Ein Anwendungsrahmenwerk für synchrone kollaborative Anwendungen in mobilen Umgebungen

Jörg Roth
Universität Hagen, Fachbereich Informatik

Zusammenfassung

Für die Unterstützung mobiler Anwender in synchronen Gruppensitzungen reichen etablierte Groupware-Konzepte oft nicht aus. Mobile Anwender sind mit der Groupware-Infrastruktur nur schwach verbunden. Kleine mobile Endgeräte wie PDAs verfügen oft nicht über genügend Rechenleistung, um notwendige Aufgaben der Koordination und Synchronisation der Anwender zu bewältigen. Zusätzlich unterliegen mobile Endgeräte anderen Benutzungsparadigmen wie stationäre Rechner, haben beispielsweise Benutzungsschnittstellen mit sehr reduzierten Möglichkeiten. In diesem Papier wird ein Rahmenwerk für mobile synchrone Groupware-Anwendungen auf der Basis von *Ressourcen* vorgestellt. Das Konzept wurde durch die Plattform *Pocket DreamTeam* realisiert und anhand mehrerer Beispielanwendungen überprüft.

1 Einleitung

Synchrone Groupware spielt eine wesentliche Rolle beispielsweise bei der gemeinsamen Erstellung von Dokumenten oder der kooperativen Software-Entwicklung. Groupware-Plattformen helfen, die Entwicklungskosten für synchrone Groupware drastisch zu senken, indem sie eine Reihe von Aufgaben im Rahmen von synchronen Gruppensitzungen übernehmen. Ein Anwendungsentwickler kann sich dadurch auf die anwendungsspezifischen Details konzentrieren. Durch die Senkung der Entwicklungskosten können Ansätze des Rapid Prototyping angewendet werden. Verfügt eine Plattform über genügend Ausdrucksmöglichkeiten, können schnell und effizient lauffähige Prototypen bereitgestellt werden. Damit kann sehr früh der Endanwender in den Entwicklungsprozess eingebunden werden. Eine Groupware-Plattform umfasst üblicherweise drei Bereiche (Roseman & Greenberg 1996; Dewan & Choudhary 1992): ein *Anwendungsrahmenwerk* gibt dem Entwickler ein Gerüst für die Anwendungsentwicklung vor; ein *Laufzeitsystem* stellt Dienste wie Gruppen- und Sitzungsmanagement zur Laufzeit bereit; eine Reihe von *Schnittstellen*, *Abstraktionen* und *Objekten* erlauben dem Entwickler die Nutzung von Plattformdiensten und verbergen dabei Implementierungsdetails.

Durch die Mobilität von Endbenutzern sind jedoch einige etablierte Konzepte existierender Plattformen nicht mehr anwendbar, die auf stationären Arbeitsplätzen mit hohen Rechenleistungen, komfortablen Benutzungsschnittstellen und zuverlässigen, breitbandigen Netzwerken beruhen. In diesem Papier wird ein Anwendungsrahmenwerk vorgestellt, das speziell für mobile Anwender in synchronen Gruppensitzungen konzipiert wurde. Dabei haben wir folgende Randbedingungen unterstellt:

- Synchrone Sitzungen werden sowohl aus mobilen als auch aus stationären Teilnehmern gebildet.
- Mobile Teilnehmer verfügen über mobile Endgeräte wie PDAs oder Handhelds (Abbildung 1). Notebooks kommen zwar prinzipiell auch in Frage, ihre Rechenleistungen und Dialogfähigkeiten entsprechen jedoch weitgehend denen der stationären PCs. Notebooks werden daher hier nicht diskutiert.
- Die Anbindung der mobilen Endgeräte erfolgt drahtlos. In unserer Testumgebung verwenden wir Wireless LAN, aber auch Mobilfunknetze (GSM, später UMTS) oder drahtlose Personenbereichsnetze (z.B. Bluetooth, IrDA) kommen in Frage. Wir gehen von der Existenz eines stationären Kernnetzwerks aus.
- Die Unterstützung von Stream-Medien (z.B. Audio und Video) sind nicht Gegenstand dieser Arbeit. Es wird vorausgesetzt, dass ein entsprechender Kommunikationskanal für die Sitzungsteilnehmer vorliegt, z.B. auf der Basis eines Mobilfunknetzes.

Abbildung 1: PDA mit einer kollaborativen Anwendung

2 Verwandte Arbeiten

Die Entwicklung von Groupware-Plattformen für synchrone Gruppenarbeit hat eine lange Tradition. Die Plattformen unterscheiden sich oft signifikant durch die Abstraktionen, die dem Entwickler zum Ausdruck von Groupware-Aspekten angeboten werden. *Habanero* (Chabert et al. 1998) bietet beispielsweise eine Verteilung von Benutzerereignissen an und verwendet zur Synchronisation einen zentralen Server. *Groupkit* (Roseman & Greenberg 1996) verwendet einen Server nur zum Sitzungsmanagement, Anwendungen laufen repliziert. Während die meisten Plattformen auf Mehrbenutzervarianten von MVC basieren (Graham 1996, Schuckmann et al. 1996), benutzt *Rendezvous* das ALV-Modell (Hill et al. 1993), das es den Entwickler explizit erlaubt, Konsistenzbedingungen zwischen dem Datenmodell und der Benutzungsschnittstelle auszudrücken. Eine Reihe weiterer Modelle gehen auf das PAC-Modell (Coutaz 1997) zurück, insbesondere PAC* berücksichtigt dabei Groupware-Aspekte.

