Universität-Gesamthochschule Paderborn
Fachbereich Wirtschaftswissenschaften
Wirtschaftsinformatik 2
Prof. Dr. L. Nastansky

Diplomarbeit

Architekturen und Anwendungskonzepte von Groupwar in der mobilen Kommunikation
–
Generische Entwicklung von Benutzungsschnittstelle und Prozeß-Modulen für ein intelligentes Display-Telephon

vorgelegt bei
Prof. Dr. L. Nastansky

von
Nico Dirks
Studiengang Wirtschaftsinformatik
Matrikelnummer 3925658
Arndtstraße 9, 33100 Paderborn

Danksagung

Besonderer Dank gebührt Herrn Prof. Dr. Nastansky, Herrn Prof. Dr. Herold und Herrn Dipl.-Wirt.-Ing. Riempp für ihre Unterstützung und die Betreuung der vorliegenden Arbeit.

Ebenfalls besonderer Dank gilt den Studenten Bernd Altmiks und Dirk Sievers für die hervorragende Arbeit im Team, ohne die diese Arbeit unmöglich gewesen wäre.

Mein Dank gilt auch der Firma Pavone Informationssysteme GmbH, die durch Rat und Tat zum Gelingen dieser Arbeit beitrug.

Eidesstattliche Erklärung

Ich erkläre hiermit an Eides statt, daß ich die vorliegende Diplomarbeit selbständig und nur unt Verwendung der angegebenen Hilfsmittel erstellt habe.

Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Paderborn, den 2. Mai 1996

Inhaltsverzeichnis

Anhang

Abbildungs- und Tabellenverzeichnis

Abkürzungsverzeichnis

ANSI - American National Standards Institute
AOCE - Apple Open Collaboration Environment
API - Application Programming Interface
ARQ - Automatic Retransmission Request
ASCII - American Standard Code for Information Interchange
ATM - Asynchronous Transfer Mode
AvALoN - Advanced Access to Lotus Notes
BMPT - Bundesministerium für Post und Telekommunikation (Deutschland)
CCITT - 'Comité Consultarif International Télégraphique et Téléphonique' mittlerweile in ITU (s.u.) umbenannt.
CGA - Color Graphics Adapter
CRC - Cyclic Redundancy Check
CSCW - Computer Supported Cooperative Work
DCE - Distributed Computing Environment
DCS - Digital Cellular System
D-AMPS - Digital Advanced Mobile Phone Service
DECT - Digital European Cordless Telecommunications
DES - Data Encryption Standard
DLL - Dynamic Link Library
ERMES - European Radio Message System
ETSI - European Telecommunications Standard Institute
FEC - Forward Error Correction
FTP - File Transfer Protocol
GDS - Group Decision Support Systems
GSM - Ehemals Normungsgremium der CCITT 'Groupe Speciale Mobile', jetzt 'Global System for Mobile Communication'
HTML - Hypertext Markup Language
ISDN - Integrated Services Digital Network
ISO - International Organisation for Standardization
ITU - International Telecommunications Union
JDC - Japanese Digital Communications Standard
LAN - Local Area Network
LCD - Liquid Crystal Display
MCB - Memory Control Block
OLE - Object Linking and Embedding
OMG - Object Management Group
OSF - Open Software Foundation
PC - Personal Computer

PCN	-	Personal Communication Network
PCMCIA	-	Personal Memory Card International Association
PCS	-	Personal Communication System
PDA	-	Personal Digital Assistant
PHS	-	Philips Home Services
POS	-	Phone Operating System
RAS	-	Remote Access Service
RISC	-	Reduced Instruction Set Computer
RNA	-	Remote Network Access
RSA	-	Rivest, Shamir, Adleman
SIM	-	Subscriber Identity Module
SMB	-	Static Memory Block
TETRA	-	Trans European Trunked Radio
EU	-	Europäische Union
WAN	-	Wide Area Network
WWW	-	World Wide Web
ZVEI	-	Zentralverband der Elektrotechnik und Elektronikindustrie (Deutschland)

1. Einleitung

1.1 Szenario

Die technische Entwicklungen der vergangenen Jahre, welche die kostengünstige Herstellung von kompakten Computern erlauben, und fortwährende Bestrebungen der Netzwerkintegration resultierten in vernetzungsfähigen Personal Computern. Bevölkerten diese anfangs die heimischen und beruflichen Arbeitsplätze noch isoliert voneinander, so ändert sich dieses Bild seit einigen Jahren derart, daß die 90'er Jahre als Dekade der Vernetzung in die DV-Geschichte eingehen werden. Parallel zu diesen technischen Entwicklungen wandelt sich auch die Nutzung dieser Techniken. Wurden anfangs gewohnte Arbeitsvorgänge übernommen und auf dem Rechner unverändert abgebildet, so wirken mittlerweile die technischen Möglichkeiten auf die Arbeitsvorgänge zurück und erlauben die Optimierung derselben. Der in den Alltag integrierte Personal Computer stellt somit nicht mehr nur eine Plattform zur Abbildung traditioneller Arbeitsvorgänge dar, sondern bietet die Möglichkeit die Arbeitsvorgänge zur effektiveren Aufgabenerfüllung umzugestalten.[1] Herkömmliche Netzwerkdienste, wie z.B. zentrale Dateihaltung auf Fileservern oder die gemeinsame Nutzung teurer Betriebsmittel, werden zunehmend durch neuere Dienste ergänzt, die gezielt zur Optimierung der Geschäftsprozesse eingesetzt werden können. Neue Möglichkeiten der rechnergestützten Kommunikation, welche von unterliegenden Hard- und Softwarekomponenten abstrahieren und die Verteilung bzw. Teilung multimedialer Informationen ermöglichen, wirken nachhaltig auf den Arbeitsablauf ein. Die Teamarbeit, befreit von den überkommenen Medienbrüchen und der Vielfalt der Dateiformate, erlebt durch die schnelle, sichere und verlustfreie Kommunikation einen gewaltigen Schub. Zusammengenommen erlauben die heutigen vernetzungsfähigen Personal Computer und die überarbeiteten Arbeitsvorgänge die räumliche Trennung der Teammitglieder, so daß z.B. internationale Teams ohne aufwendige Reisen an einer gemeinsamen Aufgabe arbeiten können.

Beschränkten sich diese neuen Möglichkeiten anfangs auf die Arbeit an stationären Rechnern, so kam mit der Einführung portabler Computer und diverser Techniken der Datenübertragung die Notwendigeit auf, auch mobile Anwender an diesen Entwicklungen teilhaben zu lassen. Techniken wie z.B. Remote Access Service (RAS, bzw. Remote Network Access, kurz RNA) bieten den mobilen Anwendern die Möglichkeit via Datenfernübertragungsgerät an Netzwerken teilzunehmen und so die Potentiale der Vernetzung zu nutzen.

Im Rahmen von Diplomarbeiten der Studenten Bernd Altmiks, Nico Dirks und Dirk Sievers an der Lehr- und Forschungseinheit Wirtschaftsinformatik 2 der Universität Gesamthochschule Paderborn wurde diesbezüglich eine Projektarbeit gefertigt. Im Mittelpunkt steht die Anbindung einfacher mobiler Frontends an Groupware-Backend Server. In Zusammenarbeit mit Philips Home Services Eindhoven (PHS) und Pavone Informationssysteme Paderborn entstand eine ge-

[1] siehe [Nastansky 1990b]

nerische Serverkomponente und eine auf dem Screen Phone P100 von Philips Home Services basierende exemplarische Clientkomponente, welche die Teilnahme mobiler Nutzer an Groupwareinfrastrukturen erlauben.

1.2 Umfeld der mobilen Kommunikation

War die mobile Kommunikation in der Vergangenheit größtenteils durch die Sprachkommunikation geprägt, so wandelt sich dieses Bild seit einigen Jahren. Neue Technologien und Lösungen drängen auf den Markt bzw. werden erprobt, mit denen mobile Anwender Datenkommunikation betreiben können. Die Beschränkung der mobilen Datenübertragungsmöglichkeiten auf einfache Pager-Dienste (wie z.B. Eurosignal und Cityruf) und einzelne geschlossene Benutzergruppen gehört dabei weitestgehend der Vergangenheit an. Zielgerichtet werden bestehende Datenübertragungsdienste ausgebaut und bestehende Kommunikationsdienste um Datenübertragungsmöglichkeiten erweitert.

Natürlich werden die sprachorientierten Übertragunsdienste durch diese Entwicklung nicht verdrängt oder gar überflüssig, sie werden vielmehr um eine neue Dimension angereichert, welche eine Vielzahl von Vorteilen und Möglichkeiten bietet[2]:

Kostengünstige Informationsübermittlung

Informationsinhalte können wesentlich schneller über eine Datenverbindung als im Gespräch übermittelt werden. Die kompaktere Struktur, die Möglichkeiten der Kompression, die vollautomatische Fehlerkorrektur und die geringere Redundanz (wie z.B. Begrüßung des Gesprächspartners, Abschweifen vom Thema, etc.) erlaubt einer Datenverbindung einen schnelleren und somit kostengünstigeren Informationstransfer. Bei dieser Betrachtung dürfen allerdings nur gleiche Kommunikationsverfahren miteinander verglichen werden, z.B. Daten- bzw. Sprachkommunikation im GSM-Netz[3]. Der Vergleich unterschiedlicher Kommunikations-verfahren, wie z.B. Modacom mit dem GSM-Netz, kann dieser Betrachtung durchaus widersprechen.

Vermeidung von Mißverständnissen und Übertragungsfehlern

Die in der Datenübertragung integrierten Verfahren der Fehlererkennung bzw. Fehlerkorrektur erlauben einen fehlerfreien Transfer von Informationen. Mißverständnisse, die in der Sprachkommunikation auftreten können und ggf. von den Gesprächsteilnehmern nicht erkannt werden, sind weitestgehend ausgeschlossen. Menschliche Übertragungsfehler bei der Eingabe der ausgetauschten Informationen in ein Rechnersystem werden durch die direkte Kommunikation des Rechnersystems mit der sendenden Gegenstelle vermieden.

Vermeidung von Medienbrüchen

Fehleranfällige und mit Informationsverlust behaftete Medienbrüche werden durch die Datenkommunikation vermieden. Daten erfahren keinerlei Veränderung, Interpretation oder

---

[2] vgl. [Jowett 1992 S. 10ff]

[3] GSM - ehemals Normungsgremium der CCITT (Comité Consultarif International Télégraphique et Téléphonique) 'Groupe Speciale Mobile', jetzt 'Global System for Mobile Communication'

Konvertierung durch die Übermittlung. Jegliche Veränderung der Daten für die Kommunikation, z.B. durch Kompression, wird vor dem Benutzer verborgen und auf der Empfängerseite automatisch rückgängig gemacht.

Bessere Wettbewerbsfähigkeit und Imagegewinn

Durch die Möglichkeit auch im mobilen Bereich jederzeit Daten abzugleichen und Informationen einzuholen, kann ein Informationsvorsprung gegenüber der Konkurrenz aufgebaut bzw. das Image bei Geschäftspartnern gehoben werden.

Dennoch mangelt es der Datenkommunikation an Flexibilität. Treten Ausnahmebedingungen, Notfälle oder ungeplante Informationsinhalte auf, bietet die Sprachkommunikation die einzige sinnvolle Möglichkeit des Austauschs von Informationen.

In diesen Punkten sind schon die wesentlichen Potentiale der mobilen Kommunikation angesprochen, sie ist nicht einseitig durch die technischen Möglichkeiten, sondern im wesentlichen durch die Nutzungsmöglichkeiten geprägt. Im hier verstandenen Sinne schließt die mobile Kommunikation somit nahtlos an die Gebiete 'Mobile Computing' ("Oberbegriff für Anwendungen tragbarer Computer innerhalb eines größeren Datenverbundes."[4]), 'Mobile Business' ("Oberbegriff für die Anwendungen tragbarer Informationstechnik, nicht nur als Datenplattform, sondern zur Unterstützung originärer Geschäftsaufgaben (Flexibilisierung des Arbeitsortes)"[5]) und 'Mobile Office' ("Die Vorstellung, daß alle zur Bewältigung der Büroaufgaben erforderlichen Hilfsmittel Informationen etc. nicht mehr ortsgebunden sind, sondern dem Mitarbeiter auch unterwegs, beim Kunden oder zu Hause zur Verfügung stehen. Das Büro ist somit stets dort, wo Büroaufgaben erledigt werden können“[6]) an und bildet unter anderem den dafür notwendigen Unterbau. Jede diese Thematiken besitzt neben eigenen spezifischen Schwerpunkten viele Schnittstellen und Schnittmengen zu den jeweils anderen, eine genaue Abgrenzung ist oftmals nicht möglich.[7]

Aus dieser Konstellation heraus ergibt sich die Sichtweise des Verfassers dieser Arbeit. Der Autor betrachtet die mobile Kommunikation als ein Standbein für die darauf aufbauenden Thematiken 'Mobile Computing', 'Mobile Business' und 'Mobile Office'.

Zu Gunsten der Ausrichtung dieser Diplomarbeit und des dahinter stehenden Praxisprojektes legt er dabei den Schwerpunkt zu Lasten der Sprach- auf die Datenkommunikation, welche er insbesondere unter dem Aspekt der räumlichen Trennung betrachtet. Funk- oder Infrarotkopplungen in lokalen Netzwerken werden nur am Rande betrachtet.

---

[4] [Niemeier, Schäfer, Engstler, Kroll 1994, S. 175]

[5] [Niemeier, Schäfer, Engstler, Kroll 1994, S. 175]

[6] [Niemeier, Schäfer, Engstler, Kroll 1994, S. 175]

[7] siehe z.B. folgende Graphik

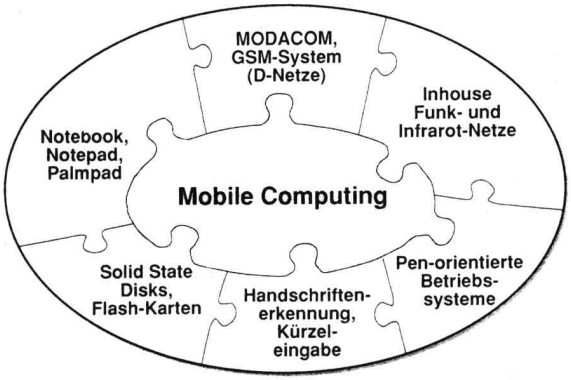

Abbildung 1: Vielfalt des Gebietes Mobile Computing[8]

Herrschen auch bei der Definition bzw. Abgrenzung und dem Zusammenspiel dieser Thematiken noch Uneinigkeit und werden die Begriffe oftmals synonym verwand, so herrscht dennoch Einigkeit ob der Notwendigkeit dieser Entwicklungen.

"Zunehmender Wettbewerbsdruck erfordert vom Außendienstmitarbeiter bis zum Manager ständig ad-hoc-Entscheidungen und hohe Auskunftsbereitschaft. Fach und Entscheidungskompetenz 'vor Ort' verlangen ein 'Mobile Business'"[9]. Dabei reicht die Spannbreite dieses Feldes von Sprachkommunikation via Mobilfunk bis zu funkgestützten Büronetzwerken und von Satellitenfunk bis zu tragbaren Personal Digital Assistants (PDA's).

Als typische Einsatzbereiche für diese Entwicklungen können dabei z.B. die folgenden gelten:[10]

- Mobiler Manager Arbeitsplatz
- Verkaufsaußendienst
- Produktkonfiguration und technischer Kundendienst
- Lagerlogistik und Materialfluß
- Kundenmanagement
- Transportdienstleistungen und Speditionslogistik

[8] [Niemeier, Schäfer 1993, S. 73]

[9] [Niemeier, Schäfer 1993, S. 73]

[10] [Niemeier, Schäfer, Engstler, Kroll 1994, S. 129ff]

1.3 Schnittstellen zwischen mobiler Kommunikation und Groupware

Informationen als Ergänzung der klassischen Produktionsfaktoren verlieren oftmals mit der Zeit an Wert und bestimmen nicht selten die Position am Markt. Schnelle Verfügbarkeit, Sammlung, Auswertung und die Bereitstellung von Informationen ist im Wettbewerb zur Pflicht geworden. Durch stockenden Informationsfluß bedingte Kapitalbindung, mangelnde Reagibilität und Flexibilität stellen sich immer mehr als wettbewerbshinderlich heraus. Erklärtes Ziel kann es daher nur sein, alle, auch mobile, Informationsnutzer und -erzeuger in eine Infrastruktur zu integrieren und den schnellen Datenaustausch zu gewährleisten[11].

Groupware[12] , als Hilfsmittel der Organisation verteilter, ggf. multimedialer Datenbestände, stellt ein ideales Medium zur Integration mobiler Nutzer in die Unternehmensinformationsstruktur dar. Den mobilen Mitarbeitern wird es ermöglicht, neu erfaßte Daten direkt in den Unternehmensdatenbestand einzuspielen und an diesem zu partizipieren. Dieser Abgleich kann durch eine Vielzahl von Transportmedien geschehen. Diverse leitungsgebundene Übertragungswege und diverse funkgestützte Übertragungswege können mittlerweile genutzt werden. Bei den funkgestützten Übertragungswegen können z.B. Handhelds (kurz Handys) nach dem GSM-Standard, Funkdienste wie Modacom oder Bündelfunksysteme genutzt werden. Jede dieser Alternativen erlaubt den Datenaustausch mit anderen Teilnehmern und Unternehmen. Von der Seite der Endgeräte sind dabei kaum Grenzen gesetzt, viele mobile Datenendgeräte lassen sich entweder direkt für die Datenkommunikation nutzen, oder dafür umrüsten. Lösungen auf Basis genormter Erweiterungssteckkarten[13] sind oftmals sogar für mehrere Gerätetypen nutzbar.

Dabei ist zu beachten, daß die zur Nutzung von Groupware notwendige Geschäftsprozeßumgestaltung (Business Process Re-Engineering) frühzeitig auf die Belange der mobilen Kommunikation ausgedehnt wird. Lösungen im Bereich des 'Mobile Business' müssen ebenso wie jeder andere Geschäftsprozeß in das Gesamtmodell integriert werden. Ohne diese institutionalisierte Veränderung arten Tätigkeiten in diesem Bereich in Provisorien aus[14].

