Reaktion von grafisch interaktiven Benutzungsschnittstellen zu gewährleisten.

Genauso wie für den Replikatsabgleich muß das System auch für eine optimistische Nebenläufigkeitskontrolle Zugriffe auf Modell-Objekte protokollieren und auswerten. Dabei muß das System Kenntnisse darüber haben, welche Operationen mit welchen anderen in Abhängigkeit stehen, so daß Konflikte auftreten können. Letzteres ist beim Kompromißansatz deutlich leichter möglich als bei message-passing, weil die protokollierten Methoden zum Framework gehören und im Framework anwendungsunabhängige Konfliktabellen angegeben werden können [10]. Auch das sichere Zurücknehmen von Transaktionen, das beim optimistischen Ansatz möglich sein muß, ist beim Kompromißansatz ohne zusätzliche Belastung des Anwendungsprogrammierers möglich, denn es können zu framework-eigenen Methoden auch framework-eigene inverse Methoden angegeben werden.

3.4 Bedeutung für multimediale SDBG

Bei multimedialen Dokumenten entstehen in erster Linie dadurch neue Anforderungen, daß i.a. große und z.T. zeitvariante Medienobjekte gehandhabt werden müssen. Wendet man auch hier sinngemäß das MVC-Muster an, erhält man teilweise sehr speicherintensive Modellobjekte.

Diese Modellobjekte können in der Regel nicht voll repliziert, sondern müssen nach Bedarf geladen werden. Da mit dem Ladevorgang häufig Zeit- und Synchronisationsbedingungen, auch zwischen verschiedenen Benutzerrechnern, verbunden sind, ist es erforderlich, auch hierfür spezielle Framework-Konzepte einzuführen. Ohne diese ist die Realisierung kooperativer multimedialer Anwendungen ökonomisch nicht möglich.

Hierzu ist es einerseits sinnvoll, die Plazierung dieser Objekte für den Anwendungsprogrammierer weiterhin transparent zu halten. Um dem System andererseits die notwendigen Kriterien zur automatischen Plazierung der Objekte mitzuteilen, besteht die Möglichkeit, das Beobachter-Muster so zu erweitern, daß der Anwendungsprogrammierer genauer angibt, wie die View-Objekte beobachten werden. Diese Angaben müssen erlauben, daß das System Rückschlüsse über das zu erwartende Zugriffsverhalten der Views ziehen kann. Solche Mechanismen existieren aber in heutigen Frameworks noch nicht und sind Gegenstand der Forschung.

3.5 Konsequenzen für die Anwendungsentwicklung

In den vorangegangenen Kapiteln wurden grundlegende Probleme verteilter Groupware dargestellt und es wurde herausgestellt, daß aus der Sicht dieser Probleme der Kompromißansatz, d.h. die ausschließliche Protokollierung bestimmter im Framework vordefinierter Methoden für den Replikatsabgleich und die Nebenläufigkeitskontrolle eine zentrale Rolle spielt.

Insgesamt entstehen für den Anwendungsprogrammierer folgende Konsequenzen: Er muß

- seine Anwendung in für alle Rechner gemeinsame Modell-Objekte und auf den verschiedenen Rechnern in lokale View- und Controller-Objekte aufteilen.
- zusammengehörende Veränderungen an Modell-Objekten in Transaktionen klammern.
- seine Modell-Objekte von allgemeinen Framework-Klassen ableiten, die den Zugriff auf Objektvariablen durch eine innere Methodenschale kapseln.

Eine weitere wichtige Konsequenz ergibt sich daraus, daß der Anwendungsprogrammierer damit rechnen muß, daß Veränderungen an Modell-Objekten auch durch Replikatsabgleich und Nebenläufigkeitskontrolle (Rücknahmen) entstehen. Diese direkten Änderungen finden zunächst statt, ohne daß Code des Anwendungsprogrammierers beteiligt wird. Es soll aber in der Regel auch auf solche Änderungen mit einer Aktualisierung der View-Objekte reagiert werden.

Hierfür bietet sich eine entsprechend angepaßte Version des „Beobachter-Musters [11] an. Das System muß dazu so erweitert werden, daß auch bei solchen Änderungen von Modell-Objekten angemeldete Beobachter geeignet benachrichtigt werden.

Für den Anwendungsprogrammierer entsteht dann zusätzlich die Konsequenz, daß er seine View-Objekte gemäß der im Framework festgelegten Konventionen als Beobachter anmelden muß.