Allen vorgenannten Modellen und Plattformen ist gemein, dass sie eine sehr idealisierte Sicht der Netzwerke und involvierten Rechner einnehmen. Insbesondere bleiben mobile Anwender unberücksichtigt. Einige neuere Plattformen schwächen den Begriff der synchronen Kooperation ab, um Mobilitätsaspekte auszudrücken. Für *QuickStep* (Roth & Unger 2001) wurde beispielsweise der Begriff der *relaxierten synchronen Kooperation* eingeführt, bei der mobile Teilnehmer lose an die Kommunikationsinfrastruktur angekoppelt sind. Weitere Plattformen wie *Sync* (Munson & Dewan 1997), *Coda* (Kistler & Satyanarayana 1992) oder *Rover* (Joseph et al. 1997) konzentrieren sich dagegen auf die Konfliktbehandlung bei asynchronen Zugriffen auf gemeinsame Daten im mobilen Umfeld.

3 Pocket DreamTeam

Um die Auswirkungen der Mobilität von Endbenutzern in synchronen Gruppenumgebungen zu untersuchen, erweiterten wir unsere eigene Groupware-Plattform *DreamTeam* (Roth 2000; Roth 2000b). DreamTeam wurde zuerst für stationäre Benutzer entworfen, beispielsweise für Mitarbeiter der Universität, aber auch für Studenten, die sich von zu Hause über Modem einwählen. Durch die Plattformerweiterung *Pocket DreamTeam* werden erstmals mobile Teilnehmer unterstützt. Pocket DreamTeam erweitert dabei das Laufzeitsystem, die Entwicklungsumgebung und das Anwendungsrahmenwerk. Bevor wir konkret auf die mobile Erweiterung eingehen, skizzieren wird kurz die stationäre Variante von DreamTeam.

DreamTeam basiert auf einer vollständig dezentralen Verteilungsarchitektur, d.h. neben den eingesetzten Arbeitsplatzrechnern der Gruppenmitglieder werden keine weiteren Rechner (z.B. Server) benötigt. Das Laufzeitsystem von DreamTeam bietet verschiedene Dienste zur Koordination der Teilnehmer untereinander an, u.a. Gruppen- und Benutzerverwaltung, Sitzungsmanagement und Ankündigungsdienste. Vorgefertigte Elemente zur Erzeugung von Gruppenbewusstsein, z.B. Teilnehmerlisten, verteilte Mauszeiger oder Überblicksfenster, können auf einfache Weise vom Entwickler in die Anwendung integriert werden. Entwickelt werden DreamTeam-Anwendungen in Java. Hierzu steht dem Entwickler eine Klassenbibliothek mit ca. 200 Klassen zur Verfügung.

3.1 Das stationäre Anwendungsmodell

Anwendungen unter DreamTeam werden gemäß dem *DreamTeam Ressourcen-Modell* (*DRM*) entwickelt (Roth 2000b). Sie bestehen aus einem *Anwendungsrahmen*, einer Menge von *Ressourcen* und einer *Benutzungsschnittstelle*. Der Anwendungsrahmen fügt alle Komponenten zusammen und liefert eine Schnittstelle für das Laufzeitsystem, z.B. zur Initialisierung. Über diese Schnittstelle kann das Laufzeitsystem auch Anwendungen *privat* starten. Damit kann ein Benutzer beispielsweise in einem privaten Bereich Dokumente für eine kollaborative Sitzung vorbereiten.

Ressourcen sind die Einheiten einer Groupware-Anwendung, die den gemeinsamen Zustand der Sitzung repräsentieren. Ressourcen sind beispielsweise Textpassagen von gemeinsamen Textdokumenten, Diagrammelemente eines Diagramms, gemeinsam betrachtete Web-Seiten oder Folien einer Folienpräsentation. Ressourcen stellen die Daten und die notwendigen Funktionen zur gemeinsamen Bearbeitung bereit. Sie besitzen drei Schnittstellen:

- Eine *interne Schnittstelle*: hierbei handelt es sich um die Methodenschnittstelle, die eine Ressource durch ihre Implementierung vorgibt. Diese Schnittstelle können alle Objekte der Anwendung gemäß den Regeln des Methodenaufrufs nutzen.
- Eine *externe Schnittstelle*: diese Schnittstelle wird von den korrespondierenden Ressourcen auf anderen Rechnern durch sogenannte *Ort-zu-Ort-Aufrufe* genutzt. Diese Aufrufe sind Methodenaufrufe, die auf allen replizierten Ressourcen synchron durchgeführt werden. Der Entwickler kennzeichnet Ort-zu-Ort-Aufrufe im Programmtext durch ein bestimmtes Schlüsselwort.
- Die *Systemschnittstelle*: eine Ressource muss Dienste anbieten, die es dem Laufzeitsystem ermöglichen, Kontrolle über die Ressource zu erlangen. Damit kann das Laufzeitsystem beispielsweise automatisch den Zustand einer Ressource an Neueinsteiger übermitteln oder die Konsistenz beim konkurrierenden Zugriff auf gemeinsame Daten erhalten.