11 siehe [Preßmar 1994]

12 siehe Kapitel 2.2

13 nach Standard der ‘Personal Memory Card International Association’, kurz PCMCIA

14 siehe [Nastansky 1994, S. 296f] u. [Niemeier, Schäfer, Engstler, Kroll 1994, S. 27ff]

1.4 Aufbau dieser Arbeit

Die vorliegende Arbeit ist vor dem Hintergrund der praktischen Arbeit 'MobileNotes' entstanden. Sie gliedert MobileNotes einerseits in die Thematik 'Mobile Kommunikation' ein und dokumentiert andererseits interessante funktionale Aspekte und Implementationsdetails. Diese Teilung spiegelt sich in dem an Grundlagen orientierten zweiten Kapitel und den an MobileNotes orientierten dritten und vierten Kapiteln wider. Doch nun zu den Kapiteln im einzelnen:

Das zweite Kapitel führt in die mobile Kommunikation und das Umfeld von Groupware ein. Zuerst werden Entwicklungen im Bereich der mobilen Kommunikation und deren Möglichkeiten geschildert und die derzeitige auf dem Markt erhältlichen Alternativen aufgezeigt. Dabei wird der Schwerpunkt auf die durch die Faktoren Liberalisierung und Wiedervereinigung geprägten Verhältnisse in der Bundesrepublik Deutschland gesetzt. Die Möglichkeiten der mobilen Sprachkommunikation werden allerdings vor dem Hintergrund dieser Arbeit und zu Gunsten der mobilen Datenkommunikation nur auszugsweise beschrieben. Des weiteren wird der Begriff Groupware eingeführt, dessen Umfeld aufgezeigt und einige exemplarische Produkte vorgestell

Das dritte Kapitel stellt die Grundlagen und Ziele des MobileNotes-Projektes dar, stellt den Prototypen 'AvALoN' vor und dokumentiert grundlegende Designentscheidungen der Lösung. Dazu gehört ebenso der Aufbau der MobileNotes-Serverkomponente als auch die Wahl der Clientplattform und der Aufbau der Clientkomponente. Potentiale dieses Projektes werden aufgezeigt und Anforderungen an eine mobile Groupwarelösung definiert.

Im vierten Kapitel werden die beiden Komponenten genauer vorgestellt, ihr Aufbau untersucht, auf Implementationsdetails eingegangen und die technischen Rahmenbedingungen aufgezeigt. Konkret werden hier die vom Autor angefertigten Client- und Servermodule vorgestellt und ihr Innenleben erklärt.

Das abschließende fünfte Kapitel faßt die aus den vorangegangenen Kapiteln gewonnenen Erkenntnisse zusammen und stellt in komprimierter Form die Ansichten des Autors bezüglich mobiler Kommunikation und Groupware dar. Es wird auf derzeitige Problempunkte ebenso hingewiesen wie auf zukünftige Chancen.

2. Grundlagen und Klärung der Begrifflichkeiten

2.1 Mobile Kommunikation

Die wachsende Bedeutung der mobilen Kommunikation läßt sich an den Entwicklungen auf den Telekommunikationsmärkten und den technischen Entwicklungen festmachen. Weltweit steigen die Bestrebungen mobile Kommunikationsinfrastrukturen aufzubauen, Telekommunikationsmärkte zu öffnen und internationale Standards durchzusetzen. Im Falle der Bundesrepublik Deutschland beeinflußt die Wiedervereinigung als weiterer Faktor die Entwicklung im gesamten Bereich der Kommunikation.

Spiegelten die ehemalige Deutsche Demokratische Republik (DDR) und die Bundesrepublik Deutschland in der Vergangenheit die unterschiedlichen Telekommunikationsinfrastrukturen in Ost und West wider, so entstand mit der Wiedervereinigung die Notwendigkeit der Integration zweier völlig verschiedener Entwicklungsstände. Die Finanzkraft und technischen Möglichkeiten der Bundesrepublik Deutschland erlauben allerdings eine Aufbauleistung, welche die Leistungsfähigkeit der osteuropäischen Länder um ein Vielfaches übersteigt. Das Programm 'Telekom 2000' der Deutschen Telekom (im folgenden kurz Telekom) wird mit Investitionen in Höhe von 60 Mrd. DM in dem Zeitraum 1991-1997 die Anzahl der ostdeutschen Telephonanschlüsse um 7,2 Millionen auf 9 Millionen erhöhen und die Infrastruktur für die mobile Kommunikation installieren.

Die Planungen für das Progamm begannen schon vor der Wiedervereinigung der beiden deutschen Staaten im Jahre 1990 und umfaßten die zur Versorgung der fünf neuen Bundesländer notwendigen, kurz-, mittel- und langfristigen Maßnahmen. Darunter z.B. die Erhöhung der Telephonanschlußversorgungsrate, die Installation eines Paketvermittelnden Netzwerkes, die Bereitstellung analoger und digitaler zellularer Mobilfunknetzwerke und einige weitere Elemen

Die Erweiterung des leitungsgebundenen Telephonnetzes wurde top-down vorangetrieben. Dem Ausbau der Ost-West Verbindungen und eines überregionalen Netzgrundgerüstes folgte der Ausbau und die Digitalisierung der regionalen Infrastruktur. Dann wurde oftmals mit Hilfe von Vertragsunternehmen der Ausbau lokaler Infrastrukturen vorangetrieben. Priorität wurde bei diesen Maßnahmen auf die schnelle Bereitstellung bzw. Erweiterung von geschäftlich genutzten Anschlüssen gelegt, damit die ostdeutsche Wirtschaft nicht länger als unbedingt notwendig von der wirtschaftlich überlebenswichtigen Telekommunikation abgeschnitten blieb. Gegebenenfalls wurden dabei temporär Funknetze verwendet, mit denen der Kontakt zum Festnetz hergestellt wurde, um schnelle Übergangslösungen auch in abgelegenen Gebieten anbieten zu können. Diese Funkanbindung wurde als Festnetzverbindung ohne zusätzliche Kosten berechnet.

Ziel der Erweiterung des leitungsgebundenen Telephonnetzes ist es, die Telephonanschlußversorgungsrate von 11% (Anschlüsse je Einwohner) im Jahre 1989 auf Westdeutsches Niveau zu heben, welches 1989 bei ca. 50% lag. Mit Abschluß dieses Programmes werden die fünf neuen

Bundesländer über ein vollständig digitalisiertes und moderneres leitungsgebundenes Telephonnetz als der Westen verfügen.

Während bei dem leitungsgebundenen Telephonnetz die bestehende Infrastruktur weiter genutzt und schrittweise ersetzt bzw. erweitert werden konnte, stand die Telekom bei der mobilen Kommunikation in Ostdeutschland vor dem Nichts. In einigen wenigen Fällen staatlich bzw. militärisch genutzt, stand die gänzlich unzureichende und technisch veraltete Mobilkommunikationsinfrastruktur der DDR den Privatpersonen nicht zur Verfügung. Daher begann die Deutsche Telekom schon 1990, das bestehende analoge C-Netz (heute C-Tel) in den Bereichen der ehemaligen innerdeutschen Grenze zu erweitern und in den ostdeutschen Ballungsräumen einzurichten.

Schon 1992 deckte C-Tel 90% der Bevölkerung und 80% der Fläche der fünf neuen Bundesländer ab und bot eine gute Grundlage für die digitalen GSM-Netze, welche von den Ballungsräumen ausgehend, über ganz Ostdeutschland ausgebreitet werden. Oftmals ersetzte das Mobiltelephon das nicht vorhandene Telephon und war somit im eigentlichen Sinne nicht mobil. Diese Ersatzfunktion, auf die anfangs viele Geschäfts- und Privatleute angewiesen waren, überlastete zeitweise das noch im Aufbau befindliche C-Netz. Mit der zunehmenden Verfügbarkeit leitungsgebundener Telephone verlor dieses Problem jedoch an Relevanz. Hoffnungen der Netzbetreiber, daß die Nutzer des C-Netzes auf die qualitativ höherwertigen und wesentlich größere Kapazitäten bietenden GSM-Netze umsteigen, haben sich nur teilweise erfüllt. Die Erwartungen der GSM-Netzbetreiber (D1, D2 und durch die späte Einführung nur bedingt E1), im Osten Deutschlands wegen der unzureichenden Festnetzinfrastruktur als dauerhafte Alternative eingesetzt zu werden, haben die Betreiber zur schnellen flächendeckenden Inbetriebnahme veranlaßt, so daß die GSM-Versorgung im Osten mittlerweile durchaus mit der des Westens vergleichbar[15] ist.

Der zweite beeinflussende Faktor, die Liberalisierung des Telekommunikationsmarktes, fußt auf Bestimmungen der Europäischen Union aus dem Jahre 1987 (im Green Paper). Diese Bestimmungen geben den EU-Mitgliedsstaaten den Weg zur Privatisierung der Telekommunikationsmärkte und die Grenzen der politischen Gestaltungsfreiheit in dieser Angelegenheit vor. Inhaltlich umfassen diese Bestimmungen die Liberalisierung des Endgerätemarktes, die Entfernung aller Wettbewerbshindernisse im Bereich der Zusatzleistungen, die Etablierung EU-weiter Standards und die Installation von Wettbewerbsaufsichtsbehörden. Die Netzwerkinfrastruktur und die Sprachkommunikation bleiben, ebenso wie die Gestaltung der Besitzverhältnisse an den Telekommunikationsunternehmen, in der Verantwortung der Mitgliedsstaaten.

In der Bundesrepublik Deutschland führten diese Vorschriften zur 'Postreform' im Jahre 1989 und somit zur Aufteilung der Deutschen Bundespost (DBP) in drei eigenständige Unternehmen (Post, Postbank und Telekom), welche schrittweise privatisiert werden. Bis auf die Ausnahme der mobilen Kommunikation ist die Deutsche Telekom in den Bereichen Netzinfrastruktur und

[15] siehe [Berlage, Schnöring 1995]

Sprachtelephonie noch Monopolist. Alle anderen Dienste und die mobilen Kommunikationsdienste können nach Absolvierung eines Lizenzverfahrens auch privat errichtet werden. In vielen Fällen kann sogar von der Lizensierung abgesehen werden. Erste Früchte trug diese Politik bei der Genehmigung eines privaten Mobilfunknetzes auf Basis des GSM-Standards, des D2-Netzes der Mannesmann-Mobilfunk GmbH. Weitere Genehmigungen folgten. Doch alle diese Entwicklungen sind nur Zwischenschritte zur vollständigen Marktöffnung im Januar 1998, ab diesem Zeitpunkt steht der Telekommunikationsmarkt allen Interessenten offen. Prognosen über die Entwicklung der Telekommunikation in der Bundesrepublik Deutschland lassen ein riesiges Marktvolumen erhoffen, welches aber, wegen der hohen notwendigen Anfangsinvestitionen, nur Raum für wenige Marktteilnehmer lassen wird.[16]

[16] siehe [Toker 1995], [Berlage, Schnöring 1995] u. [Esser-Wellie 1996]

2.1.1 Entwicklungsgeschichte

Die Entwicklungsgeschichte der mobilen Kommunikation war lange Zeit von dem Wunsch nach ortsunabhängiger Sprachkommunikation geprägt und begann 1946 in den USA. Militärische Entwicklungen wurden erstmals für Polizei- und Taxifunk genutzt.[17] Die schwedische Telecom (bzw. ab 1993 Telia) kommerzialisierte 1956 das erste europäische System, das 'System A'. Eine einzelne Funkstation versorgte ein Gebiet mit 30km Radius um Stockholm. Die Endgeräte waren schwer, unhandlich und verbrauchten viel Energie. Letztendlich stagnierte das System mit wenigen Benutzern, führte aber in Schweden und in anderen Ländern zur Weiterentwicklung dieser Technologie.[18] In der Bundesrepublik Deutschland begann die Entwicklung mit der Einführung des A-Netzes 1958 und setzte sich 1972 mit der Einführung des B-Netzes fort. Dieses Netz bzw. dessen Nachfolger B/2, welches ebenfalls in Österreich, Luxemburg und den Niederlanden eingeführt wurde und bis ca. 1994 in Betrieb war, ermöglichte schon den vollautomatischen Verbindungsaufbau in beiden Richtungen. Allerdings mußte der Anrufer den Aufenthaltsbereich des Adressaten kennen und Sendestationen konnten nicht während einer Verbindung gewechselt werden. Diese Mißstände wurden im Jahr 1986 durch das analoge C-Netz behoben. Sendestationen ermitteln den Aufenthaltsbereich der C-Netz-Nutzer (sogenanntes Roaming) und befreien den Anrufer somit von der Pflicht, den Aufenthaltsbereich anzugeben. Zusätzlich können die Sendestationen während einer Verbindung gewechselt werden (sogenanntes Handover). Diese wichtigen Merkmale eines zellularen Mobilfunknetzes wurden beim C-Netz um Zusatzdienste angereichert. Neben Rufumleitungen und Anschlußsperren stehen dem Nutzer auch Datendienste, wie z.B. Mobilfax, Mobilbox (Anbindung an Telebox 400), Datex-J und Datex-P, zur Verfügung. Die analoge Datenübertragung erlaubt es mit handelsüblichen Modems und einem Adapter Verbindungen mit 2400 bps, im Faxdienst der Gruppe 3 sogar mit 9600 bps aufzubauen.[19]

Die Entwicklung im Bereich des Funktelephons ging weiter. Das C-Netz erlebte noch einen Höhepunkt nach der Wiedervereinigung der beiden deutschen Staaten. Als Ersatz für die fehlende Telephoninfrastruktur wurde es massiv ausgebaut, verliert aber z.Zt. durch Netze nach dem international genormten GSM-Standard an Bedeutung. Die neuen Netze bieten durch die damit verbundenen größeren Märkte und der daraus resultierenden größeren Konkurrenz die attraktiveren Endgerätepreise. Die auf wenige nationale Märkte beschränkte C-Netz-Technologie kann diesen Entwicklungen nichts entgegensetzen.[20]

Diese neuen zellularen Netze überzeugen durch digitale Übertragungstechnik, Kompatibilität zu den ISDN-Standards (Integrated Services Digital Network) und internationale Verfügbarkeit.

---

[17] siehe [Lobensommer 1994, S. 99]

[18] siehe [Hulten, Mölleryd 1995]

[19] siehe [Lobensommer 1994, S. 121] u. [Decker, Walke 1993]

[20] siehe [Lobensommer 1994, S. 123]

Roamingabkommen mit 118 Netzanbietern in 60 Ländern[21] erlauben die grenzüberschreitende Nutzung ein und desselben Endgeräts in vielen Staaten dieser Erde. Positiver Nebeneffekt der Digitalisierung ist die vereinfachte Datenkommunikation. Die Sprachkommunikation erfolgt mit derselben Datenübertragungstechnik, die auch den Transfer von Binärdaten erlaubt. Komplexe fehlerträchtige Analog-Digital-Wandler werden somit nicht mehr benötigt.

Ein interessantes Detail, das seit der Entwicklung des C-Netzes in den Alltag des mobilen Nutzers Einzug hält, soll hier nicht unerwähnt bleiben. Alle modernen Funktelephone, für Modacom (s.u.) ist es angedacht, werden nicht mehr statisch mit den Benutzerdaten kodiert, sondern über eine Chipkarte (Subscriber Identity Module, kurz SIM) mit den Benutzerdaten versorgt. Die Telephonnummer, der Dienstanbieter, die Kostenabrechnung und die freigeschalteten Dienste sind somit nicht mehr endgerätespezifisch, sondern können mit der SIM auf beliebige Endgeräte übertragen werden. Dem Neukauf, der kurzzeitigen Benutzung eines fremden Handhelds und der Nutzung öffentlicher Telephonzellen, die diese Karten ebenfalls akzeptieren, steht somit nichts mehr entgegen. Die umständliche und langwierige Neukodierung der Geräte beim Händler oder gar Hersteller entfällt. Mittlerweile auch schon in der zweiten Generation gefertigt, die erste basierte auf Magnetkarten, bieten die SIM's einen sehr hohen Grad an Fälschungssicherheit. Zusammengenommen bedeutet die Einführung der SIM einen weiteren Schritt in Richtung Mobilität.[22]

Neben diesen primär zur Sprachkommunikation gedachten Netzen entstand schon im Jahre 1974 der erste reine Datendienst, das Eurosignal. Dieser Funkruf-Dienst (sogenannter Paging-Dienst) erlaubt die Simplex-Datenübertragung und war, wie der Name schon andeutet, für die europaweite Nutzung gedacht. Mit zigarettenschachtelgroßen Endgeräten kann der Benutzer Tonsignale empfangen. Die Einführung neuerer Entwicklungen, wie z.B. der Cityruf 1989 in der Bundesrepublik Deutschland, weitete das Spektrum auf kurze numerische und alphanumerische Datensätze aus. Beide Systeme haben jedoch Schwächen, die eine Weiterentwicklung notwendig machten. Das für den europäischen Markt geplante Eurosignal wird dem internationalen Anspruch nicht gerecht[23], der Cityruf erlaubt nur geringes, kostenpflichtiges Roaming.