Die hier genannten Konsequenzen stellen, verglichen mit heute üblichen Programmiergepflogenheiten bei der Entwicklung von Einbenutzersystemen, geringe zusätzliche Anforderungen an den Anwendungsprogrammierer. Die vorgestellten Grundlagen sind daher geeignet, die Programmierung komplexer SDBG-Anwendungen zu erlauben und es dem Programmierer zu ermöglichen, sich auf die wesentlichen Schwierigkeiten bei der Entwicklung von Groupware, wie die Gestaltung der um mehrbenutzerspezifische Funktionen und Informationen erweiterten Benutzungsschnittstellen, zu konzentrieren.

4 Zusammenfassung und Ausblick

Gesellschaftliche, organisatorische und technologische Veränderungen führen zunehmend zu neuen Anwendungsgebieten, in denen multimediale, kommunikative und kooperative Techniken die Basis für geeignete Softwarelösungen sind. Als Beispiel sind virtuelle Unternehmen und Organisationen zu nennen, in denen Teams über Standorte hinweg (ko)operieren. Verschiedenste Lösungen für die Kommunikation innerhalb der Organisationen existieren, angefangen von e-mail bis hin zu in den Arbeitsplatz integrierten audiovisuellen Konferenzsystemen. Zur Unterstützung formaler Arbeitsvorgänge gibt es seit längerem die verschiedensten Formen von Workflow Systemen. Das ‚World Wide Web' hat sich zudem in diesem Umfeld als das weltumspannende, professionelle wie auch private Nutzer erreichende, multimediale Informationssystem etabliert. Ein Defizit existiert im Bereich der CSCW-Anwendungen, die in optimaler Weise nicht-formale Zusammenarbeit unterstützen und sich multimedialer Dokumente und WWW-Techniken bedienen. Ein wesentlicher Grund hierfür ist, daß die Entwicklung insbesondere synchroner Groupware-

Anwendungen relativ aufwendig und komplex ist. Der hier vorgestellte Ansatz verspricht, die Komplexität des Entwicklungsprozesses wie auch den Entwicklungsaufwand für synchrone, dokumenten-basierte Anwendungen durch das vorgeschlagene Modell und die Separierung wiederverwendbarer Komponenten in einem Application Framework substantiell zu reduzieren und leistet so einen Betrag dazu, daß zukünftige innovative Sofwareprodukte kooperatives Arbeiten vermehrt unterstützen.

Literatur

[1] Borghoff, U.; Schlichter, J.:
Rechnergestützte Gruppenarbeit - eine Einführung in verteilte Anwendungen;
Berlin: Springer-Verlag, 1995

[2] Teufel, S.:
Computerunterstütze Gruppenarbeit - eine Einführung.
In: Österle, H.; Vogler, P.: Praxis des Workflow-Managemnts - Grundlagen, Vorgehen Beispiele. Braunschweig: Vieweg Verlag, 1996

[3] Mayer, E.;
Synchronisation in kooperativen Systemen;
Friedr. Viehweg & Sohn Verlagsgesellschaft mbH, 1994. S. 20-23

[4] Minenko, W.:
The Application Sharing Technology.
http://www.motifzone.com/tmd/articles/XpleXer/XpleXer.html

[5] Krasner, G. E., Pope S. T.;
A Cookbook for Using the Model-View-Controller User Interface Paradigm in Smalltalk;
Journal on Object Oriented Programming, August/ September 1988.

[6] Gamma, E., Helm, R., Johnson, R., Vlissides, J.;
Entwurfsmuster. Bonn: Addison-Wesley, 1996

[7] Schuckmann, C., Kirchner, L., Schümmer, J., Haake, J.M. Designing;
Object-oriented synchronous groupware with COAST;
Proceedings of the ACM, 1996, Conference on Computer Supported Cooperative Work (CSCW'96), Boston, Massachusetts, November 16-20, 1996, 30-38

[8] Lea, D.:
Concurrent Programming in Java: Design Principles and Patterns.
Addison Wesley, 1996

[9] Greenberg, S., Marwood, D.;
Real time groupware as a distributed system: Concurrency control and its effect on the interface;
Proceedings of the ACM 1994 Conference on Computer Supported Cooperative Work (CSCW'94) (Chapel Hill, N.C., USA, October 22-26, 1994), 207-218.

[10] Wehner, F.;
Entwicklung eines Framework-Kerns für multimediale Mehrbenutzeranwendungen;
Diplomarbeit. TU Dresden, 1997. S. 30-37

[11] Kirchner, L., Schuckmann, C.;
Groupware Developers Need More Than Replicated Objects;
Proceedings of the OO-Groupware-Platforms-Workshop, European Conference on Computer Supported Cooperative Work, Lancaster, UK, September 7-11, 1997, ISBN 90-75176-13-9.