Die Architektur von Anwendungen ist in Abbildung 2 dargestellt.

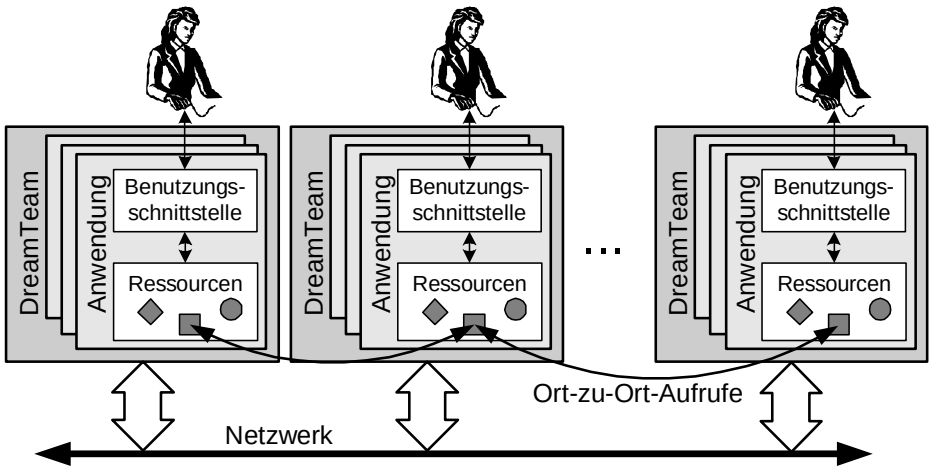

Abbildung 2: Stationäre DreamTeam-Anwendungen

Diese Architektur hat verschiedene Vorteile. Innerhalb einer Anwendungsinstanz können Ressourcen wie Objekte einer Einbenutzer-Anwendung angesprochen werden. Ein Anwendungsentwickler geht dadurch mit vertrauten Paradigmen aus der Software-Entwicklung um. Durch

die Organisation der Ressourcen können nicht nur mehrere Anwendungen parallel in einer Sitzung ausgeführt werden, es ist auch möglich mehrere Anwendungsinstanzen *derselben* Anwendung zu öffnen.

Über die Systemschnittstelle übt das Laufzeitsystem die notwendige Kontrolle über die Ressourcen aus. Der Entwickler muss sich deshalb nicht um die Verteilung der Daten, die Versorgung von Neueinsteigern, der Synchronisation der Zugriffe etc. kümmern. Insbesondere wird der komplexe Bereich der Kommunikation vollständig vor dem Entwickler verhüllt.

Das Ressourcen-Modell stellt ein Rahmenwerk dar, mit dem die verschiedensten Anwendungsmodelle wie MVC, PAC oder ALV ausgedrückt werden können. Ressourcen sind von ihrer Komplexität nicht beschränkt. So können Ressourcen selbst wieder aus Ressourcen aufgebaut werden, stellen damit kleine Anwendungen innerhalb einer Anwendung dar. Hierarchisch aufgebaute Ressourcen bilden die Grundlage des Komponentenkonzeptes von DreamTeam (Roth & Unger 2000).

Weitere Einschränkungen über die konkrete Strukturierung der Anwendungen werden bewusst nicht gemacht, um dem Entwickler die notwendige Flexibilität zu bieten. So wird in (Roth 2000b) beispielsweise ein Ansatz diskutiert, auch Teile der Benutzungsoberfläche durch Ressourcen darzustellen. Damit können Dialogelemente zur Erzeugung von Gruppenbewusstsein wie verteilte Mauszeiger oder Überblicksfenster sehr effizient entwickelt werden.

3.2 Mobile Sitzungsteilnehmer

Durch die Mobilität einiger Sitzungsteilnehmer wird eine Reihe neuer Probleme aufgeworfen, die weitreichende Auswirkungen auf die Laufzeitumgebung, auf das Anwendungsrahmenwerk und auf die Anwendungsentwicklung haben.

Tragbare mobile Endgeräte haben reduzierte Möglichkeiten der Benutzungsschnittstelle, z.B. eine geringe Größe und Auflösung des Bildschirms und keine oder nur eine rudimentäre Tastatur. Daher können gewohnte Paradigmen aus der Welt stationärer Rechner bei der Gestaltung von Benutzungsschnittstellen oft nicht direkt auf mobile Rechner übertragen werden. Überlappende Fenster, Ikonen, Drag-and-Drop, Kontextmenüs etc. sind auf mobilen Rechnern nur bedingt einsetzbar. Insbesondere das Konzept der direkten Manipulation ist problematisch (Kristoffersen & Ljungberg 1999). Um die Übersicht auf kleinen Bildschirmen zu erlangen, wurden für PDAs oder Handhelds ganz spezielle Sätze von Dialogelementen und Designrichtlinien entwickelt, die sich von denen der traditionellen Computer oft signifikant unterscheiden.