Internationale Standardisierungsbemühungen, die in dem ERMES-Standard (European Radio Message System) resultierten, haben seit dem Jahre 1993 zur Internationalisierung der Paging-Dienste beigetragen. 28 Netzbetreiber in 18 Ländern schlossen sich zusammen, nutzen identische Übertragungsfrequenzen und gewährleisten internationales Roaming. Die geringe Bandbreite, die Simplex-Datenübertragung, die fehlende Rückmeldung von Verbindungsfehlschlägen beim Sender und die Beschränkung auf kurze Nachrichten qualifizieren dieses Netz allerdings nicht zur Nu-

[21] siehe [Huber 1995]

[22] siehe [Lobensommer 1994, S. 134 f] u. [Becker 1991b]

[23] derzeit nutzen nur Frankreich, die Schweiz und die Bundesrepublik Deutschland das Eurosignal

zung im Bereich des 'Mobile Computing'. Anwendungen, wie z.B. Alarmierungen und Telemetrie stellen noch immer die Hauptanwendungsfelder dar.[24]

Weiterhin etablieren sich seit 1993 zwei reine Datenfunkdienste, der Modacom-Dienst der Telekom und das GfD-Netz[25] der Gesellschaft für Datenfunk. Technisch basieren diese beiden Systeme auf ähnlichen Grundlagen, unterscheiden sich allerdings in den verwendeten Trägerfrequenzen und der Kompatibilität zu anderen europäischen Datenfunkdiensten. Die Endgeräte sind somit nicht beliebig austauschbar. Sendeanlagen bedecken großflächig das gesamte Bundesgebiet. Die niedrigen Trägerfrequenzen erlauben einzelnen Anlagen die Versorgung großer Einzugsgebiete. Im Falle Modacom erlaubten diese 1995 95% der Bevölkerung auf 90% der Fläche die Nutzung dieses Dienstes. Eigenschaften dieser Netze, wie Roaming (mit der Einführung von Modacom in der Schweiz auch grenzüberschreitendes Roaming) und Handover erlauben uneingeschränkte Mobilität der Benutzer. Funktional bieten diese Netze einen Übergang mit 9600 bps in das Datex-P Netz der Telekom. Gleich schnelle Verbindungen zwischen zwei Endgeräten (Mobilen Terminals) werden zusätzlich unterstützt. Allerdings liegt die Nettotransferrate in beiden Fällen weit unter 9600 bps, da mehrere Endgeräte die gleiche Frequenz verwenden können. Die Nutzung geschlossener Benutzergruppen, Broadcasting, Paketspeicherung im Datex-P Netz (z.B. bei ausgeschaltetem Endgerät Speicherung für maximal 24 Stunden) und die Bindung der Kosten an die Übertragungsmenge machen diese Netze für eine Reihe von Anwendungen attraktiv. Während bei anderen Netzen, z.B. GSM-Netzen, zeitabhängige Verbindungskosten anfallen, die allein für den Verbindungsaufbau sehr schnell in den D-Mark-Bereich wachsen, wird bei den zwei Datenfunkdiensten die Übertragungsmenge in Einheiten von wenigen Bytes abgerechnet (32 Byte kosteten 1995 im Modacom-Dienst 0,008 DM). Hieraus erwächst der finanzielle Vorteil dieser Datenfunkdienste bei der Übertragung kleiner Datenmengen, die ggf. über große Zeiträume verteilt anfallen. Fallen große Datenmengen an, die ggf. auch noch zu einem Zeitpunkt übergeben werden müssen, ist es allerdings günstiger eine Infrastruktur mit Zeitabrechnung zu nutzen.[26]

Dieser Abrechnungsmodus schränkt die Nutzungsmöglichkeiten in der mobilen Kommunikation ein. Der Transfer multimedialer Daten in der Größenordnung von Megabytes wird schnell zur finanziellen Belastungsprobe. Die Nutzung zur Übertragung von Daten eines Groupwareproduktes erscheint somit recht unwahrscheinlich. Typische Anwendungsfelder dieser beiden Datenfunkdienste liegen in der Telemetrie, der Meßdatenübermittlung und der Flottenverwaltur

Auch der klassische Betriebsfunk erfuhr in der Vergangenheit einige Veränderungen. In den siebziger Jahren eingeführt, wurden den Nutzerkreisen, wie z.B. Behörden und der Großindustrie, Frequenzen zugeteilt, die in einem regional begrenzten Gebiet um eine Feststation herum die Kommunikation ermöglichen. Verbindungen können semiduplex, d.h. explizites Umschalten zwischen Sende- und Empfangsmodus am Endgerät, zwischen Nutzern des gleichen Netzes oder

---

[24] siehe [Lobensommer 1994, S. 171ff] u. [Decker, Walke 1993]

[25] siehe [Reder 1996]

[26] siehe [Zimmermann, Bayard 1993], [Kaiser 1994] u. [Feichtinger, Hofreiter 1995]

über spezielle Einrichtungen auch in andere Netze, z.B. Festnetz, Mobilfunknetz etc. aufgebaut werden. Innerhalb des Netzes wird sogar der Rundruf unterstützt. Seit 1990 wird der Betriebsfunk durch neue Bündelfunksysteme verdrängt. In Ballungsgebieten sind jeweils zwei Lizenzen zum Betrieb derartiger Netze durch den Bundesminister für Post und Telekommunikation (kurz BMPT) vergeben worden, eine jeweils an die Telekom mit ihrem Bündelfunksystem Chekker[27]. Die andere wurde jeweils ausgeschrieben. Diese neuartigen Bündelfunksysteme arbeiten ebenfalls analog und semiduplex, erlauben aber größere Netze. Typischerweise wird jeweils ein Stadtgebiet von einem Netz abgedeckt. Jedes dieser Netze kann dabei aus mehreren Zellen (und somit Feststationen) bestehen, zwischen denen der Benutzer beliebig wechseln kann (Netzweites Roaming). Die exklusive Zuordnung von Frequenzen zu einzelnen Benutzergruppen entfällt, stattdessen wird ein ganzes Bündel von Frequenzen genutzt, aus dem dynamisch die Anforderungen nach Übertragungsfrequenzen befriedigt werden können. Geschlossene Benutzergruppen, wie vormals im Betriebsfunk, werden hierbei durch spezielle technische Maßnahmen realisiert. Zu beachten ist, daß kein Handover unterstützt wird. Zu Gunsten einer einfachen und kostengünstigeren Implementation der Bündelfunksysteme wurde bewußt auf diese Fähigkeit verzichtet. Eine Reihe von Zusatzdiensten und die niedrigen Preise lassen die neuen Dienste dabei durchaus mit den Mobiltelephonen konkurrieren. Chekker bietet z.B. Rundruf, Anruferidentifizierung, Anrufumleitung, Kurzdatenübertragung (wenige alphanumerische Zeichen) und diverse Übergänge in andere Netze an.

Doch die Entwicklung geht weiter, seit 1994 ist das European Telecommunications Standard Institute (ETSI) bestrebt die verschiedenen nationalen Bündelfunksysteme auf den gemeinsamen TETRA-Standard zu heben (TETRA - Trans European Trunked Radio). Die Umstellung auf digitale Übertragungstechnik, neue Datendienste, netzüberschreitendes Roaming und Verschlüsselungsmöglichkeiten sollen dabei die Attraktivität dieser Systeme anheben. Die hohen Endgerätepreise der Vergangenheit können dabei durch internationale Standards und den dadurch entstehenden großen Absatzmarkt verhindert werden. Wert wurde bei diesem gemeinsamen Standard auch auf die Datenübertragung gelegt. Während bei den aktuellen Bündelfunksystemen, wie z.B. bei Chekker, nur 2 400 bps möglich sind, erlaubt der neue Standard Raten bis zu 28 200 bps. Ernsthaftes 'Mobile Computing' ist also durchaus mit den TETRA-Bündelfunksystemen möglich und die Aussicht auf neue Entwicklungen läßt dabei sogar auf eine noch bessere Zukunft hoffen[28].

---

[27] Bündelfunk nach Standard 'ZVEI-RegioNet 43' von 1989
(ZVEI - Zentralverband Elektrotechnik und Elektronikindustrie e.V.)

[28] siehe [Lobensommer 1994, S. 201ff] u. [Decker, Walke 1993]

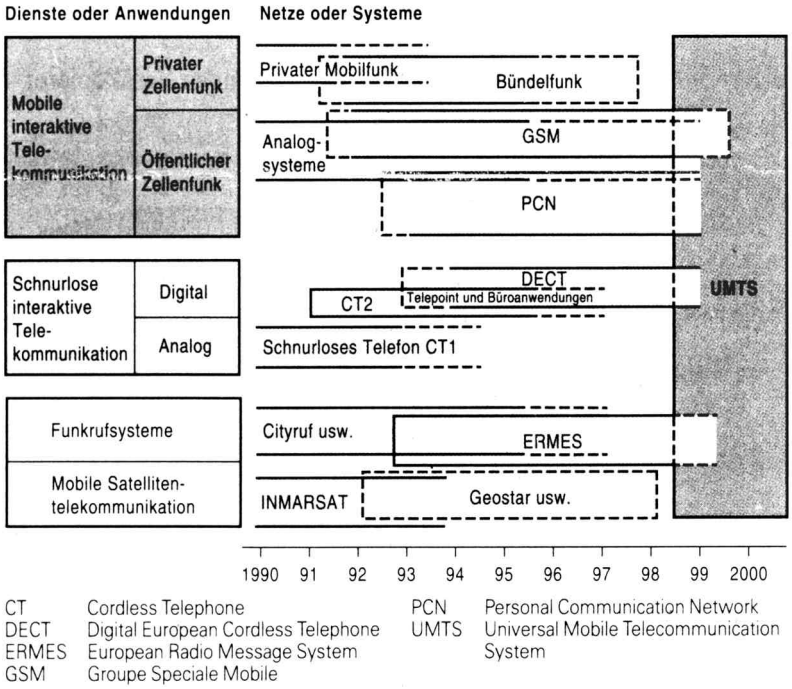

Abbildung 2: Mobile Kommunikation im Wandel der Zeit[29]

[29] [Becker 1991]

2.1.2 Stand heute

Der heutige Stand in der mobilen Kommunikation ist durch die Installation von GSM-Netzen in zahlreichen europäischen, afrikanischen und asiatischen Ländern geprägt. Die Anerkennung dieses Systems als länderübergreifenden Standard und die Verknüpfung der nationalen Netze zu einem internationalen Netzwerk ermöglicht die Nutzung GSM-fähiger Handhelds in vielen Staaten. Der ostasiatische und der amerikanische Raum bleiben bei dieser Entwicklung jedoch derzeit außen vor. Deren Lösungen, der japanische 'Japanese Digital Communications Standard' (JDC) und der amerikanische 'Digital Advanced Mobile Phone Service' (D-AMPS), sind zum GSM-Standard nicht kompatibel.[30] In Deutschland haben sich seit Juli 1992 zwei zellulare Netze nach dem GSM-Standard etabliert, das D1-Netz der DeTe-Mobil (Tochter der Deutschen Telekom) und das D2-Netz der Mannesmann Mobilfunk GmbH. Die verwendete Technik ist grundsätzlich gleichwertig und beliebig austauschbar. Die Netze unterscheiden sich allein im Ausbaugrad, in der Gebührenstruktur und dem Dienstangebot, doch auch hier nivellieren sich die Unterschiede immer weiter. Flächendeckung ist bei beiden Diensten für 1997 vorgesehen und die letzten ausstehenden, im GSM-Standard definierten Dienste werden zügig implementiert.[31]

GSM-Zusatzdienste:

- Sperren in beiden Richtungen
- Rufumleitung
- Rufnummernanzeige und Gebührenanzeige
- Gespräche halten
- Warteschlange für Anrufe
- Anklopfen
- Konferenzgespräch
- Festlegung geschlossener Benutzergruppen
- Automatischer Rückruf bei vorübergehend besetzter Zielrufnummer
- Gebührenfreier Anruf und Gebührenübernahme (R-Gespräch)
- Prioritätsruf
- Operatorgestützte Dienste
- weitere Dienste, wie z.B. Mailbox, Auskunfts- und Informationsdienste

[30] siehe [Huber 1995]

[31] vgl. [Niemeier, Schäfer, Engstler, Kroll 1994, S. 58f]

GSM-Basisdienste:

- Sprachübertragung
  Telephonie und Notruf
- Datenübertragung
- Faxübertragung
- Cell Broadcast
  Punkt-zu-Mehrpunkt-Kurznachrichtendienst
- Short Data Message (SDM)
  Punkt-zu-Punkt-Verbindung (Übertragung alphanumerischer Zeichen)
  mit Empfangsbestätigung von der Mobilstation
  und Zwischenspeicherung der Nachricht bei Nichterreichbarkeit

Ähnlich länderübergreifend erweist sich der europäische Paging Dienst nach dem Funkruf-Standard 'ERMES' und die Normierung des DECT-Standards (Digital European Cordless Telecommunication) für die Heimtelephonie über Funk. Aus der Standardisierung und des damit stark gewachsenen Nutzerpotentials dieser Systeme resultieren fallende Preise bei den Endgeräten, denn teure Entwicklungen können auf eine größere Menge von Benutzern umgelegt werden. Die großen Märkte locken neue Marktteilnehmer an und führen zu Konkurrenz und Preisdruck.[32]

Währenddessen erscheint eine neue Vision am Horizont, die des Personal Communication Network (kurz PCN, auch Personal Communication Service - PCS). Die fortwährende Erreichbarkeit der Teilnehmer. Ob zu Hause, im Büro oder am Arbeitsplatz, alle Anrufe werden zum Aufenthaltsort der Person in die unterschiedlichsten Netze umgeleitet. Jeder Teilnehmer des PCN erhält dafür eine eindeutige universelle persönliche Rufnummer und eine Ausweiskarte, ähnlich des SIM für Handhelds. Diese Ausweiskarte, eingeschoben in ein Endgerät, bewirkt die Umleitung sämtlicher Anrufe für die zugehörige Rufnummer an das Endgerät. Dabei werden sowohl die stationären Endgeräte als auch die mobilen Endgeräte unterstützt. Zusätzlich kann die Ausweiskarte dazu genutzt werden, abgehende Verbindungen von dem persönlichen Gebührenkonto abzubuchen.

Als Ergänzung für die privaten Hauptanschlüsse gedacht, stellt PCN hohe Anforderungen an die verfügbaren Teilnehmerkapazitäten. Herkömmliche Mobilfunknetze, wie das analoge C-Netz oder die beiden GSM-Netze (D1, D2), können diese nicht erfüllen. Das ETSI untersuchte daher schon seit Ende der 80'er Jahre potentielle Kandidaten. Dieser Untersuchung ist das Digital Cellular System 1800 (DCS 1800) entsprungen, ein an dem GSM-Standard angelehntes Mobilfunknetz mit wesentlich höherer Teilnehmerkapazität. Die Übertragungstechnik dieses Standards bedingt

[32] siehe [Becker 1991]

wesentlich kleinere Zellradien als bei GSM, führt aber wegen der geringeren Leistungsaufnahme und der bewährten Technik zu kleineren, handlicheren und kostengünstigen Endgeräten. In einigen europäischen Ländern sind schon Lizenzen für den Betrieb von DCS 1800 Netzen vergeben worden. In Großbritannien betreibt z.B. Mercury (eine Tochter von Cable & Wireless) das 'One 2 One'-Netz. Selbst für die USA liegt die Einführung des DCS 1800 Standards im Bereich des Möglichen. In Deutschland ist im Februar 1993 eine DCS 1800 Lizenz, in der Tradition der deutschen Mobiltelephonnetze mit E1, bzw. E-Plus benannt, an ein Konsortium vergeben worden. Dieses Konsortium besteht aus elf Unternehmen mit den Hauptaktionären Thyssen AG und Veba AG. Aber trotz Einführung dieser Netze bleibt PCN derzeit eine Vision. Die Integration der bestehenden Festnetze und der DCS 1800 Netze in ein gemeinsames PCN-Netz ist momentan nicht abzusehen. Die Liberalisierung der Festnetze und die Erteilung neuer DCS 1800 Lizenzen, ab 1997 haben die beiden D-Netz Betreiber die Möglichkeit DCS 1800 Lizenzen zu erwerben, können langfristig diesen Sachverhalt ändern.[33]

Vor dem Hintergrund der mobilen Datenkommunikation fallen die Schwächen der derzeitigen Systeme zur mobilen Kommunikation auf. Jedes der derzeit angebotenen Systeme kann nur bedingt Ersatz für das Festnetz sein. Bietet sich eine Alternative der Festnetznutzung für die Datenübertragung, so sollte sie derzeit präferriert werden. Im folgenden ein kurzer Überblick über die aktuellen Möglichkeiten der mobilen Datenkommunikation via Funk in der Bundesrepublik Deutschland:

Funkruf (bzw. Paging)

Die derzeitig angebotenen Paging-Dienste wie Eurosignal und Cityruf können nur als bedingt geeignet für die Datenkommunikation klassifiziert werden. Die Möglichkeiten der Übertragung von Tonsignalen und kurzen alphanumerischen Zeichenketten werden wohl in den wenigsten Fällen ausreichen. Hinzu kommt, daß diese Systeme nur unidirektional arbeiten und die Signalaussendung eingeschränkt ist. Die Bestätigung des Informationserhalts oder gar bidirektionaler Informationstransfer ist mit den Endgeräten nicht möglich. Eine gewisse Erleichterung ist erst durch den international genormten ERMES-Standard abzusehen, der die unidirektionale Übertragung von bis zu 8KByte erlaubt.[34] Die ersten Lizenzen für ERMES-Dienste werden im April 1996 vergeben.

Bündelfunk

Der analoge Bündelfunk, wie z.B. die Telekom-Implementation Chekker in den Ballungsgebieten, basiert auf analoger semiduplex Übertragungstechnik mit Ausrichtung auf die Sprachübertragung. Mit dieser Konstellation ist Datenübertragung mit maximal 2 400 bps möglich. Erst der neue europäische Standard TETRA mit seiner digitalen Übermittlungstechnik und den Zusatzmöglichkeiten, wie Verschlüsselung und Roaming, wird den Bündelfunk als Plattform der mobilen

[33] siehe [Lobensommer 1994, S. 160ff]

[34] siehe [Lobensommer 1994, S.171ff]

Datenkommunikation attraktiver machen. Dabei werden spezielle Datendienste, die bis zu 28 200 bps leisten können, angeboten werden. Ein weiterer Vorteil von TETRA wird es sein, Datenübertragung ggf. paketorientiert abwickeln zu können. Damit ist es nicht mehr notwendig, während der gesamten Kommunikation eine Verbindung zwischen Sender und Empfänger aufrecht zu erhalten. Die Daten werden vielmehr als Pakete in Einzelverbindungen versendet. Vorteile dieser Kommunikation liegen im Gebührenabrechnungsmodus, der nicht mehr auf Zeit sondern auf Übertragungsmenge basiert, und in der erhöhten Verfügbarkeit von Frequenzen.[35] Allerdings liegen dem ETSI derzeit zwei verschiedene Normungsvorschläge vor, zwischen denen eine Entscheidung noch aussteht.[36]

Zellulare Mobilfunksysteme

Bei den zellularen Mobilfunksystemen ist zwischen der älteren Generation, dem analogen C-Netz, und der neueren Generation, den GSM-Netzen, zu unterscheiden. Das C-Netz wurde auf die Sprachkommunikation optimiert, bietet aber aufgrund der analogen Übertragungstechnik die Möglichkeit, herkömmliche Analogmodems an die Endgeräte anzuschließen. Allerdings ist bis auf den Faxdienst, der 9 600 bps erlaubt, nur Datenübertragung mit 2 400 bps möglich. Spezielle Dienstangebote, wie die oben angesprochene Datex-J und Datex-P-Anbindung und der Mobilbox, tragen dieser Möglichkeit Rechnung. Die drei in der Bundesrepublik Deutschland installierten GSM-Netze D1, D2 und E-Plus unterscheiden sich wegen der gleichen zu Grunde liegenden Technik nicht in den Möglichkeiten zur Datenübertragung. Die maximale Übertragungsrate liegt bei 9 600 bps. Dabei ist zu beachten, daß Datenübertragung mit GSM als Zusatzleistung aufgefaßt wird und der Nutzer somit mehr oder minder hohe Zusatzgebühren an den Netzbetreiber entrichten muß. Ähnlich dem Bündelfunk steht auch bei GSM noch die Einführung von paketorientierten Datenübertragungsdiensten aus. Wie bei TETRA würde dieser Dienst erhöhte Frequenzverfügbarkeit und oftmals eine günstigere Gebührenabrechnung anbieten.

Mittlerweile existiert eine Lösung des Unternehmens AVM, welche die Übertragung digitaler Daten ohne den Umweg der Emulation eines Analog-Modems erlaubt. In Kombination mit einem Siemens S4 Handheld kann die AVM-PCMCIA-Karte M1 als ISDN-Endgerät zur Datenübertragung genutzt werden. Die im Vergleich zu anderen Lösungen wesentlich kürzeren Verbindungsaufbauzeiten (ca. 4 sec. mit der AVM-Lösung gegenüber 20 sec. bei herkömmlichen Lösungen) erlauben den kostengünstigen Aufbau und die Nutzung semi-permanenter Verbindungen.[37]

---

[35] siehe [Decker, Walke 1993]

[36] TETRA25, protegiert von Motorola, Bosch, Philips Kommunikations Industrie und TETRAPOL protegiert von AEG Mobile Communications

[37] siehe [Eberlen 1996]

Nicht unerwähnt soll an dieser Stelle bleiben, daß nicht an allen Orten, an denen Telephonieren mit Handhelds möglich ist, auch die Datenübertragung möglich ist. Die hohen Anforderungen des Datenübertragungsdienstes an die Verbindungsqualität lassen die Nutzung desselben an manchen Orten zum Glücksspiel werden, an einer Datenübertragung aus schnell fahrenden Autos ist z.B. gar nicht zu denken.

2.2 Groupware

Die relativ kurze Geschichte von Groupware schlägt sich in der Offenheit und Weite der behandelnden Literatur nieder. Strömungen innerhalb der Forschung sind ausgeprägt und mannigfaltig differenziert. Klare griffige Definitionen sind somit nicht einfach zu erlangen. Hier aber dennoch ein kurzer Abriß über die Vielfältigkeit des Feldes Groupware.

"Die informationstechnische Unterstützung der Gruppenarbeit ergänzt zunehmend den computergestützten Einzelarbeitsplatz und steht etwa seit 1987 im Blickpunkt der Forschung. Diese Unterstützung zielt auf die Verbesserung der Gruppenarbeit. Unter Bezeichnungen, wie 'Computer Supported Cooperative Work (CSCW)', 'computergestützte Gruppenarbeit', 'Computer Aided Team (CATeam)' und 'Groupware' entwickelt sich ein neues Anwendungsfeld für Informations- und Kommunikationstechnologien[38]."

"Groupware ist ein computer-basiertes System zur Unterstützung von Gruppen bei der Erfüllung einer gemeinsamen Aufgabe, wobei vorrangig die Koordination, das Treffen von Gruppenentscheidungen, die Kommunikation sowie die gemeinsame Bearbeitung eines Objektes unterstützt werden. Groupware kann Gruppen unterstützen, deren Mitglieder sich am selben Ort oder an unterschiedlichen Orten befinden, wobei diese Unterstützung synchron oder asynchron erfolgen kann. Die Interaktion kann formal oder informal, spontan oder geplant sowie strukturiert oder unstrukturiert erfolgen.[39]

Abgetragen auf einer Orts-Zeitmatrix ergeben sich hierbei folgende Kategorien[40]:

Kooperative Arbeit von Gruppen zur gleichen Zeit am gleichen Ort

Typischer Vertreter dieser Kategorie sind Group Decision Support Systems (GDS), der gemeinsame Arbeitsprozeß im 'Electronic Meeting Room' unter Zuhilfenahme von vollcomputerisierten, vernetzten und durch zentrale Einheiten koordinierten Sitzungsplätzen.

Kooperative Arbeit von Gruppen zu unterschiedlichen Zeiten an beliebigen Orten

Die Gruppe der asynchronen CSCW-Systeme ist relativ unabhängig von räumlicher Trennung, so daß diese Kategorie beide Möglichkeiten umfaßt. Typische Vertreter dieser Kategorie sind E-Mailsysteme, Projektmanagementsysteme, Informationsharing-Systeme und Workflow-System

[38] [Krcmar 1992, S. 425]

[39] [Petrovic 92, S. 17]

[40] siehe [Encarnacao, Hornung, Noll 1994] u. [Lewe, Krcmar 1991]

Kooperative Arbeit zur gleichen Zeit an unterschiedlichen Orten

Diese Gruppe bietet den Vorteil der persönlichen, ggf. bildgestützten Kommunikation, trotz gegebener räumlicher Trennung. Entwicklungen, wie Joint-Editing und Konferenzsysteme, resultieren im Desktop Conferencing und vermeiden somit oftmals hohen zeitlichen und finanziellen Aufwand.

Neben diesen Ordnungskriterien existieren weitere, die hier nicht aufgeführt werden.[41]

Kennzeichen des Forschungsgebietes computergestützten Gruppenarbeit ist die interdisziplinäre Ausrichtung: "CSCW as a field of research is interdisciplinary. It involves computing, mathematics, psychology, sociology, linguistics, system theory, engeneering and other perspectives"[42]. Gerade diese Vielfältigkeit und der Facettenreichtum des Begriffes Groupware erschwert die Benennung konkreter Anforderungen an Groupware-Produkte. Spezialisierte, hoch differenzierte Produkte, die nur wenigen Aspekten der oben angesprochenen Auffassungen genügen, werden sich bestimmt nicht mit dem Prädikat 'Groupware' schmücken können. Unter besonderer Berücksichtigung von Workflow-Systemen sollte Groupware z.B. die folgenden Merkmale aufweisen:[43]

- Verteilte Datenbanken
- Verbunddokumente (Compound Documents)
- Integrierte Gruppenkommunikation
- Text- und Dokumentenmanagement
- Import und Export
- Verwaltung externer Datenbestände
- Electronic Mailing
- Benutzerfreundliches und leistungsfähiges Login
- Sicherheitskonzepte

"Im hier verwendeten Sinn handelt es sich bei Groupwaresystemen um Softwareprodukte, die es Arbeitsgruppen ermöglichen, effizient und effektiv im Rahmen gemeinsamer Aufgabenstellungen zusammenzuarbeiten und die gleichzeitig dazu beitragen, Informationen im Rahmen von Arbeitsprozessen besser zu erschließen und zu verwerten."[44]

---

[41] siehe [Krcmar 1992]

[42] [Robinson 1993, S. 157]

[43] vgl. [Nastansky 1994a, S. 283ff]

[44] [Finke 1992, S. 25]

2.2.1 Derzeitiger Entwicklungsstand

Die Euphorie in der wissenschaftlichen Forschung auf dem Gebiet der 'Computer Supported Cooperative Work' greift zunehmend in Form konstruktiver Auseinandersetzung auch auf die Praxis über. Wie bei vielen Anwendungen eilt der tradierte Pragmatismus der Anwender den Forschungsergebnissen dabei jedoch mit mehr oder minder großem Zeitabstand hinterher. Möglichkeiten existierender Produkte werden dabei oftmals nur unzureichend ausgeschöpft und die notwendige Geschäftsprozeßumgestaltung wird nur halbherzig oder gar überhaupt nicht durchgeführt. Erfahrungen aus der Vergangenheit, daß Forschungsergebnisse erst nach einigen Zögern von den Zielgruppen adaptiert werden, und die steigende Notwendigkeit, Geschäftsprozesse aus Gründen der Wettbewerbsfähigkeit zu reorganisieren, lassen dabei jedoch stetig steigende Akzeptanz erhoffen.

Viele Unternehmen bahnen sich ihren Weg dabei Schritt für Schritt. So legt die in vielen Produkten angebotene Mailingfunktion oftmals den Grundstein für die weitere Beschäftigung mit den Groupware-Produkten. Nach und nach werden dabei Erkenntnisse aus der Arbeit mit dem Produkt gewonnen und mit den Unternehmensbedürfnissen abgeglichen.

Es ist also zwischen dem aktuellen Forschungsstand und der Nutzung in den Unternehmen zu differenzieren. Während die Forschung in einer Vielzahl von Projekten die Wege für zukünftige Anwendungsfelder ebnete, basieren die meisten in der Praxis eingesetzten Lösungen auf wenigen ausgewählten Produkten und Einsatzfeldern.

Im folgenden ein kurzer Auszug der auf dem Markt verfügbaren Groupware-Produkte:

Notes der Lotus Development Corporation

Notes stellt den derzeitigen de-facto Standard auf dem Groupware-Markt dar. Die frühe Präsenz auf dem Markt ließ Notes zum Marktführer werden, der Aufkauf des Unternehmens Lotus Development Corporation durch IBM trägt zum Ausbau dieser Position bei. Notes basiert auf der Haltung multimedialer Daten in verteilten Datenbanken, die unter der bewußten Hinnahme von Inkonsistenzen zyklisch abgeglichen werden. Diese Grundlage erlaubt die Nutzung von Notes in dezentralen Strukturen. Ob im LAN- oder WAN-Verbund, ob im mobilen Betrieb via Modem, Notes ermöglicht jedem Anwender die Teilnahme an der computergestützten Gruppenarbeit. Angelehnt an dem Client-Server-Konzept treten dabei die auf den Computern der Anwender lokalisierten Notes-Clients mit den als zentrale Instanzen gehaltenen Notes-Servern in Kontakt und können so multimediale Informationsinhalte austauschen. Unter Wahrung von Sicherheitsaspekten und abhängig von den konkreten Zugriffsrechten können Informationen damit anderen Benutzern zugänglich gemacht werden. Die Unterstützung verschiedenster Betriebssystem-Plattformen hilft dabei, die Barriere der Dateiformate beim Informationsaustausch zu überwinden und zeichnet dieses Produkt gegenüber vielen Konkurrenzprodukten aus.

Lotus Notes, Ende des Jahres 1989 auf den Markt gekommen[45], liegt mittlerweile, im Frühjahr 1996, in der Version 4.0 vor und glänzt durch zusätzliche neue Funktionalitäten. Die Einbindungsmöglichkeit in den World-Wide-Web-Dienst des Internet und die neue Lotus-eigene anwendungsübergreifende Scriptsprache LotusScript sind nur einige davon. Zusammenfassend läßt sich sagen, daß die lange Marktpräsenz und der Entwicklungsvorsprung Notes als das ausgereifteste Groupware-Produkt auf dem Markt erscheinen lassen. Die freie Programmierbarkeit des Systems und die Verfügbarkeit von Individuallösungen auf Basis von Notes bei diversen Softwarehäusern tragen einen wesentlichen Teil dazu bei[46].

GroupWare von Novell

Unter dem Namen Novell GroupWare firmiert ein Konglomerat von drei sich ergänzenden Anwendungen: Zum ersten GroupWise, ehemals WordPerfect Office 4.0, welche E-Mail-, Kalender- und Aufgabenverwaltungsfunktionalität integriert. Zum zweiten InForms, welche die Erstellung, Dissemination und Auswertung elektronischer Formulare erlaubt. Zum dritten SoftSolutions, welche die Verwaltung verteilt abgelegter Dokumente und die Nutzung von Volltextsuche über alle Dokumente ermöglicht. Ebenfalls nach dem Client-Server-Konzept aufgebaut, treten dabei GroupWare-Clients mit GroupWare-Servern in Kontakt und können so unter Wahrung von Sicherheitsmechanismen und Zugriffsrechten neue Informationen ablegen bzw. erfaßte Informationen einsehen.[47] Zu beachten ist dabei allerdings, daß, im Gegensatz zu Lotus Notes, diese Produkte nur bedingt programmierbar sind und jede Anwendung einzeln erstanden werden muß. Sollen nur die mitgelieferten Funktionen genutzt werden oder lassen sich die Vorhaben mit InForms in Formularen abbilden, so reicht GroupWare vollkommen aus. Projekte in Richtung Workflow-Automatisierung hingegen werden nur schwer zu realisieren sein.

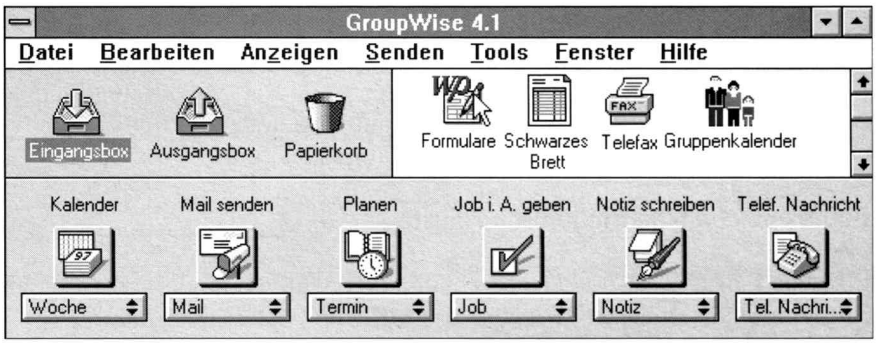

Abbildung 3: Novell-GroupWare - GroupWise

[45] vgl. [Nastansky 1990]

[46] siehe [Lotus 1995]

[47] siehe [Novell 1995]

Apple Open Collaborative Environment der Apple Computer Inc.

Das Unternehmen Apple integriert seit einigen Jahren in das System 7 des Macintosh eine Technologie, die mit 'Apple Open Collaboration Environment' (AOCE) betitelt wird. In jedem Macintosh-Betriebssystem ist dabei das Client-Modul 'PowerTalk' integriert. Das Server-Modul für Apples Workgroup-Server 'PowerShare' ist zusätzlich zu erwerben. PowerTalk ermöglicht die Nutzung von E-Mail und Verzeichnisdiensten. Steht kein PowerShare-Server zur Verfügung, kann E-Mail nur dann versandt werden, wenn sich der Adressat ebenfalls an einem eingeschalteten Macintosh im gleichen Netz befindet.

Gateway-Services zur Vermittlung von Mails in andere Netze und Mailzwischenspeicherung für aktuell nicht erreichbare Adressaten sind erst mit Installation eines PowerShare-Server verfügbar. Auch der Verzeichnisdienst von PowerTalk ist ohne Server auf lokale Verzeichnisse beschränkt. Erst mit der Einführung eines Servers sind Verzeichnisse möglich, die von einer Vielzahl von Anwendern mit Daten versehen bzw. eingesehen werden können. Die Gestaltung solcher Verzeichnisse ist dabei beliebig modifizierbar. Von einfachen Adreßbüchern, in denen z.B. PowerShare selbst seine Benutzerinformationen abspeichert, bis hin zu Dokumentenrepositories ist vieles realisierbar. Auch hier ergibt sich ein Markt für Lösungen unabhängiger Softwarehäuser. Apple verfolgte von Anfang an eine offene Architektur bezüglich des AOCE, Programmierschnittstellen und Gestaltungstemplates erlauben die Integration in eine Vielzahl von Anwendungsprogrammen und die Nutzung des Verzeichnisdienstes für eine Vielzahl von Zwecken.[48]

[48] siehe [Apple 1993]

2.2.2 Aktuelle Entwicklungen

Wie schon in 2.2.1. erwähnt, muß zwischen den Entwicklungen in der Forschung und in der Praxis differenziert werden. Wie in den meisten Gebieten der Informationstechnologie hinkt die Praxis der Forschung um einige Zeit hinterher. Mangelnde Flexibilität, Befürchtungen ob ungewisser Risiken und Unkenntnis bestimmen dabei die Distanz der Praxis zur Forschung. Im folgenden daher ein kurzer Blick auf die derzeit aktuellen Entwicklungen in den Unternehmen bzw. auf den Märkten.

Aktuelle Entwicklungen in der Groupware-Technologie werden durch eine Vielzahl von Einzelentwicklungen initiiert. Viele sind ursprünglich noch nicht einmal im Hinblick auf Groupware erforscht worden. Aus dem Hardwarebereich diffundieren Entwicklungen im Netzwerksektor, die den Rahmen des Local Area Networks (LAN) sprengen und somit den Schritt hin zur Globalität ermöglichen. Im Softwarebereich deutet sich die Verlagerung des klassischen Kampfes um das bessere Betriebssystem ab. Ansätze Groupware-Produkte mit Betriebssystemen zu bündeln oder Betriebssysteme mit integrierter Groupware-Funktionalität anzubieten weiten diese Auseinandersetzung auf das Gebiet der computergestützten Gruppenarbeit aus.

Die Ausweitung der Netzwerke auf dezentrale Organisationsstrukturen, unternehmensübergreifende Netze und globale Strukturen werden durch Fortschritte in der Datenübertragungstechnologie ermöglicht. Die folgende Stufe, die Anbindung von Privathaushalten an eine globale Informationsgesellschaft, ist schon über das Anfangsstadium hinausgewachsen. Online-Dienste, wie z.B. AOL, CompuServe, Europe Online und Prodigy, bieten jeder interessierten Person eine Teilnahme an diesem Geschehen an. Technisch basieren diese Wide-Area-Networks (WAN) auf immer neueren Technologien, die in immer schnellerer Folge immer höhere Leistungsfähigkeit bieten. So baut z.B. die Deutsche Telekom AG zur Zeit in einem Modellversuch DATEX-M auf, ein auf dem Asynchronous Transfer Mode (ATM) basierendes Netzwerk mit Übertragungsraten bis zu 155 MBit/sec. Geplant sind Bandbreiten bis zu 650 MBit/sec. Dieses Netz erlaubt neben anderen Merkmalen die Reservierung von Bandbreiten für bestimmte Anwendungen, welche einen Ansatz zur Echtzeitfähigkeit darstellt und z.B. Videokonferenzen erst ermöglicht.[49]

Parallel zur Entwicklung der Netzwerktechnologie entwachsen auch die Netzwerkanwendungen ihren Kinderschuhen. Verteilte heterogene Systeme, die in Netzwerkbetriebssystemen resultieren, das Client-Server Konzept und viele weitere Entwicklungen aus der Informatik wirken sich auf die derzeitigen Anwendungsfelder und somit auch auf Groupware aus.

Die Chance wahrnehmend und aus der Vergangenheit lernend haben sich viele Computerunternehmen in Normungsgremien versammelt, um die Interoperabilität ihrer Lösungen zu wahren. (Derzeit existieren z.B. folgende Herstellervereinigungen und Normungsgremien, Open Software

[49] siehe [Vertegaal, Guest 1995]

Foundation (OSF), Distributed Computing Environment (DCE), Object Management Group (OMG))[50]

Die Entwicklung schreitet somit fortwährend voran. Während in der Vergangenheit eher einfache Dienste, wie z.B. E-Mailing, genutzt wurden, nehmen die Möglichkeiten in den tradierten Anwendungsfeldern zu. Bereichert um Entwicklungen bezüglich verteilter Datenbanken, Echtzeitfähigkeit, plattformübergreifender Datenaustausch und multimediale Datenformate profitieren auch Groupware-Produkte davon. Die Ausweitung von Workflow-Systemen auf Wide-Area-Workflow-Systeme, um die dezentral organisierte Unternehmung zu integrieren oder mit den Geschäftskontakten zu interagieren, und Videokonferenz-Systeme halten Einzug in den Alltag.