Mobile Endgeräte haben in der Regel eine mobile Stromversorgung. Batterielebenszeiten stellen immer noch einen begrenzenden Faktor dar. Im Dauerbetrieb reichen aktuelle Batterien oft nur für einige Stunden. Das mobile Endgerät ist deshalb meistens ausgeschaltet oder befindet sich in einem stromsparenden Modus mit reduzierter Aktivität. Mobile Endgeräte sind in der Regel drahtlos angebunden. Drahtlose Netzwerke haben bzgl. der Kriterien Bandbreite, Latenzzeit und Verlässlichkeit sehr ungünstige Eigenschaften.

Schließlich sind weitere technische Eigenschaften mobiler Endgeräte zu beachten. Sie haben verglichen mit Schreibtischrechnern geringe Prozessorleistungen, kleine Speicher und keine oder nur rudimentäre Dateisysteme. Betriebssysteme mobiler Endgeräte liefern dem Entwickler nur einen geringen Teil der Dienste, die man von Betriebssystemen stationärer Rechner gewohnt ist.

Den Problemen stehen teilweise hohe Anforderungen gegenüber, die Benutzer an eine mobile Anwendung stellen. So erwarten Benutzer trotz der Restriktionen kurze Antwortzeiten von den eingesetzten Anwendungen. In (Bey et al. 2001) wird eine Zeit von einer Sekunde als Maximum angesehen.

Ein Benutzer erwartet darüber hinaus, dass er das mobile Gerät jederzeit ausschalten und später mit dem aktuellen Zustand weiterarbeiten kann. Hier unterscheiden sich mobile Endgeräte signifikant von stationären Computern: während man bei Bürotätigkeiten den Rechner über Stunden angeschaltet lässt, werden mobile Endgeräte mehrfach ein- und ausgeschaltet, häufig laufen sie dabei nur für einige Sekunden.

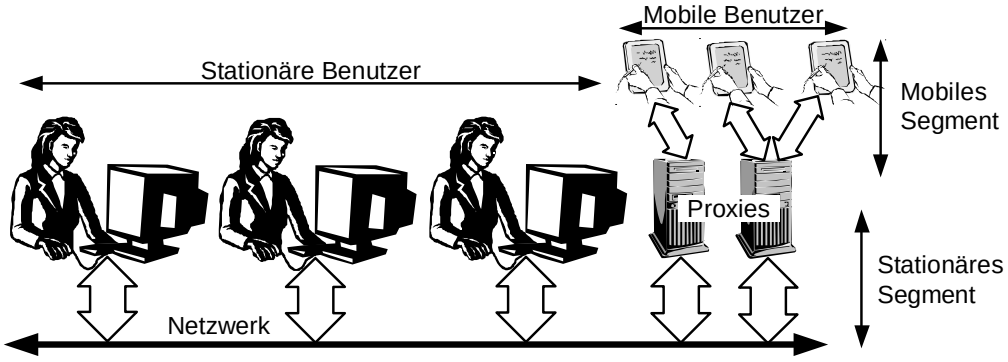

Abbildung 3: Verteilungsarchitektur von Pocket DreamTeam

Die Restriktionen auf der einen Seite und die berechtigten Forderungen auf der anderen Seite führen zu einer Architektur, wie sie in Abbildung 3 dargestellt ist.

Ein mobiler Benutzer erfordert in dieser Architektur den Einsatz zweier Rechner: neben dem mobilen Endgerät wird ein weiterer Rechner, der *Proxy*, benutzt. Der Proxy führt rechenaufwendige Operationen der Anwendung durch. Während das mobile Endgerät ausgeschaltet oder die Verbindung unterbrochen ist, wird der Zustand der Sitzung konsistent mitgeführt. Für die stationären Sitzungspartner stellt der Proxy einen ständig verfügbaren Ansprechpartner dar. Dabei läuft der Proxy ohne Benutzereingriff. Durch diese Architektur können beliebige Kombinationen aus mobilen und stationären Benutzern in Sitzungen zusammenarbeiten. Aus der Sicht des Netzwerks verhalten sich beide Benutzertypen identisch.

Proxy-Rechner stellen auf den ersten Blick einen Bruch in der dezentralen Architektur von DreamTeam dar. Allerdings sieht das Konzept vor, eine beliebige Anzahl von Proxy-Rechnern im Netzwerk einzurichten, die bei Bedarf von mobilen Benutzern in Anspruch genommen werden können. Insbesondere bedeutet der Ausfall eines Proxy-Rechners nicht, dass ein mobiler Benutzer von der Sitzung abgekoppelt ist. Automatische Wiederanlaufmechanismen erlauben das nahezu verzögerungsfreie Umschalten auf einen anderen Proxy.

Durch die Einführung von Proxy-Rechnern muss das Anwendungsrahmenwerk angepasst werden (Abbildung 4).