Auf Seiten der Software bestechen zwei aktuelle Entwicklungen, einerseits der Aufkauf der Lotus Development Corporation durch IBM und andererseits die Bestrebungen Microsofts ein Groupwareprodukt ('Exchange') in ihre Betriebssysteme zu integrieren. IBM hofft durch den Kauf des renommierten Softwarehauses die Softwarebasis für das Betriebssystem OS/2 zu erhöhen und die direkte Konfrontation mit dem Konkurrenten Microsoft durch das Groupware-Produkt Notes zu überwinden. Microsoft folgt damit dem Beispiel von Apple, die mit Ihrer 'Apple Open Collaborative Environment' (AOCE)-Technologie schon seit einigen Jahren über eine betriebssystemintegrierte Groupwarelösung verfügt, und versucht die Windows-Plattformen weiter als de-facto-Standard auszubauen. Als Resultat dieser beiden Bestrebungen liegt heute jedem OS/2-Installationssatz eine Notes-Desktop-Version bei und ist in jedem Microsoft Windows 4.0 (bzw. Windows 95) Installationssatz ein Exchange-Client enthalten. Dabei ist zu beachten, daß Exchange eine rein auf die Windows-Plattformen zugeschnittene Lösung ist. Portierungen auf andere Betriebssysteme sind nicht vorgesehen und der Auslieferungstermin für die Serverkomponente ist noch ungewiß. Allein Notes bietet plattformübergreifende Groupwarefunktionalität (16 Plattformen sind geplant) und stellt dem Nutzer in der Version 4.0 noch zusätzliche die Anbindung an den WWW-Dienst des Internet zur Verfügung.[51]

Vergessen werden darf an dieser Stelle nicht das Aufkommen von Intranets, d.h. die Nutzung von Internet-Diensten für die unternehmensweite Kommunikation. Die Verfügbarkeit der notwendigen Client-Software, wie z.B. WWW-Browser, FTP-Clients, Usenet-Clients und Gopher-Clients für eine Vielzahl verschiedener Plattformen machen Intranets gerade für heterogenen Rechnerlandschaften attraktiv.

Die kostengünstige, in vielen Fällen gar kostenlose Verfügbarkeit dieser Software trägt noch zusätzlich dazu bei. Allerdings darf der Wartungsaufwand und der damit verbundene Personalaufwand derartig heterogener Lösungen nicht unterschätzt werden. Als Resultat derartiger Bemühungen stehen dem Benutzer Dokumentenarchive im Hypertext-Markup-Language-Format

[50] vgl. [Geihs 1993]

[51] siehe [Weber 1996]

(kurz HTML) zur Verfügung, die mit WWW-Browsern eingesehen werden können. Datenbanken können über HTML-Masken in WWW-Browsern abgefragt werden und Diskussionsforen können mit den im Internet gewohnten Usenet-System organisiert werden. Steckt die Technologie der Intranets auch noch in den Kinderschuhen, so zeigt sich doch, daß in Zukunft in diese Richtung zahlreiche Entwicklungen zielen werden.

3. Das MobileNotes Projekt

3.1 Ziele des Projektes

Intention von MobileNotes ist die Anbindung mobiler Computer, die von Groupwarelösungs-Hersteller nicht mit zugehöriger Client-Software bedacht wurden, an Groupware-Backend-Server. Die Vielfalt der Kategorie 'Mobiler Computer' und der sich darin tummelnden Lösungen verbietet den Groupwarelösungs-Hersteller oftmals eine wirtschaftlich zu rechtfertigende Anbindung. Dabei können einzelne Mitglieder dieser Kategorie durchaus eine wirtschaftlich attraktive Basis für eine Client-Implementation darstellen. Die Integration von Telephon-, Modem- und Computertechnologie in Klein- und Kleinstgeräte wird heute schon in einer Vielzahl von Geräten angeboten und stellt für viele Einsatzfelder einen sinnvollen Ersatz bzw. Ergänzung für die herkömmliche Peripherie dar.

Klasse	Gewicht/Größe	Merkmale
Netzabhängiger Portable	6-11 kg	Gebrauch von Standard/Add-On -Karten
Batteriebetriebener Laptop	4-9kg	Spezielle Ad-On-Karten begrenzte Portabilität
Notebook	<3,5 kg 30*25*5 cm	Tastatur optimale Portabilität erfordert Unterlage
Sub-Notebook	1-2 kg 21*16*3 cm	kleine Tastatur optimale Portabilität geringere Leistung als Notebooks
Palmtop	0,3-0,6 kg 20*9*2,5 cm	kleine Tastatur kleine LCD-Anzeige geringe Leistung
Notepad, Tablet, Pentop	<2,5 kg 30*25*2,5 cm	Handschrifteingabe Zeigefunktionalitäten kann gehalten werden
Palmpad	0,5 kg	Datensammlung
Personal Digital Assistant (PDA)	0,5 kg	Handtellerformat fokussierte Funktionen optimale Mobilität drahtlose Datenübertragung

Tabelle 1: Hierarchie mobiler Endgeräte[52]

Die fortwährende Neu- und Weiterentwicklung von Klein- und Kleinstcomputern, viele davon tragbar und somit für die mobile Nutzung geeignet, ließ eine Entwicklung der vormals nur auf relativ leistungsstarken und bedingt mobilen Personal Computern lauffähigen Groupware-Client-Software für diese Geräte sinnvoll erscheinen. Bisher ungenügend in der Nutzung von Groupware unterstützte Bereiche können so in diesen Reigen aufgenommen und der Bogen mobiler Groupware-Anbindungen erweitert werden. Denn Geräte, wie z.B. das Philips Screen Phone P100, die

[52] [Niemeier, Schäfer, Engstler, Kroll 1994, S. 90]

eine Telephon-, eine Modem- und eine Computereinheit integrieren, stellen im direkten Vergleich zur üblichen Infrastruktur eines Büroarbeitsplatzes eine kostengünstige und in vielen Fällen ausreichende Lösung dar. Bei der Ausstattung mobiler Anwender mit Computertechnologie kann die Verwendung derartiger Geräte zu wesentlich größerer Unabhängigkeit führen. In ihrer Funktionalität und Arbeitsweise sind sie oftmals besser auf die mobile Arbeit zugeschnitten als die Personal Computer im Notebook- oder gar Laptop-Format. Der Wunsch, die vorhandenen Groupware-Lösungen auch auf diesen Plattformen verwenden zu können, kann daher nur als konsequent angesehen werden. Einschnitte in dem Leistungsumfang derartiger Lösungen können vor dem Hintergrund der geringeren Leistungsfähigkeit der Geräte bezügl. Speicherausstattung, Prozessorleistung und Datenübertragungsgeschwindigkeiten toleriert werden.

Ziel dieses Projektes war es, für das konkrete Groupware-Produkt Notes der Lotus Development GmbH eine Backend-Serverkomponente zu schaffen, welche eine An- und Einbindung derartiger Geräte an das Groupware Produkt erlaubt. Auf der Basis des Screen Phone P100 des Unternehmens Philips Home Services (PHS) Eindhoven entstand zusätzlich ein exemplarischer Client, der die Mächtigkeit der Serverkomponente voll ausnutzt und durch seinen generischen Aufbau als Basis für zukünftige Entwicklungen dienen kann. Spiegelt das P100 auch nicht die angestrebte Kategorie 'mobile Computer' wider, so kann es doch mit seinen eingebauten Möglichkeiten (Telephon, Modem, nicht flüchtiger Speicher) als Referenzplattform verwendet und bezeichnet werden.

Die Wahl des Groupware-Produktes und der damit verbundenen Möglichkeiten beeinflußte den zu implementierenden Funktionsumfang und wirkte sich auf die gestellten Anforderungen aus. Die dokumentenorientierte Arbeitsweise in Notes-Datenbanken sollte adäquat übernommen und abgebildet werden. Die Arbeit mit Dokumenten, die Browsing-Möglichkeiten in Ansichten, die Informationsrepräsentation und -eingabe in Masken, die Nutzungsmöglichkeit eines Mail-Systems und dies alles unter Wahrung von Sicherheits- und Authentisierungsaspekten sollte mit den zur Verfügung stehenden Mitteln realisiert werden.

Die angestrebten Möglichkeiten eines Clients im Überblick:

- Einsicht in Datenbanken und der darin enthaltenen Dokumente auf dem Server
- Erstellung von neuen Dokumenten und Antwortdokumenten
- Abspeicherung von Datenbankinformationen und Dokumenten auf dem Client zu Ermöglichung von Offline-Arbeit
- Wahrung von Sicherheitsaspekten durch Authentisierungs- und Verschlüsselungsmechanismen
- Ermöglichen der Arbeit mit fremden Clients unter der Wahrung von Sicherheitsaspekten

Um die Leistungsfähigkeit vieler Clients nicht zu überfordern, wurden Einschränkungen in diesen Möglichkeiten vorgenommen, die z.B. nur die Verwendung einer Ansicht und einer Maske pro

Datenbank erlauben, nur die Nutzung textueller Informationen erlauben (nicht textuelle Informationen werden vom Server nicht weitergereicht) und nur eine maximale Anzahl von Feldern pro Dokument zulassen.

3.2 Grundlagen und Vorarbeiten des Projektes

Konkrete Grundlage für das Projekt MobileNotes bildete die in einer Seminararbeit an der Lehr- und Forschungseinheit Wirtschaftsinformatik 2 der Universität Gesamthochschule Paderborn von den Studenten Bernd Altmiks, Nico Dirks und Dirk Sievers erstellte Studie 'AvALoN' (Advanced Access to Lotus Notes). Diese Studie (im folgenden kurz AvALoN) erlaubte einem alternativen Frontend den Zugriff auf Datenbanken des Groupware-Produktes Lotus Notes. Miteinander über Modems und Telephonleitungen verbunden konnte der Client mit dem Server Kontakt aufnehmen und nach erfolgter Authentisierung in beschränktem Umfang mit dem Notes-System interagiere

Kurz erwähnt werden muß, daß die Initialzündung der Studie AvALoN und somit auch des Projektes MobileNotes von dem an der gleichen Lehr- und Forschungseinheit entstandenen Projekt TermAcc ausging. Dieses von den beiden Studenten Claudia Köhler und Bernd Altmiks bearbeitete Projekt erlaubte den Notes-Zugriff via Telephonverbindung und Terminalemulation, ging allerdings, durch die großen strukturellen Unterschiede bedingt, nicht direkt in die Kette der Folgeprojekte ein.

Technisch basierte die AvALoN-Lösung auf drei Komponenten, einer Serverkomponente, einer Initialisierungsdatenbank und der Client Komponente, die in gleicher Art im MobileNotes-Projekt wiederzufinden sind. Die erste Komponente, die Serverkomponente wurde in MS Visual Basic 3.0 geschrieben und bediente sich der beiden Produkte MACROWARE.DLL[53] und PDQCom. Die MACROWARE.DLL erlaubt den Zugriff auf Notes-Datenbanken, die PDQCom vereinfacht die Visual Basic gestützte Modemkommunikation. Durch die verwendeten Produkte und die gewählte Programmiersprache bedingt, diente für diese Studie Windows 3.1x als Serverplattform und benötigte als Ausführungsbasis eine Notes-Workstation Installation. Die zweite Komponente, die Initialisierungsdatenbank, erlaubte einem Systemadministrator die Organisation des AvALoN-Systems zu beeinflussen. Durch die Anlage bzw. Modifikation von Benutzerkonten, die Festlegung erreichbarer Datenbanken und die Definition von Maildatenbanken konnte er für einzelne Benutzer Arbeitsumgebungen schaffen, die den Client-Benutzer in seinen Möglichkeiten mehr oder weniger beschränkten. Die dritte Komponente, die Client-Komponente, wurde auf Basis des Screen Phones P100 des Unternehmens Philips Home Services entwickelt. Die verwendete Programmiersprache, Borland C in der Version 3.1, wurde durch die mitgelieferte Entwicklungsumgebung vorgegeben. Das P100 dient auch im MobileNotes-Projekt als Implementationsgrundlage für den Client und wird im weiteren Text intensiver betrachtet.

[53] entwickelt an der Lehr- und Forschungseinheit Wirtschaftsinformatik 2 der Universität-Gesamthochschule Paderborn

Waren die Möglichkeiten dieser Studie noch ziemlich beschränkt, so wurden doch schon viele Grundsteine für das MobileNotes-Projekt gelegt. Doch nun eine kurze Ausführung der AvALoN-Möglichkeiten:

Der Client konnte mit dem eingebauten Modem eine Verbindung zu der Serverkomponente aufbauen. Nach erfolgter Authentisierung, welche die Anmeldeangaben (Name, Paßwort) mit denen in der Initalisierungsdatenbank gespeicherten verglich, stand dem Client-Benutzer seine spezifische Arbeitsumgebung zur Verfügung. Acht Datenbanken konnten in dieser Arbeitsumgebung definiert und somit für den Benutzer zugänglich gemacht werden. In diesen Datenbanken konnte mit den, in der Initialisierungsdatenbank vorgegebenen Ansichten navigiert werden. Enthaltene Dokumente konnten ausgewählt, eingesehen und für den späteren Gebrauch, z.B. Offline-Recherche, auf der Speicherkarte des Telephons abgelegt werden. Dokumentenerstellung war allerdings nur für die persönliche Maildatenbank möglich, jegliche andere Datenbank konnte nur eingesehen werden.

War diese Funktionalität an sich schon recht interessant, so werden im folgenden einige Schwachstellen aufgezählt, welche diese Studie für eine ernsthafte Verwendung disqualifizierten und den Weg für das Projekt MobileNotes bereiteten:

- Arbeitsumgebungen wurden auf Einzelpersonenebene erstellt, Gruppenkonstrukte wurden nicht unterstützt
- Auf acht Datenbanken beschränkte Arbeitsumgebungen
- Unverschlüsselter Kommunikation
- Kommunikation verlief ungeschützt gegen Kommunikationsfehler
- Das verwendete Kommunikationsprotokoll war nicht ausreichend ausgefeilt und zukunftssicher
- Die Erstellung von Dokumenten für beliebige Datenbanken war ausgeschlossen
- Implementation war in großen Teilen sehr hardwareabhängig programmiert

3.3 Alternativen und eingeschlagene Lösungswege

Die aus der Studie 'AvALoN' gewonnenen Erfahrungen und die an dieses Projekt gestellten Anforderungen definierten den für dieses Projekt gültigen Lösungsraum, aber nicht die einzuschlagenden Lösungswege. Bewährtes aus dem AvALoN-Projekt wurde übernommen, zu klein dimensionierte Elemente erneut überdacht und neue Lösungen aufgenommen.

Die vorhandenen Erfahrungen aller Beteiligten mit dem Groupware-Produkt Notes der Lotus Development Corporation bedingten dessen Wahl zur Implementations-Basis. Dieses Produkt erlaubt Arbeitsgruppen den Zugriff auf verteilte Datenbanken mit multimedialen Inhalten und führt momentan den Groupware-Markt an. Aufgrund der ungewissen Verfügbarkeit der Version 4.0 und der Probleme in der Arbeit mit Beta-Versionen wurde anfangs die Entscheidung zu Gunsten der stabilen und erhältlichen Version 3.3 gefällt. Alternative Groupware-Produkte stellten auf Grund der mangelnden Erfahrungen seitens der Projektbearbeiter, der ungewissen Verfügbarkeit und der langwierigen Einarbeitung in komplexe Entwicklungsumgebungen keine ernsthafte Konkurrenz für diese Wahl dar.

Die Wahl des exemplarischen Clients wurde von zwei Faktoren geleitet: einerseits der Wunsch eine generische Client-Implementation zu schaffen, andererseits Bestrebungen vorhandene Erfahrungen zu nutzen. Das Screen Phone P100 der Philips Home Services genügte beiden Faktoren. Programmierung in der Sprache C ist möglich und Erfahrungen waren durch die Studie 'AvALoN' zahlreich vorhanden. Andere Clients, die eventuell das Attribut 'mobil' eher verdient haben, kamen teilweise durch eingeschränkte Entwicklungsumgebungen (Verwendung von Scriptsprachen), den hohen Anschaffungspreis (insbesondere benötigte Zusatzkomponenten) oder ungenügende Erfahrungen nicht in die nähere Wahl.

Technisch wurde bei den Clients die Einhaltung einiger Rahmenbedingungen gefordert:

- Kommunikationsfähig über Modem (Integriert oder anschließbar)
- Programmierbarkeit
- Angebot einer Benutzungsschnittstelle (Ein- und Ausgabemöglichkeit)
- Möglichkeit der dauerhaften Datenspeicherung

Eine Alternative für diese Wahl wäre der Apple Newton gewesen. Der Apple Newton integriert in einem Gerät der Größe einer Videokassette einen ARM-610 RISC-Prozessor, ein 320*240 Pixel LCD-Display, eine serielle Schnittstelle, ein AppleTalk/LocalTalk-Interface und einen PCMCIA-Karteneinschub der Spezifikation 2. Neuere Modelle werden zusätzlich mit einer Infrarotschnittstelle ausgerüstet. Benutzerinteraktion ist über das als Touch-Screen ausgeführte LCD-Display mit Hilfe eines Plastikgriffels möglich. Eine eingebaute Schrifterkennung auf Wortbasis setzt die Benutzereingaben in Zeichen um, die rechnergestützt weiterverarbeitet und gespeichert werden können.

In Zusammenarbeit mit Nokia bietet Apple ein Connectivity-Kit an, welches aus E-Plus Handheld und einem Adapter für den Newton besteht. Miteinander verbunden kann der Newton über das Handheld Telefaxe versenden und Daten übertragen. Diese Kombination stellt eine interessante Plattform für eine Implementation dar. Allerdings ließen der Anspruch, eine generische Client-Implementation durchzuführen und die hohen Hardware-/Entwicklungsumgebungskosten eine Wahl dieses Produktes nicht zu. Denn die Programmierung des Apple Newton ist nur mit dem Newton Toolkit (NTK) auf Macintosh- und mittlerweile auch Windows-Rechnern möglich, welches die Programmerstellung in der Newton-spezifischen Sprache NewtonScript erlaubt.[54] Mittlerweile sind für den Apple Newton von zwei Unternehmen Verbindungsmöglichkeiten zu Notes geschaffen worden. Einerseits bietet das englische Unternehmen Ives Development seit Januar 1996 eine derartige Lösung unter dem Namen 'TeamAgent' an, mit der beliebige Notes-Datenbanken auf dem Newton genutzt werden können. Andererseits bietet das amerikanische Unternehmen SkyNotes Inc. mit ihrem Produkt 'SkyNotes' seit Sommer 1995 eine auf vier vorgefertigte Notes-Datenbanken limitierte Lösung an.

Abbildung 4: Apple Newton

Ein weiterer interessanter Aspirant wäre das HP OmniGo, eine Integration des bewährten HP Palmtop auf Basis von DOS 5.0 und eines Nokia-GSM-Handhelds, gewesen. Die Programmentwicklung ist durch die Unterstützung von DOS 5.0 auf tradierten PC-Entwicklungsumgebungen möglich. Der späte Erscheinungstermin, zur CeBit 1996, ließ dieses Gerät allerdings gar nicht erst in die nähere Wahl gelangen.

Die aus der Studie bewährte Teilung in drei Komponenten (Serverkomponente, Initialisierungsdatenbank und Client-Komponente) wurde beibehalten und wirkte auf die gewählten und schließlich eingeschlagenen Lösungswege ein. Im folgenden werden diese drei Komponenten und

[54] siehe [Flohr 1994] u. [Österle, Riehm 1994]

die in ihnen realisierten Lösungen näher vorgestellt. Die der Serverkomponente beigeordnete Initialisierungsdatenbank wird dabei im Zusammenhang mit der Serverkomponente ausgeführt und nicht gesondert angesprochen.