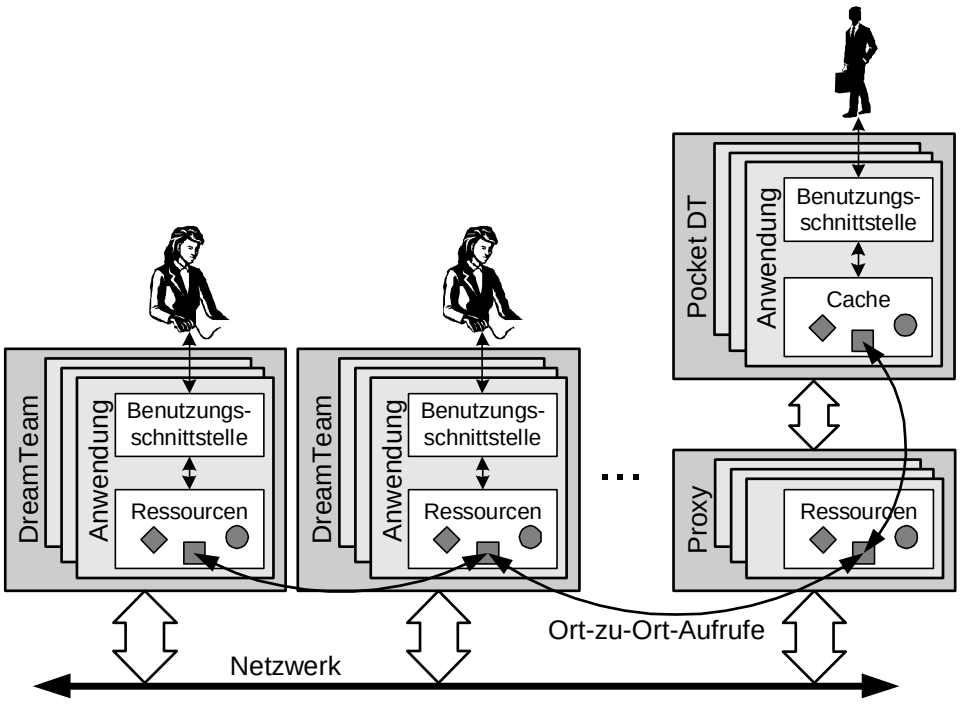

Abbildung 4: Erweiterte Architektur von Anwendungen

Ein Proxy-Rechner verwaltet die Ressourcen für einen mobilen Benutzer. Ort-zu-Ort-Aufrufe werden im Proxy durchgeführt. Somit bleibt der Zustand aktuell, auch wenn die Verbindung zum mobilen Rechner unterbrochen ist.

Der mobile Rechner besitzt von jeder Ressource einen Cache-Eintrag. Damit kann die mobile Anwendung lokal auf die Daten der Ressourcen zugreifen, ohne langwierige Netzwerktransaktionen durchzuführen. Zusätzlich kann bei Unterbrechungen zumindest eingeschränkt weitergearbeitet werden.

Dieser Komfort wird allerdings durch komplexe Protokolle zur Datenverteilung, Cache-Kohärenz und Konsistenzerhaltung erkauft, die in (Roth 2002) beschrieben werden. Insbesondere die Konsistenzerhaltung stellt ein großes Problem dar: für optimal angebundene Benutzer in stationären Netzen eignen sich pessimistische Verfahren, da sie einerseits sehr effizient arbeiten, andererseits vom Entwickler kaum Realisierungsaufwand erfordern. Bei mobilen, schlecht angebundenen Benutzern versagen jedoch pessimistische Verfahren. Als Lösung bietet Pocket DreamTeam eine Kombination aus einem pessimistischen Verfahren für das stationäre Segment und einem optimistischen Verfahren für das mobile Segment. Das Laufzeitsystem von Pocket DreamTeam hält die Problematik der Konsistenzerhaltung weitgehend vom Anwendungsentwickler fern.

3.3 Die Entwicklung mobiler kollaborativer Anwendungen

Die Entwicklung von Software für mobile Endgeräte ist prinzipiell kostenintensiver als die Entwicklung für stationäre Rechner. Bei der Entwicklung von Groupware, die sowohl stationär als auch mobil laufen soll, entsteht konkret noch ein weiteres Problem: sowohl die Betriebssystem- als auch die Programmierumgebung sind verschieden. So werden DreamTeam-Anwendungen unter Java, Pocket DreamTeam-Anwendungen jedoch unter C++

entwickelt. Plattformübergreifende Ansätze, z.B. Java ME, sind zwar prinzipiell angedacht, erweisen sich jedoch in der Realität derzeit als nicht tragfähig.

Trotz dieser Einschränkungen sollte mit Pocket DreamTeam der Rapid-Prototyping-Ansatz nicht aufgeben werden. Erreicht wird dieses Ziel durch eine starke Wiederverwendung von eingesetztem Quellcode.