3.3.1 MobileNotes-Server

Der MobileNotes-Server stellt die Kernkomponente dieses Projektes dar. Prinzipiell zugänglich und nutzbar für jegliche Art von Client, realisiert er einen offenen Zugang zum Groupware-System Notes für eine Vielzahl bisher nicht unterstützter Geräte. Als eigenständiger Prozeß auf einem Wirtssystem laufend, bietet er über angeschlossene Modems einen einfachen und sicheren Zugang an. Geeignete Clients können über eine Telephonverbindung Kontakt mit ihm aufnehmen und über definierte Protokollbefehle Daten abfragen und neu erstellen. Die Wahl des zu Grunde liegenden Wirtssystems erstreckte sich auf alle von Notes unterstützten Plattformen, die endgültige Entscheidung fiel zu Gunsten von IBM OS/2 2.x/3.0 und MS Windows NT 3.5x. Diese Wahl erlaubte die Entwicklung auf Personal Computern und die Verwendung von Hilfsmitteln, wie z.B. die HiTest-Entwicklungsumgebung, die nicht für alle Plattformen verfügbar sind.

Wie schon bei der Studie 'AvALoN' fügt sich die Serverkomponente in ein Gefüge aus Initialisierungsdatenbank und einer Notes-Installation ein. Die Initialisierungsdatenbank dient der Parametrisierung der Server-Komponente und befreit die Beteiligten davon, umständliche Konfigurationsmöglichkeiten warten bzw. ansprechen zu müssen. Die Verwendung einer übersichtlichen Datenbank mit der Möglichkeit alle für den Server relevanten Parameter beeinflussen zu können, stellt einen wesentlichen Schritt hin zu mehr Benutzerfreundlichkeit dar.

Im Überblick integriert sich die Serverkomponente wie folgt:

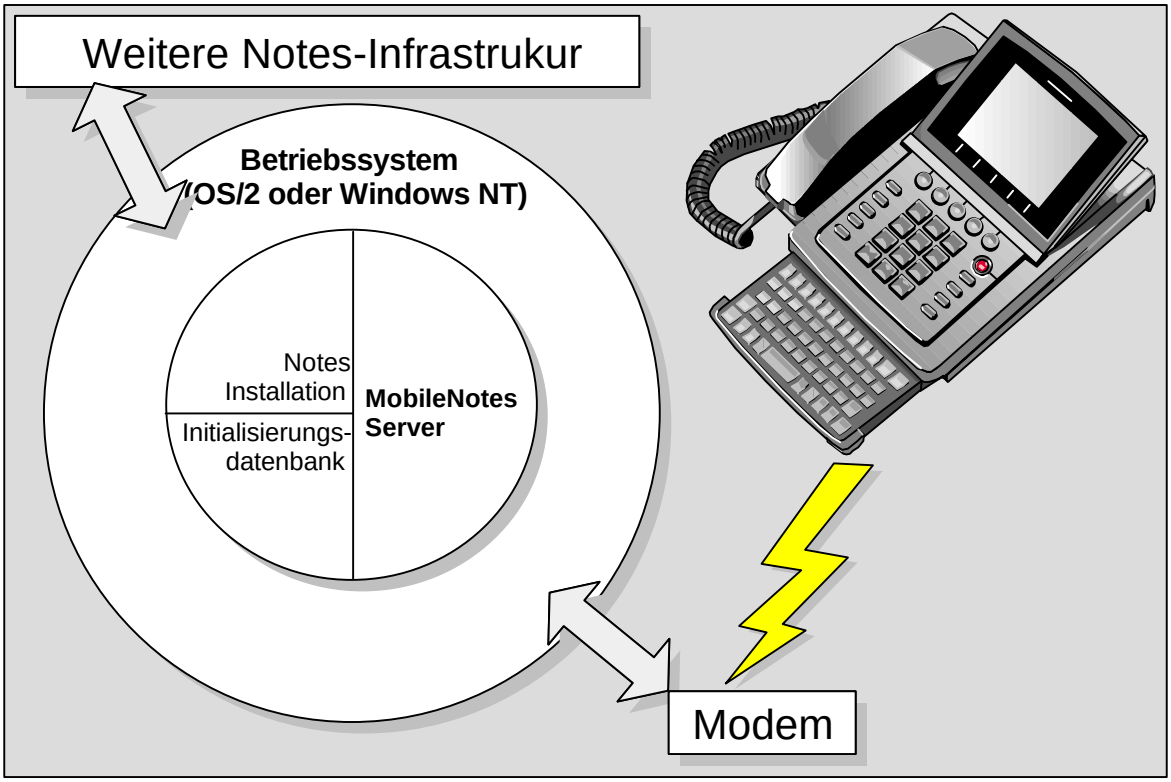

Abbildung 5: Integration der Serverkomponente

Die Kommunikation des Servertasks mit einem über Telephonverbindung verbundenen Client basiert auf strukturierten Protokollbefehlen, die in beide Richtungen ausgetauscht werden. Verlief

die zwingend vorausgesetzte Authentisierung des Client-Benutzers gegenüber dem Servertask erfolgreiIch, so können auf Basis dieser Befehle Informationen zwischen dem Client und dem Server ausgetauscht werden. Grundsätzlich sind die verwendeten Protokollbefehle für alle Clients gleich. Clients können aber, da sie ggf. auf unterschiedlichen Protokollversionen basieren und sich die Protokollentwicklung im fortwährenden Fluß befindet, gezielt bestimmte Protokollversionen anfordern und bestimmte Kommunikationsparameter einstellen (z.B. Fehlererkorrektur im Protokoll einschalten, da das Clientmodem diese evtl. nicht selbständig anbietet).

Grob klassifizieren lassen sich die Protokollbefehle in die folgenden Kategorien, werden darüber hinaus in der Diplomarbeit des Studenten Bernd Altmiks ausführlicher behandelt:

- An-/Abmeldung beim MobileNotes-Servertask
- Abgleich von Datenbankstrukturinformationen
- Navigation in Ansichten
- Austausch von Dokumenteninformationen
- Austausch von Statusinformationen

Die derzeitige Protokollversion ist im Hinblick auf das P100 auf einfache Frontends mit bedingten graphischen Fähigkeiten abgestimmt. Informationen werden auf rein textueller Basis zwischen den Beteiligten ausgetauscht. Es wird hierbei der ASCII-Kode (American Standard Code for Information Interchange, kurz ASCII) verwendet. Multimediale Informationen, wie z.B. eingebettete Graphik-, Klang- oder gar Filmobjekte und die unter Notes möglichen binären Objekte, wie z.B. OLE-Objekte (Object Linking and Embedding, kurz OLE) und Dateianhänge, werden bei der Datenübertragung übergangen. Sie können aber für andere Frontends durch die Implementation einer erweiterten Protokollversion zugänglich gemacht werden. Eine Erweiterung diesbezüglich bietet sich bei Geräten mit Klang-/Sprachwiedergabemöglichkeiten an. Eine Kommunikation mit übertragenen Sprachobjekten, die auf dem Client abgehört werden können, erhöht die Attraktivität eines derartigen Systems wesentlich.

Als Informationsgrundlage dient dem Servertask die MobileNotes-Initialisierungsdatenbank, sie enthält Benutzerdaten, die eine Authentisierung ermöglichen und definiert für die einzelnen Benutzer Arbeitsbereiche. Ein Arbeitsbereich bestimmt die angebotenen Möglichkeiten und legt die dem Benutzer zur Verfügung stehenden Datenbanken inkl. der darauf eingeräumten Rechte fest. Diese Definition wird auf Basis von Personen- bzw. Gruppendokumenten durchgeführt. Jeder Benutzer des MobileNotes-Servers wird durch ein Notes-Dokument definiert, welches seine Authentisierungsdaten, den zu verwendenden Kommunikationsschlüssel und eine Liste von zur Verfügung stehenden Datenbanken enthält. Ist der Benutzer in einer oder mehreren Gruppen enthalten, so kann diese Datenbankliste durch die zugehörigen Gruppendokumente erweitert werden.

Die Liste besteht somit im Endeffekt sowohl aus einzeln, als auch gruppenweise zugeordneten Datenbanken. Jedes Element dieser Ergebnisliste repräsentiert eine nutzbare Datenbank inklusive

der darauf definierten Rechte. Der Servertask ist aber dennoch nicht in der Lage diese Datenbanken direkt dem Benutzer zur Interaktion anzubieten. Die Vielzahl von Ansichten, Masken und Feldern würde viele Clients technisch überfordern und eine automatische Informationsreduktion seitens des Servertasks würde zu ungeeigneten Beschneidungen führen. Aus diesem Grunde wird eine Zwischenschicht, die Datenbankdefinitionsschicht, verwendet. Jede über den Server ansprechbare Datenbank besitzt in der Initialsisierungsdatenbank ein eigenes Dokument, welches die Informationsreduktion durch den Systemadministrator erlaubt und das Erscheinungsbild der Datenbank auf dem Client bestimmt. In diesen Dokumenten werden die anzuzeigenden Ansichten und die zu verwendenden Masken durch die MobileNotes-Administratoren definiert. Es müssen dabei keine konkret in den Datenbanken existierenden Ansichten und Masken verwendet werden, sondern es können mit der Kombination von Feldnamen (ggf. Feldtypen) und Feldlängen ganz neue Konstrukte geschaffen werden.

3.3.2 Exemplarischer Client

Die Client-Implementation wurde auf Basis des Screen Phones P100 von Philips gefertigt, diese Einheit aus Telephon, Modem, LCD-Bildschirm, alphanumerischer Tastatur und Mikroprozessor läßt sich mit einer speziellen Entwicklungsumgebung auf einem IBM kompatiblen PC programmieren. Ursprünglich für die amerikanische City-Bank entwickelt und für das Homebanking positioniert bietet dieses Gerät einige interessante Leistungsmerkmale:

- PCMCIA-Kartenschacht Typ 1 zur Aufnahme von Speicherkarten bis 4 MB Größe
- 256 KB RAM
- 256 KB Flash ROM
- Echtzeitbetriebssystem AMX[55] von Kadak mit dem darauf aufbauenden Phone Operating System (POS)
- Intel 808x kompatibler Prozessor von NEC
- 2400 Baud Modem, den Standards CCITT V21, V23, V33,V22bis bzw. Bell 103, 212 und 212A genügend
- Befehlssyntax nach dem De-facto-Standard des Unternehmens Hayes
- 55 Tasten Tastatur
- Smart-Card Connector nach ISO 7816
- 26 Pin Interface zum Anschluß von Drucker und Tastatur (Adapter wird benötigt)
- LCD-Bildschirm 320*240 Bildpunkte, im CGA-(Color Graphics Adapter) und Textmodus ansteuerbar

Die Möglichkeit der Entwicklung in der Programmiersprache C führte zu einer Implementation, die für viele alternative Frontends angepaßt werden kann. Der Programmfluß ist prinzipiell für eine Vielzahl von Geräten geeignet, die in der Benutzerführung ein ähnliches Paradigma wie das P100 verfolgen. Weicht die Benutzerführung eines Gerätes jedoch sehr von den implementierten Interaktionsmustern ab, (z.B. durch Bestrebungen die Benutzerführung objektorientiert zu gestalten) muß der Programmfluß modifiziert werden.

Dieser Offenheit gegenüber alternativen Frontends tat auch die Implementation in nur einem Prozeß genüge. Die prinzipiell offenstehende Möglichkeit der Verwendung mehrerer Prozesse auf dem P100 wurde zu Gunsten der Plattformunabhängigkeit nicht verwendet.

[55] siehe [Kadak]

Abbildung 6: Screen Phone P100

Kleine Ausnahmen sind die Bildschirmschonerfunktionalität und die Zeit- bzw. Statuseinblendungen. Der Bildschirmschoner wacht als zeitgesteuerte Routine über die Tastatureingaben und schaltet bei längerer Nichteingabe den Bildschirm aus. Der Status wird, ebenfalls durch eine zeitgesteuerte Routine, zyklisch im oberen Bildschirmbereich eingeblendet.

Die aus der AvALoN-Studie erworbenen Erfahrungen mit dem P100 wurden konsequent genutzt und in Verbesserungen umgesetzt. Die oben aufgeführten Schwachstellen konnten, aufgrund des mächtigeren Übertragungsprotokolls, beseitigt werden. Das Ziel der generischen Client-Implementation wurde durch die Nutzung definierter Schnittstellen und die Kapselung von Programmteilen in funktional getrennten Einheiten ermöglicht. Ein Portierungswunsch resultiert somit nur noch in dem Austausch einiger plattformabhängiger Codeteile bzw. Module.

4. Implementation

4.1 Rahmenbedingungen

Die Implementation des vorliegenden Projektes 'MobileNotes' wurde von den drei Studenten Bernd Altmiks, Nico Dirks und Dirk Sievers durchgeführt. Erwachsen aus der im Prototypen 'AvALoN' erprobten Architektur entstanden ein von Grund auf neu implementierter MobileNotes-Server und ein von Grund auf neu entwickelter MobileNotes-Client.

Als Programmiersprache empfahl sich ANSI-C (American National Standars Institute, kurz ANSI) (s. [Schirmer 1992]), dessen Mächtigkeit und Verfügbarkeit diese Wahl bedingte. Für die angestrebten MobileNotes-Serverplattformen existieren zahlreiche C-Compiler, für den Client existiert eine Cross-Entwicklungsumgebung, welche auf dem 'Borland C-Compiler für DOS' in der Version 3.1 basiert. Durch die Fixierung der Cross-Entwicklungsumgebung auf den Borland C-Compiler und der dadurch entstandenen Vertrautheit mit Borland Produkten wurden auch für die Serverimplementationen Produkte von Borland gewählt. Im Einzelnen waren dies:

- Borland C 2.0 für OS/2
- Borland C 4.52 für Windows und Windows NT

In einem Teilbereich, die Kompilierung der Fehlererkennungsroutine für die Datenkommunikation unter Windows NT, mußte auf den 'Visual C++ 2.0' Compiler von Microsoft zurückgegriffen werden. Da dieses Codesegement in Assembler geschrieben wurde und dessen Kompilierung mit dem 'Borland C 4.52 für Windows und Windows NT' nicht ohne weitere Zusatzprodukte möglich ist.

Die aus der Implementation des Prototypen 'AvALoN' gewonnenen Erfahrungen führten bei der Implementation des vorliegenden Projektes zu einer Reihe von Maßnahmen, die den dezentral organisierten Entwicklern die Programmierung wesentlich erleichterten. Zu diesen Maßnahmen zählten:

- Modularisierung der Programme

  Funktionale Einheiten in den zu erstellenden Programmen wurden identifiziert und deren nach außen angebotenen Schnittstellen gemeinsam spezifiziert. Es entstanden modulare Funktionseinheiten mit definierten C-Schnittstellen, deren Implementation in die Verantwortung Einzelner gelegt wurde.

- Verwendung der ungarischen Notation

  Orientierung der Benamung von Funktionsaufrufen und Variablen an der verbreiteten ungarischen Notation, um die Aufgaben derselben an deren Namen ablesen zu können.[56]

[56] siehe [Bucheit 1992, S. 1060ff]

- Definition einer gemeinsamen Verzeichnisstruktur

  Die Verwendung einer gemeinsamen Verzeichnisstruktur erlaubte die uneingeschränkte Nutzung von Projektdateien[57] auf den Rechnern aller Projektteilnehmer.

- Fixierung von Verantwortung

  Die Verantwortung Einzelner für Quellen und Module befreite die anderen Projektteilnehmer von der Einarbeitung in die jeweiligen speziellen Themen.

Wert wurde in allen Bereichen auf die Wiederverwertbarkeit der entwickelten Module in anderen Projekten gelegt. Module, wie z.B. die im folgenden vorgestellte Kommunikationskapsel oder die Benutzungsoberfläche, wurden erprobt und bieten in diesen speziellen Funktionsbereichen eine für viele Anwendungen vollkommen ausreichende Funktionalität. Die natürlich primär auf das bestimmte Projekt 'MobileNotes' abgestimmten Lösungen erhielten eine offene Struktur, und anwendungsspezifische Lösungen traten oftmals zu Gunsten offener und allgemein verwendbarer Lösungen zurück.

[57] sogenannte Makefiles

4.2 MobileNotes-Server

Der Wunsch die beiden Betriebssysteme IBM OS/2 2.x/3.0 und MS Windows NT 3.5x zu unterstützen, bedingte die vollständige Neuimplementation des MobileNotes-Servers. Der auf Visual Basic 3.0 und somit auf 16 Bit Windows fixierte Prototyp 'AvALoN' wurde zu Gunsten einer plattformübergreifenden Version in der Programmiersprache C ersetzt. Vormals genutzte Komponenten, wie z.B. das Visual Basic Control 'PDQCom' und die 'MACROWARE.DLL', mußten durch Eigenentwicklungen oder durch andere Produkte ersetzt werden.

Auf beiden Plattformen wurde das Serverprogramm als Textapplikation realisiert, die wenig Interaktionsmöglichkeiten bietet, für dieses Projekt aber vollständig ausreicht. Einstellungsmöglichkeiten sind während des Programmablaufs nicht vorhanden. Jegliche Parametrisierung des Serverprogramms ist über die Initialisierungsdatenbank und durch Setzen einiger Kommandozeilenparameter vorzunehmen.

Die angestrebte Modularisierung des Servertasks kristallisierte sich in folgenden Modulen aus:

- Hauptprogramm (AVASPROG, Ersteller: Bernd Altmiks)
  Steuerung des MobileNotes-Servers. Initiiert die Entgegennahme von Anrufen und führt deren Behandlung durch.
- Modemkommunikation (AVASCOM, Ersteller: Nico Dirks)
  Kapselung der Modemkommunikation und der damit verbundenen Betriebssystemabhängigkeiten vor dem Hauptprogramm.
- Notes-Datenbankzugriffe (AVASNOTE, Ersteller: Nico Dirks)
  Kapselung der Notes-Datenbankzugriffe, um häufig verwendete Zugriffsaktionen einfacher, übersichtlicher und sicherer zu nutzen.
- Verschlüsselung (AVABDES, Ersteller: Bernd Altmiks)
  Verschlüsselung bzw. Entschlüsselung der übertragenen Daten, um Lauschangriffen vorzubeugen.
- Fehlererkennung (AVABCRC, Ersteller: Nico Dirks)
  Ersatz für die eventuell im nicht Modem vorhandene Fehlererkennung.

Deren Zusammenspiel gestaltet sich wie folgt:

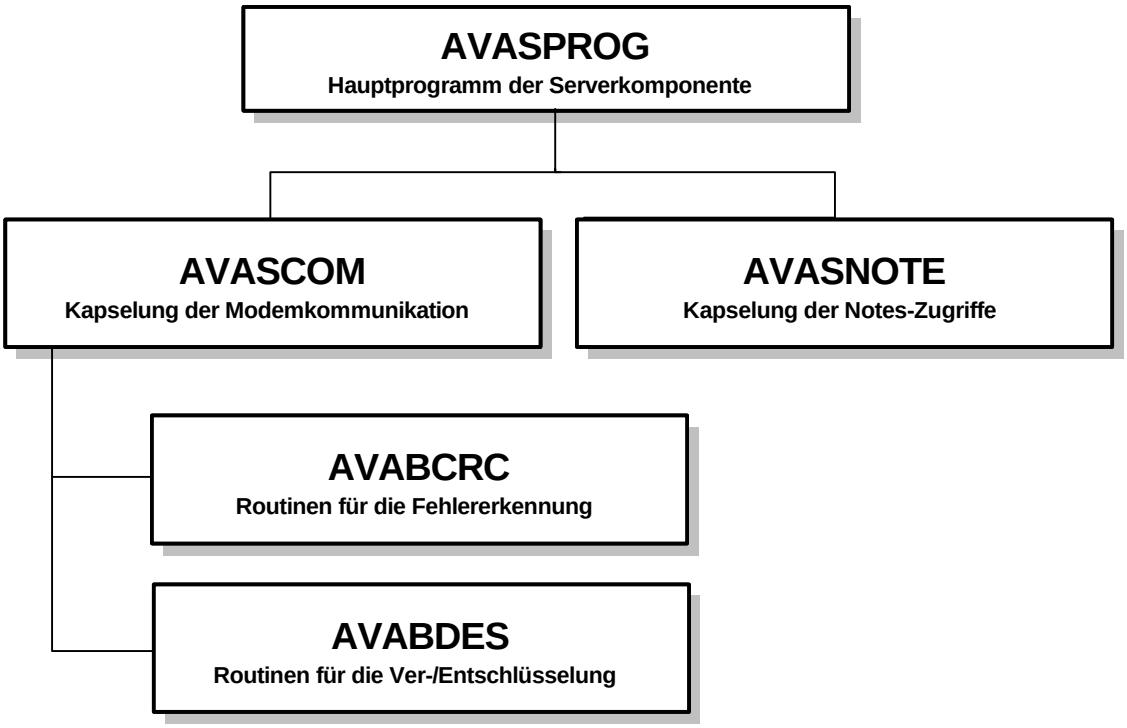

Abbildung 7: Modularisierung der Serverkomponente

Im folgenden werden die von Nico Dirks erstellten Module näher betrachtet. Eine nähere Beschreibung der Serverkomponente ist in [Altmiks 1996] enthalten.

4.2.1 Organisation des Zugriffs auf Notes-Datenbanken

Wurde beim Prototype 'AvALoN' noch das Notes-Programmier-API 'MACROWARE' (Application Programming Interface, kurz API) der Pavone Informationssysteme GmbH verwendet, so basiert die jetzige Realisation des Notes-Datenbankenzugriffs auf dem HiTest-API[58] der Lotus Development Corporation (eine Entwicklung des zu Lotus gehörigen Unternehmens Edge Research Inc.). Das HiTest-API, mittlerweile in der Version 2.2, ist in Versionen für Windows 3.1, Windows NT 3.5 und OS/2 2.x/3.0 als C-Entwicklungsumgebung und für Windows als Visual-Basic-Control verfügbar. Der höhere Abstraktionsgrad des HiTest-API's gegenüber dem Notes-API und die Unterstützung aller für das Projekt 'MobileNotes' relevanten Plattformen erlauben einen einfachen, sicheren und plattformunabhängigen Zugriff auf jegliche Notes-Datenbank.

Um das MobileNotes-Serverprogramm übersichtlich zu halten, oft benötigte HiTest-Befehlssequenzen in Funktionen zu aggregieren und eine gewisse Abstraktion von dem HiTest-API zu schaffen, wurde eine eigene Schnittstelle[59] entwickelt, mit der die HiTest-Programmierung von dem MobileNotes-Serverprogramm abgekapselt werden konnte. Diese Schnittstelle, im folgenden auch Kapsel genannt, aggregiert in einzelnen Funktionen eine Vielzahl von HiTest- und somit Notes-API-Funktionen. Die Aggregation geschah vor dem Hintergrund der konkreten MobileNotes-Implementation und spiegelt somit die spezifischen Bedürfnisse in diesem Projekt wider, kann jedoch für andere Projekte sehr nutzbringend eingesetzt werden.

Das HiTest C-API 2.2

Das HiTest C-API Version 2.2 beinhaltet neue bzw. erweiterte Möglichkeiten der Fehlererkennung und der impliziten Datentypkonvertierung, so daß viele Erschwernisse der direkten Notes-API-Programmierung (z.B. die fatalen Auswirkungen einer fehlerhaften Parametrisierung und die oftmals notwendigen explizite Datentypkonvertierungen) vor dem HiTest-Nutzer verborgen bleiben.

---

[58] siehe [Edge 1995]

[59] siehe Anhang B

Im folgenden ein kurzer Auszug der von Lotus genannten Vorteile gegenüber der direkten Notes-API-Programmierung:[60]

- Abfangen von Situationen und Operationen, welche Systemabstürze nach sich ziehen würden
- Abstraktion von Dokumentenmengen durch das Konstrukt des Index
- Einfachere und abstraktere Browsing-Funktionalitäten
- Implizite Datentypkonvertierungen
- Organisation der Funktionen in funktionalen Gruppen, welche die Übersichtlichk verbessern

Die Funktionen des API's sind in Gruppen mit identischem Wirkungskreis organisiert, so existieren z.B. die Gruppen Session (Sitzung), Item (Feld) und Index (allgemeine Form der Dokumentenmenge, z.B. als Resultat einer Volltextsuche oder eine Ansicht). Diese Gruppierung vereinfacht die Nutzung der Funktionen, stellt aber nur einen kleinen Schritt hin zur Objektorientierung dar. Situationsabhängig können Funktionen aus diesen Gruppen verwendet werden, so kann z.B. erst dann eine Ansicht geöffnet werden, nachdem eine Datenbank geöffnet wurde.

Das HiTest-API implementiert mit dieser von Notes vorgegebenen Vorgehensweise die Arbeit mit Kontexten. Ein Kontext definiert sich durch den aktuellen Zustand des HiTest-API's und legt somit die Menge der erlaubten Funktionen fest. Jeder Kontextwechsel impliziert automatisch eine Änderung dieser Menge. Durch die Verwendung von Handles und Identifiers können mehrere Kontexte parallel mit dem API verwendet werden. So ist es z.B. möglich, zwei Datenbanken geöffnet zu halten, in der ersten eine Ansicht zu öffnen und das letzte Dokument einzulesen um es sofort wieder in der zweiten Datenbank zu speichern. Die Kontextübergänge bei der Nutzung des HiTest-API lassen sich durch folgende Graphik auszugsweise darstellen, wobei jedes Rechteck einen Kontext symbolisiert und die inneren Rechtecke bzw. Kontexte nur von dem nächst äußeren Kontext erreicht werden können. Dies gilt natürlich auch für den umgekehrten Weg.

[60] siehe [Edge 1995]

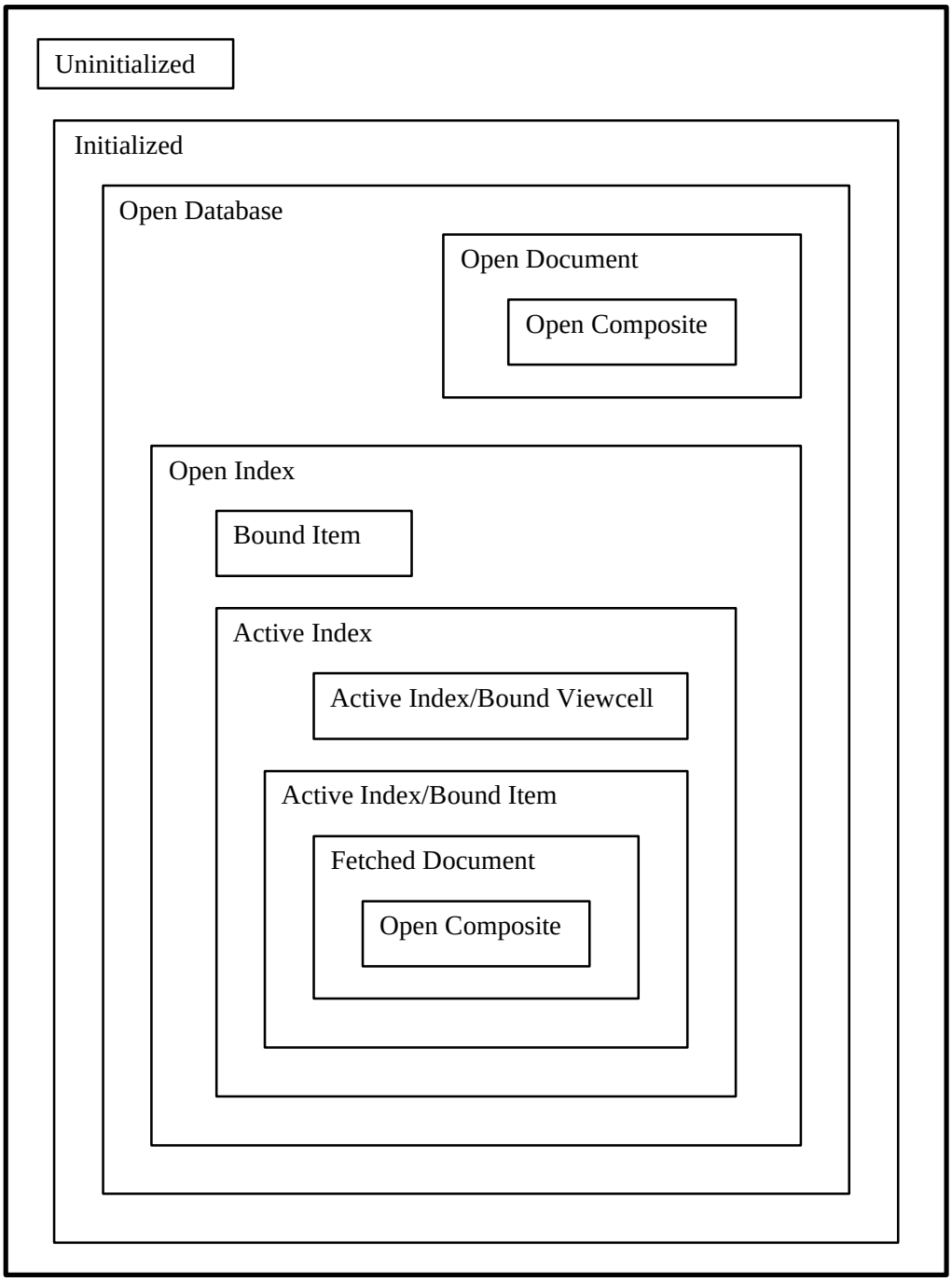

Abbildung 8: Kontexte des HiTest-APIs[61]

Die Laufzeitumgebung der HiTest-C-Entwicklungsumgebung besteht aus einer plattformabhängigen DLL (Dynamic Link Library), welche, wie die mit dem Notes-API erstellten Programme, auf die DLL's einer vorhandenen Notes-Workstation (bzw. Server) Installation zurückgreift. Ohne die vorherige Installation einer Notes-Workstation bzw. eines Notes-Servers ist die HiTest-DLL nicht lauffähig. Unter Windows NT wird momentan die Installation eines Notes-Servers vorausgesetzt, da keine Workstation-Version speziell für Windows NT verfügbar ist (unter Windows NT muß die Windows 3.1 16 Bit Version genutzt werden).

[61] [Edge 1995]

Die 32 Bit HiTest-Version kann nur mit den 32 Bit Komponenten des Notes Servers für Windows NT kooperieren.[62] Findet die HiTest-DLL diese Bedingungen vor, so kann mit ihr gearbeitet werden. Sie bietet, wie der Notes-Client, die Möglichkeit lokale Datenbanken zu nutzen oder über einen Notes-Server auf Datenbanken zuzugreifen. Die Nutzung lokal vorhandener Datenbanken (es ist unter gewissen Umständen möglich auch lokale Datenbanken mit einem Notes-Server zu nutzen, z.B. durch einen lokal installierten Server, doch gehe ich hier von lokaler Nutzung ohne Notes-Server aus) muß allerdings mit der Notes inhärenten Einschränkung leben, daß eine Datenbank zu einem Zeitpunkt nur einmal geöffnet sein kann. Erst ein Notes-Server versteht es, mehrere Zugriffe auf eine Datenbank zu koordinieren und somit zu ermöglichen.

Die Notes-Kapsel

Die Kapsel, im Projekt mit AVASNOTE bezeichnet, bildet eine hoch aggregierte Schicht zwischen dem MobileNotes-Serverprogramm und der konkreten Notes-Programmierschnittstelle. Wie oben angesprochen, wird z.Zt. die HiTest C-Entwicklungsumgebung als Notes-Programmierschnittstelle verwendet. Sie kann aber in zukünftigen Versionen, Umbau der Kapsel genügt, durch Alternativprodukte[63] ersetzt werden. Ein Instruktionssatz von 17 C-Funktionen erlaubt es der Kapsel, sämtliche von dem MobileNotes-Serverprogramm benötigten Notes-Zugriffe abzubilden. Die Implementation der Kapsel in einem separaten Modul (abhängig vom Betriebssystem werden DLL's oder statische Objekte erzeugt) erlaubte den MobileNotes-Entwicklern die Nutzung derselben, ohne mit deren Innenleben belastet zu werden. Diese modulare Vorgehensweise, wie sie z.B. auch bei der Kommunikationsschnittstelle des Serverprogramms vorzufinden ist, ermöglichte es den Programmierern, unter Wahrung der definierten Schnittstellen (C-Funktionsaufrufe) und Funktionalitäten, die Art und Weise der Implementation selbständig zu bestimmen.

---

[62] Die Kooperation von 32 Bit mit 16 Bit Komponenten ist unter Windows NT und Windows 95 ein sehr komplexes Thema, von dem selbst Microsoft nachhaltig abrät

[63] z.B. MACROWARE der Pavone Informationssysteme GmbH oder das Notes-API der Lotus Development Corporation

Die Schnittstellen der Kapsel lassen sich wie die des HiTest C-API's in Funktionsgruppen mit ähnlichem Wirkungskreis unterteilen, es existieren die folgenden Gruppen:

	Prefix	Umfang	Wirkungskreis
Datenbankfunktionen	usAVASNotes_Db...	2 Funktionen	Öffnen und Schließen von Datenbanken
Ansichtfunktionen	usAVASNotes_View...	2 Funktionen	Öffnen von Ansichten und zugehörige Informationen einholen
Dokumentfunktionen	usAVASNotes_Doc...	9 Funktionen	Navigation in Dokumentenmengen, Löschen, Erzeugen von Dokumenten, Informationen zu Dokumenten einholen
Feldfunktionen	usAVASNotes_Field...	3 Funktionen	Feldinhalt/-länge auslesen, Feldinhalt setzen
Mailfunktionen	usAVASNotes_Mail...	1 Funktion	Nachricht erstellen

Tabelle 2: Funktionsgruppen der Notes-Kapsel

Im Vergleich zu den anderen Notes-Programmierschnittstellen stellt sich diese Schnittstelle als klein und handlich, aber für diese bestimmte Problematik durchaus ausreichend dar. Um einen Eindruck von der Mächtigkeit anderer Schnittstellen zu bekommen ein kurzer Blick auf die Anzahl der dort angebotenen Funktionen:

Notes-C-API Version 3.2	HiTest-C-API Version 2.1	MACROWARE-C-API Version 1.0
ca. 352 Funktionen	ca. 175 Funktionen	ca. 82 Funktionen

Tabelle 3: Vergleich der Notes-Programmierschnittstellen

Verbirgt sich das Kontextkonzept bei dem HiTest C-API überwiegend im Hintergrund und tritt nicht permanent in das Bewußtsein des Benutzers, so basiert die Kapsel vollständig auf dem Kontextkonzept und ruft dies fortwährend in das Bewußtsein des Benutzers. Die erfolgreiche Öffnung einer Datenbank kreiert einen neuen Kontext innerhalb der Notes-Kapsel und liefert als Ergebnis einen Kontextidentifier (natürliche Zahl), mit dem in Zukunft alle Zugriffe auf diesen Kontext parametrisiert werden müssen. Jede weitere Datenbanköffnung erzeugt einen weiteren Kontext, der wiederum über einen eigenen Kontextidentifier referenziert werden kann. Diese Kontexte bestehen so lange, bis die zugehörigen Datenbanken geschlossen werden. Kontextabhängig sind verschiedene Funktionsaufrufe zulässig, so ist z.B. erst dann Navigation in einer Ansicht möglich, nachdem eine Ansicht geöffnet wurde. Die folgende Graphik gibt einen Überblick über die möglichen Kontexte, Kontextübergänge und die innerhalb eines Kontextes erlaubten Funktionen.