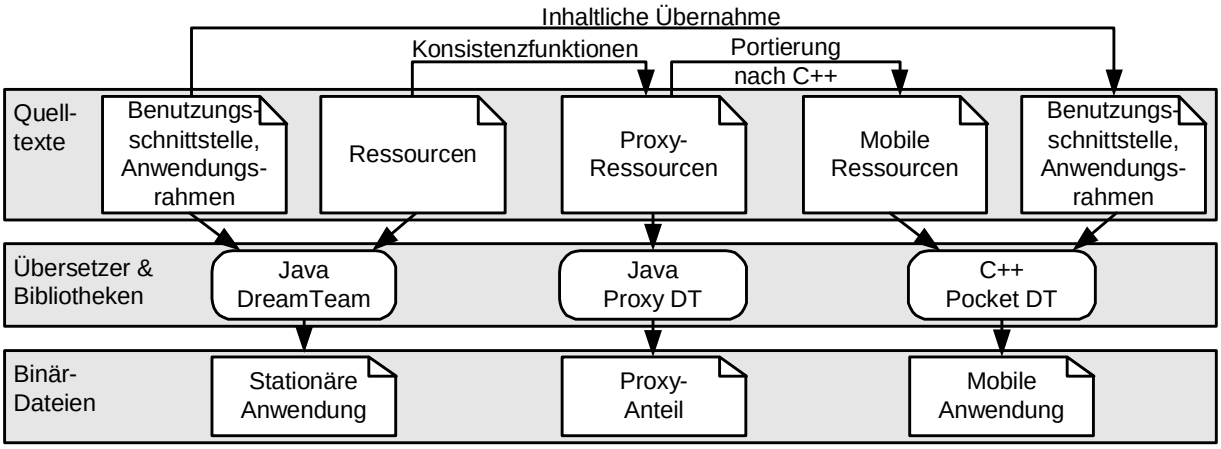

Abbildung 5: Schritte zur Entwicklung kollaborativer Anwendungen

Der erste Schritt stellt die Entwicklung einer stationären Anwendungsvariante dar. Hierzu werden die Quelltexte für die Benutzungsschnittstelle sowie die Ressourcen erzeugt. In einem zweiten Schritt werden die Proxy-Varianten der Ressourcen abgeleitet. Hierzu muss der Entwickler vor allem Code hinterlegen, um die optimistische Konsistenzerhaltung zu unterstützen.

Da der mobile Teil unter einer anderen Programmiersprache entwickelt wird, muss die Syntax der Ressourcen entsprechend angepasst werden. In der Regel muss nicht die gesamte Ressource portiert werden, sondern nur die internen Datenstrukturen sowie die Zugriffe zum Lesen des Status. Ort-zu-Ort-Aufrufe werden vom Laufzeitsystem automatisch an den Proxy weitergeleitet, so dass hierzu kein Portierungsaufwand anfällt. Bisher werden die Ressourcen-Quelltexte noch von Hand durch den Anwendungsentwickler abgeleitet. In Zukunft sollen diese Schritte jedoch weitgehend automatisiert ablaufen. Hierzu wird die Syntax des Quellcodes erweitert, um den Einsatz von Konvertierungsprogrammen zu ermöglichen.

Der größte Aufwand fällt derzeit durch die Generierung der Benutzungsschnittstelle des mobilen Geräts an, die nur inhaltlich von der stationären Anwendung übernommen werden kann. Eine direkte Umsetzung ist jedoch auch nicht sinnvoll, da spezielle Dialog-Werkzeuge existieren, die auf die Besonderheiten der mobile Endgeräte abgestimmt sind. Eine Lösung zu diesem Problem würden Ansätze auf der Basis von *User Interface Plasticity* (Calvary et al. 2001) bieten, bei denen ein Entwickler eine Benutzungsschnittstelle geräteunabhängig spezifiziert und die jeweiligen gerätespezifischen Quellen automatisch ableiten lässt. Solche Ansätze sind jedoch Gegenstand intensiver Forschung und haben derzeit noch zu keinen einsetzbaren Werkzeugen geführt.

3.4 Beispielanwendungen und Auswertungen

Um das Konzept zu überprüfen, wurden zwei Kernanwendungen von DreamTeam sowie zwei kooperative Anwendungen mit Hilfe von Pocket DreamTeam für ein mobiles Endgerät portiert. Insbesondere sollte überprüft werden, inwieweit Pocket DreamTeam dem Ansatz von Rapid Prototyping gerecht wird.

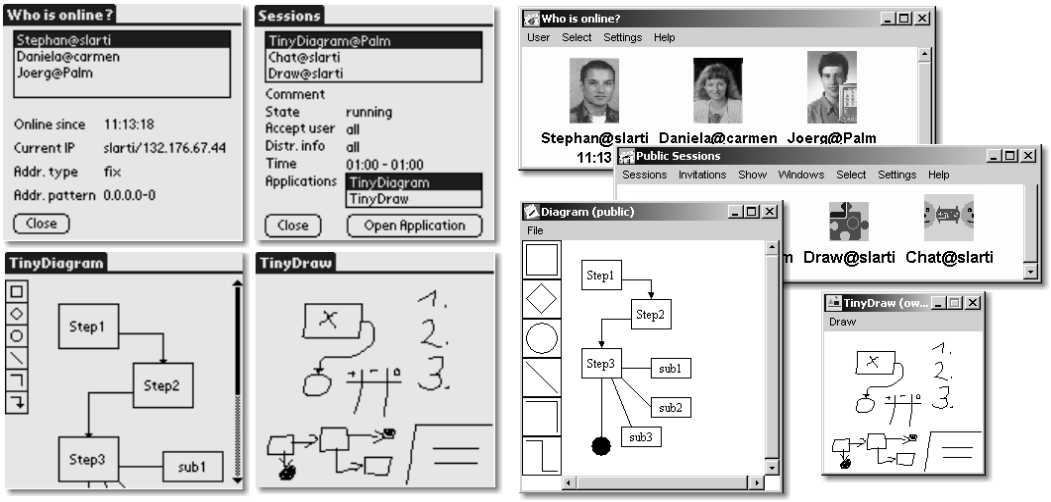

Abbildung 6: Pocket DreamTeam- (links) und DreamTeam-Fenster (rechts)

Für die Implementierung der Benutzungsschnittstellen wurden die Eigenschaften der jeweiligen Plattformen berücksichtigt. Während im Desktop-Bereich Ikonen, überlappende Fenster, große Menüs etc. üblich sind, wurde die mobile Variante auf die wesentlichen Funktionen reduziert. Auf Ikonen wurde verzichtet. Die Funktionen mehrerer Dialoge wurden teilweise in einem Dialog vereint, um dem Benutzer zeitraubende Umschaltungen zwischen Fenstern zu ersparen.

Es wurden die folgenden Anwendungen für die mobile Plattform portiert (Abbildung 6):

- Die *Online-Liste* ermöglicht es abzufragen, welche Gruppenmitglieder zur Zeit aktiv sind, also als potentielle Partner für Sitzungen in Frage kommen. Diese Liste stellt ein wesentliches Hilfsmittel zur Erzeugung von Gruppenbewusstsein dar und erlaubt die Einberufung spontaner Sitzungen.
- Im *Sitzungsmanagement* kann der Benutzer Sitzungen planen, ankündigen sowie zu laufenden Sitzungen beitreten.
- Mit einer kollaborativen *Diagramm*-Anwendung können Diagramme, z.B. Entity-Relationship-Diagramme, Klassendiagramme oder Flow-Charts, in der Gruppe entwickelt werden.
- Mit einem kollaborativen Freihand-Zeichenwerkzeug (*Draw*) können kleine Skizzen in der Gruppe erstellt werden, z.B. für eine Brainstorming-Sitzung.

Ziel dieser Implementierungen war, neben dem Test des Gesamtsystems, die Aufwände für die jeweiligen Anwendungen abzuschätzen. In der Summe hat ein erfahrener Entwickler für alle vier Anwendungen zusammen weniger als zwei Arbeitswochen benötigt. Es ist problematisch, die Aufwände für die Entwicklungen exakt zu quantifizieren, da diese sich selbst bei derselben Aufgabenstellung von Entwickler zu Entwickler stark unterscheiden. Wir weisen hier die Zahlen der Programmzeilen aus. Uns ist bewusst, dass diese nur eine Tendenz darstel-

len und nicht als absolutes Maß zu sehen sind. Neben den vier Anwendungen ist der Aufwand für die Entwicklung der Kernplattform ausgewiesen.

	Stationär	Proxy	Wiederverwendet	Mobil
Kernplattform	125000 Zeilen	110000 Zeilen	98 %	9500 Zeilen
Online-Liste	4400 Zeilen	4000 Zeilen	95 %	930 Zeilen
Sitzungsmanagement	7100 Zeilen	6900 Zeilen	93 %	2100 Zeilen
Diagram	6900 Zeilen	5200 Zeilen	92 %	600 Zeilen
Draw	930 Zeilen	520 Zeilen	90 %	410 Zeilen

Tabelle 1: Ergebnisse der Testimplementierungen

Die Spalten *Stationär*, *Proxy* und *Mobil* zeigen die Aufwände der Programmteile für die jeweilige Rechnerkategorie. Die Spalte *Wiederverwendet* zeigt den Anteil der Programmzeilen der Proxy-Anwendung, die aus der stationären Anwendung wiederverwendet wurden. Man sieht einen sehr hohen Anteil von mindestens 90%. Zusätzlich fällt auf, dass für die mobilen Anwendungen nur relativ wenig Programmzeilen anfallen. Dies liegt daran, dass hier im Wesentlichen die Benutzungsschnittstelle implementiert werden musste und komplexe Funktionen des funktionalen Kerns im Proxy ablaufen.

4 Zusammenfassung und Ausblick

Pocket DreamTeam stellt einen Startpunkt für weitere Forschungen auf dem Gebiet der mobilen Groupware dar. Ein Entwickler kann kostengünstig Prototypen für mobile kollaborative Anwendungen erstellen und den Endbenutzer schnell in die Weiterentwicklung einbeziehen. Ermöglicht wird dies durch einen hohen Anteil von wiederverwendbarem Quellcode und einem Laufzeitsystem, das anspruchsvolle Dienste im Hintergrund ausführt.

Weitere Forschungen gehen in zwei Richtungen. Auf der einen Seite soll die Entwicklung mobiler Groupware weiter vereinfacht werden, indem Konzepte aus dem Bereich der User Interface Plasticity einfließen. Hier ist sicherlich ein enormes Potential zu erwarten.

Zusätzlich sollen weitere Aspekte der Mobilität verfolgt werden. Neben den behandelten Aspekten entstehen Problemkreise durch die Berücksichtigung der räumlichen Position oder des aktuellen Benutzungskontextes. Zusätzlich spielt der komplexe Bereich der Sicherheit eine bedeutende Rolle für mobile Benutzer.

Literatur

Bey, C.; Freeman, E.; Hillerson, G.; Ostrem, J.; Rodriguez, R.; Wilson, G.; Dugger, M. (2001): *Palm OS Programmer's Companion*, Volume I, Palm Inc, Juli 2001

Calvary, J.; Coutaz, J.; Thevenin, D. (2001): A Unifying Reference Framework for the Development of Pastic User Interfaces, *8th IFIP Working Conference on Engineering for Human-Computer Interaction (EHCI'01)*, Toronto, 11.-13. Mai 2001, LNCS 2254, Springer, 173-192

Chabert, A.; Grossman, E.; Jackson, L.; Pietrowizc, S.; Seguin, C. (1998): Java Object-Sharing in Habanero, *Communications of the ACM*, Vol. 41, No. 6, Jun. 1998, 69-76

Coutaz, J. (1997): PAC-ing the Architecture of Your User Interface, In Proceedings of the DSV-IS'97, 4. *Eurographics Workshop on Design, Specification and Verification of Interactive Systems*, Springer Verlag, 1997, 15-32

Dewan, P.; Choudhary, R. (1992): A High-Level and Flexible Framework for Implementing Multiuser Interfaces, *ACM Transactions on Information Systems*, Vol. 10, No. 4, Okt. 1992, 345-380

Graham, N. (1996): *The Clock Language: Preliminary Reference Manual*, York University, Canada, 1996

Hill, R. D.; Brinck, T.; Patterson, J. F.; Rohall, S. L.; Wilner, W. T. (1993): Rendezvous Language, *Communications of the ACM*, Vol. 36, No. 1, Jan. 1993, 62-67

Joseph, A. D.; Tauber, J. A.; Kaashoek, M. F. (1997): Mobile Computing with the Rover Toolkit, *IEEE Transactions on Computers*, Vol. 46, No. 3, März 1997, 337-352

Kistler, J. J.; Satyanarayana, M. (1992): Disconnected Operation in the Coda File System, *ACM Transaction on Computer Systems*, Vol. 10, No. 1, Feb. 1992, 3-25

Kristoffersen, S.; Ljungberg, F. (1999): Designing Interaction Styles for a Mobile Use Context, *First International Symposion on Handheld and Ubiquitous Computing 1999 (HUC'99)*, Karlsruhe, 27.-29. Sept. 1999, LNCS 1707, Springer, 281-288

Munson, J. P.; Dewan, P. (1997): Sync: A Java Framework for Mobile Collaborative Applications, *Special issue on Executable Content in Java*, IEEE Computer, 1997, 59-66

Roth, J. (2000): DreamTeam - A Platform for Synchronous Collaborative Applications, AI & Society (2000), Vol. 14, No. 1, *Special Issue on Computer-Supported Cooperative Work*, Springer London, März 2000, 98-119

Roth, J. (2000b): *Entwicklungs- und Laufzeitunterstützung für synchrone Groupware*, dissertation.de, Berlin, 2000

Roth, J. (2002): Mobility Support for Replicated Real-time Applications (2001), *Innovative Internet Computing Systems (I2CS)*, Kühlungsborn, 20.-22. Juni 2002, LNCS 2346, Springer

Roth, J.; Unger, C. (2000): Developing synchronous collaborative applications with TeamComponents, in Dieng R. et al. (eds): *Fourth International Conference on the Design of Cooperative Systems*, Sophia Antipolis (Frankreich), 23.-26. Mai 2000, IOS Press, 353-368

Roth, J.; Unger, C. (2001): Using handheld devices in synchronous collaborative scenarios, *Personal and Ubiquitous Computing*, Vol. 5, Issue 4, Springer London, Dez. 2001, 243-252

Roseman, M.; Greenberg S. (1996): Building Real-Time Groupware with GroupKit, a Groupware Toolkit, *ACM Transactions on Computer-Human Interaction*, Vol. 3, No. 1, 1996, 66-106

Schuckmann, C.; Kirchner, L.; Schümmer, J.; Haake, J. M. (1996): Designing Object-Oriented Synchronous Groupware With COAST, In: *Proceedings of the ACM Conference on Computer Supported Cooperative Work*, ACM Press, Nov. 1996, 30-38

Zander, M. (2001): Anbindungen von Handhelds an das Sitzungsmanagement der Groupware-Umgebung DreamTeam, Diplomarbeit Praktischer Informatik II, Fernuniveristät Hagen, 2001