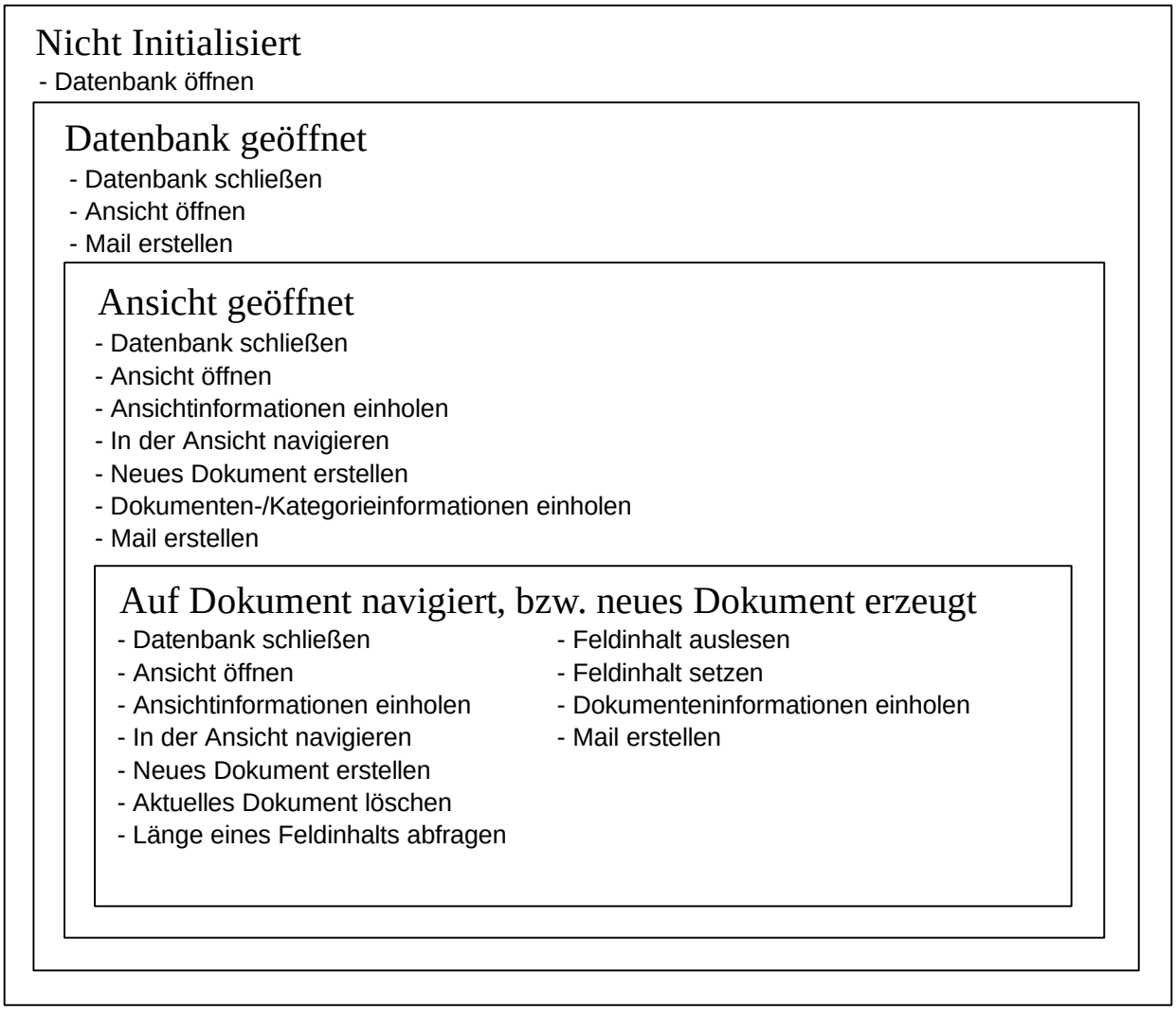

Abbildung 9: Kontexte der Notes-Kapsel

Da ein Kontext intern aus folgenden Elementen besteht:

- Verweis auf geöffnete Datenbank
- Verweis auf geöffnete Ansicht (falls Ansicht geöffnet wurde)
- Aktuelle Position in der geöffneten Ansicht (falls Ansicht geöffnet wurde)

und jedes dieser Attribute nur einfach ausgeprägt ist, ist es innerhalb eines Kontextes nicht möglich, Felder verschiedener Dokumente gleichzeitig auszulesen oder mehrere Ansichten geöffnet zu halten. Um dieses zu erreichen, muß ein neuer Kontext durch erneutes Öffnen der Datenbank angelegt werden und die gewünschte Aktion auf dem neuen Kontext ausgeführt werden. In dieser Situation dramatisiert sich das oben angesprochene Problem der lokalen Datenbanken. Lokale Datenbanken können nur einfach geöffnet werden, jeder weitere Öffnungsversuch schlägt fehl.

Folgende Besonderheiten der Notes-Kapsel qualifizieren sie zur Nutzung im MobileNotes-Projekt:

- Explizites Öffnen von Dokumenten ist nicht notwendig, jeder Zugriff auf ein Dokument (z.B. Feldinhalt auslesen) öffnet das Dokument implizit
- Explizites Schließen von Dokumenten ist nicht notwendig, jede Veränderung der Position in der Ansicht, jede Neuöffnung einer Ansicht und jedes Schließen einer Datenbank schließt geöffnete Dokumente und fixiert somit die ggf. daran vorgenommenen Änderungen
- Es existiert keine Ansicht-Schließen Funktion, da eine geöffnete Ansicht durch ei erneutes Ansicht-Öffnen automatisch geschlossen wird und ein Schließen der Datenbank auch eine Schließung der Ansicht zur Folge hat
- Eine Initialisierung bzw. Terminierung der Notes-Schnittstellen ist nicht notwend (s. z.B. HiTest-Befehle HTInit und HTTerm), da dies implizit durchgeführt wird, wenn die erste Datenbank geöffnet bzw. die letzte geöffnete Datenbank geschlossen wird.

Abschließend bleibt zu bemerken, daß die Spezialisierung der Notes-Kapsel auf das MobileNotes-Projekt keinen wesentlichen Nachteil darstellt. Versuche, die Notes-Kapsel aus Makro-/Script-Sprachen heraus zu nutzen, verliefen erfolgreich. Einer Nutzung der Kapsel mit Visual Basic, Visual Basic for Applications, Rexx oder anderen Programmierwerkzeugen steht nichts entgegen. Die implementierte Funktionalität ist für viele Aufgaben vollkommen ausreichend und erlaubt eine wesentlich schnellere, sicherere und übersichtlichere Anwendungsentwicklung als die komplexen Alternativ-Notes-Schnittstellen. Natürlich ist vor jeder Nutzung der Notes-Kapsel eine Analyse durchzuführen, ob die geleistete Funktionalität ausreicht und zu bedenken, daß derzeit die HiTest-Laufzeitumgebung benötigt wird, um die Kapsel zu nutzen.

4.2.2 Abstraktion von der Modemkommunikation

Die Modemkommunikation ist ebenso wie die Notes-Kapsel als separates Modul implementiert worden, welches nur über definierte Schnittstellen[64] angesprochen werden kann. Derzeit existieren Versionen für OS/2 2.x/3.0 und Windows NT. Die unterschiedliche Behandlung der Kommunikation mit der seriellen Schnittstelle (Modems werden an der seriellen Schnittstelle angeschlossen und sind nur über dieselben erreichbar) unter OS/2 und Windows NT erfordert einige plattformspezifische Codesegmente in dem Modemkommunikationsmodul (im folgenden auch Modem-Kapsel genannt, da es von dem zu Grunde liegenden Betriebssystem abstrahiert). So behandelt Win32, das Windows NT Programmier-API, die seriellen Schnittstellen als Spezialdateien, auf welche die normalen Dateioperationen, wie z.B. ReadFile, WriteFile oder CloseFile, ausgeführt werden können. Darüber hinaus existieren Spezialfunktionen, welche Sonderaufgaben bezüglich der seriellen Schnittstellen wahrnehmen, z.B. Geschwindigkeit der Schnittstelle bestimmen, Anzahl der Datenbits, Stopbits und Paritybits definieren. Das Programmier-API zu OS/2 behandelt serielle Schnittstellen dagegen als Spezialgeräte, welche mit einem eigenen, sehr spezifischen Instruktionssatz programmiert werden müssen.

Aufgaben

Die Hauptaufgaben des Moduls liegen in der Kapselung plattformspezifischer Codesegmente, der Verbesserung der Übersichtlichkeit im MobileNotes-Hauptprogramm und der Abstraktion von den verwendeten Übertragungs-, Datensicherungs- und Verschlüsselungsverfahren. Diese Ziele resultieren in einem minimalen Funktionssatz, der folgende Operationen umfaßt:

- Verbindung aufbauen, indem eingehende Anrufe entgegengenommen werden
- Daten empfangen
- Daten senden
- Verbindung abbrechen
- Abfragen, ob Verbindung noch besteht

Ablauf

Grundsätzlich verläuft eine MobileNotes-Server - MobileNotes-Client Kommunikation nach folgendem Muster:

1. Server wartet auf eingehenden Anruf
2. Client wählt Server an
3. Server nimmt eingehenden Anruf entgegen
4. .... Kommunikation ....
5. Verbindung wird vom Client abgebrochen

[64] siehe Anhang C

Da in Zukunft ein MobileNotes-Server mehrere Verbindungen parallel aufrecht erhalten und bedienen soll, bietet die Kommunikationskapsel die Möglichkeit, Threads abzuspalten, die auf eingehende Anrufe warten und bei Verbindungsaufbau eine benutzerdefinierte Funktion aufrufen. Mit diesem Mechanismus ist es einem Hauptprogramm möglich, für mehrere angeschlossene Modems und den ggf. damit verbundenen Clients jeweils einen spezifischen Thread zu starten.

Dies geschieht folgendermaßen: Das MobileNotes-Serverprogramm ruft für eine definierte serielle Schnittstelle die Polling-Routine[65] der Kommunikationskapsel auf, als Parameter werden dieser Polling-Routine ein Modeminitialisierungsstring und ein Funktionspointer mitgegeben. Die Polling-Routine öffnet und initialisiert die serielle Schnittstelle, initialisiert das angehängte Modem mit dem übergebenen Initialisierungsstring und instruiert das Modem, alle eingehenden Telephonate nach dem ersten Klingelzeichen zu übernehmen. Nach diesen Aktionen spaltet die Polling-Routine einen Thread ab und gibt die Kontrolle an das Serverprogramm zurück. Das Serverprogramm kann dann dieses Procedere für weitere serielle Schnittstellen durchführen, während der vorher abgespaltene Thread auf eingehende Telephonanrufe wartet, sie entgegennimmt und bei erfolgreichem Verbindungsaufbau die vom Serverprogramm spezifizierte Funktion (den referenzierten Funktionspointer) aufruft. Diese Funktionsweise erlaubt die einfache Abspaltung von Threads, welche einzelne Anrufe entgegennehmen können und die weitere Behandlung dieser Anrufe einleiten. Durch die Kapselung dieser Thread-Erzeugung in der Kommunikationskapsel bleibt dem Serverprogramm die konkrete Implementation verborgen. (Threaderzeugung ist systemnah und betriebssystemabhängig)

Momentan verbietet sich die Nutzung dieser Möglichkeiten, da die Notes-Umgebung, unabhängig vom konkret verwendeten API, nicht aus mehreren Threads angesprochen werden kann. In der Notes Version 4.x wird diese Fähigkeit enthalten sein, und dieser Mißstand kann behoben werden. Aus diesem Grunde wird z.Zt. eine abgespeckte Variante der oben angegebenen Vorgehensweise verwendet. Anstatt einen Thread abzuspalten, der ein eingehendes Telephonat behandelt, wird jetzt eine Unterroutine genutzt, die ein eingehendes Telephonat behandelt. Dies hat zur Folge, daß nur ein Thread im Serverprogramm besteht und nicht mehrere serielle Schnittstellen parallel behandelt werden können.

Kommunikationsprotokoll

Wurde eine Client-Server-Verbindung aufgebaut, können Daten in beide Richtungen übertragen werden. Es gibt dabei zu jedem Zeitpunkt genau einen Sender und einen Empfänger. Zwecks Datensicherung und Verschlüsselung wird ein Protokoll verwendet, welches die optionale Nutzung von Fehlererkennungsmechanismen und Datenverschlüsselung erlaubt. Z.Zt. wird für die Fehlererkennung der CRC[66] auf 16 Bit-Basis und für die Datenverschlüsselung der DES[67]

---

[65] engl. to poll - to receive - ‘Empfangen’, im hier verwendeten Sinne 'Anrufentgegennahmefunktion'

[66] Cyclic Redundancy Check nach CCITT, siehe nächstes Kapitel

[67] Data Encryption Standard der IBM, siehe [Altmiks 1996]

verwendet. Zukünftig ist die Nutzung anderer ggf. sogar fehlerkorrigierender Verfahren denkbar. Ähnlich der Netzwerkkommunikation kann der Datenfluß in einem Schichtenmodell dargestellt werden. Die Daten durchqueren dabei beim Sender die gleichen Kommunikationsschichten (von Anwendungsschicht an Modem), wie beim Empfänger, allerdings bewegen sie sich beim Empfänger in entgegengesetzter Richtung (von Modem an Anwendungsschicht).

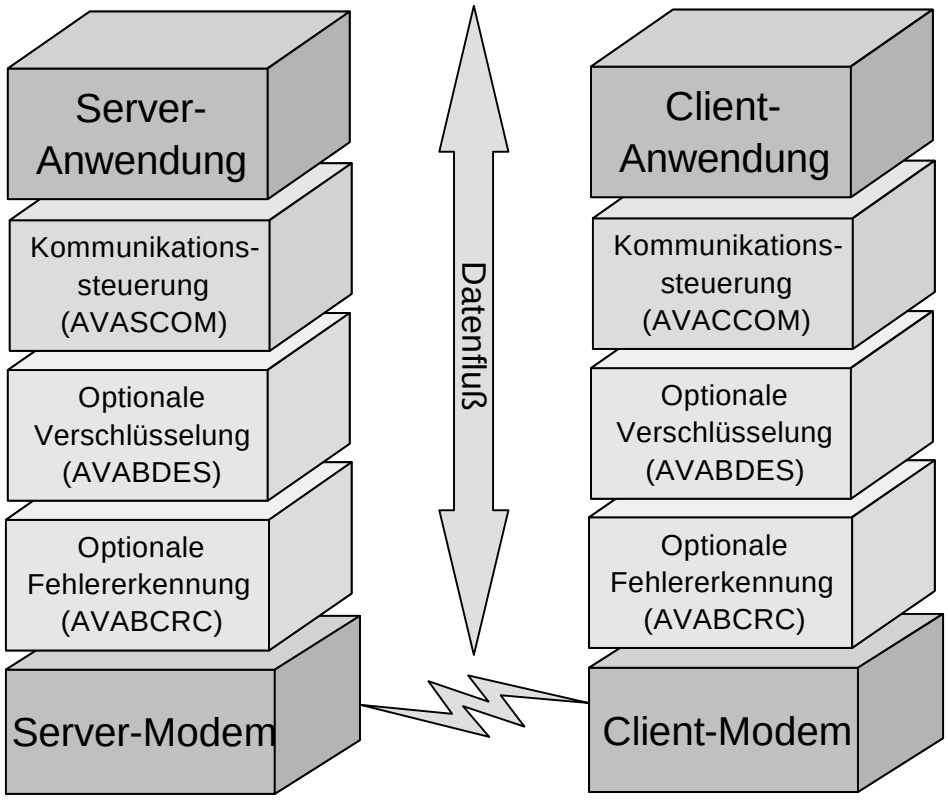

Abbildung 10: Schichtenmodell der Kommunikation

Die Daten werden bei dieser Kommunikationsform in Datenpakete vorgegebener Struktur aufgeteilt, die jeweils vom Empfänger positiv bestätigt werden müssen.

	Anzahl Bytes	Erklärung
SYNC	1	Synchronisationsbyte leitet immer ein Datenpaket ein. (Wert: 0x01)
Paketnummer, Verschlüsselungsflag, Fehlererkennungsflag	1	Paketnummer (Bits: 0-5) Verschlüsselungsflag (Bit: 6) Fehlererkennungsflag (Bit: 7)
Anzahl Datenbytes	1	Anzahl der im Datenteil folgenden Datenbytes
Daten	max. 255	Zu übertragende Daten
Fülldaten	max. 7	ggf. notwendig gewordene Fülldaten, die z.B. für die Verschlüsselung oder Fehlererkennung notwendig geworden sind
Fehlererkennungscode	ggf. 2	ggf. Fehlererkennungscode, falls Fehlererkennung eingeschaltet ist

Tabelle 4: Aufbau eines Kommunikationspaketes

Die Kommunikation mit Datenpaketen fester Struktur beeinflußt die zu übertragenen Inhalte nicht, bedingt aber in Fällen großer Sendeaufträge die Aufteilung der Sendedaten auf mehrere Pakete, da pro Paket maximal 255 Bytes verschickt werden können. Diese Aufteilung bleibt allerdings allen auf der Kommunikation aufsetzenden Schichten verborgen, da empfangene Pakete automatisch wieder zusammengesetzt und nicht als 255 Byte-Einheiten weitergereicht werden.

Zu den Paketelementen im Einzelnen:

1) SYNC

   Dieses Synchronisationsbyte leitet jeden Sendeauftrag ein, hat den Dezimalwert 1 und dient der Empfangsseite als Erkennungszeichen für ein eingehendes Datenpaket.

2) Paketnummer

   Der Sender markiert jedes zu verschickende Paket mit einer fortlaufenden Paketnummer, die dem Empfänger die Unterscheidung zwischen aktuellen und veralteten Paketen erlaubt. (Wird der Maximalwert 63 erreicht, beginnt die Zählweise wieder bei 0.)

3) Verschlüsselungsflag

   Dieses Flag dient als Indikator für eine Datenteilverschlüsselung und weist darauf hin, daß ggf. der Datenteil vergrößert wurde. (Z.B. kann der DES-Verschlüsselungalgorithmus nur Vielfache von 8 Byte verschlüsseln, liegt kein Vielfaches von 8 Bytes vor, so müssen zusätzliche Fülldaten an den Datenteil gehängt werden.)

4) Fehlererkennungsflag

   Dieses Flag dient als Indikator für eine Nutzung der Fehlererkennung und weist darauf hin, daß ggf. der Datenteil vergrößert wurde. (Z.B. kann der CRC-Fehlererkennungsalgorithmus nur Vielfache von 2 Byte verschlüsseln, liegt kein Vielfaches von 2 Bytes vor, so müssen zusätzliche Fülldaten an den Datenteil gehängt werden. Der CRC-Fehlererkennungsalgorithmus benötigt ferner zwei zusätzliche Bytes in denen die berechnete Prüfsumme übertragen wird.)

5) Anzahl Datenbytes

   In diesem Byte wird die Anzahl folgender Datenbytes festgehalten. Fülldaten, wie z.B. für die Fehlererkennung und Verschlüsselung benötigt, werden hier nicht mitgezählt, damit der Empfänger die Möglichkeit besitzt eventuell angehängte Fülldaten auszusondern. (Würden die Fülldaten mitgezählt, bestände keine Möglichkeit die reale Länge des Datenteils zu ermitteln.)

6) Daten

   Es folgen die zu übertragenen Daten, ggf. verschlüsselt, aber ansonsten unverändert.

7) Fülldaten

   Ggf. notwendig gewordene Fülldaten folgen direkt dem Datenteil.

8) Fehlererkennungscode

   Wurde mit dem Fehlererkennungsflag die Fehlererkennung eingeschaltet, so folgen hinter den Daten und den ggf. existierenden Fülldaten die Prüfsummenbytes. (Beim CRC 2 Bytes.)

Der Transfer von Datenpaketen ist eingebettet in ein Kommunikationsverfahren, welches weitestgehend Kommunikationsfehler wie Paketverluste, Paketverstümmelung, verzögerte Übertragungen, Leitungsechos, konkurrierendes Empfangen und weitere Fehler erkennt und behebt. Lautete der Arbeitstitel dieser Protokollimplementation noch 'Master-Master-Protokoll', so ist der Begriff 'Master-Slave'-Protokoll eher zutreffend.

Prinzipiell verläuft das Protokoll in folgender Art und Weise:

1) Der Empfänger schickt dem Sender eine Sendeaufforderung und geht in den Empfangsmodus.

2) Der Sender verschickt, nach Erhalt der Aufforderung, ein Datenpaket.

3) Der Empfänger empfängt die angeforderten Daten .

4) Der Empfänger untersucht das ggf. empfangene Datenpaket auf Fehler.
   - Wurde kein Paket empfangen, so wird bei (1) neu aufgesetzt.
   - Wurde ein defektes Paket empfangen, so wird eine negative Bestätigung geschickt und bei (3) fortgefahren.
   - Wurde ein altes Paket empfangen (d.h. Paketnummer ist schon bekannt), so wird die alte Bestätigung geschickt und bei (3) fortgefahren.
   - Wurde ein intaktes und aktuelles Paket empfangen, so wird eine positive Bestätigung geschickt und bei (5) fortgefahren.

5) Verarbeitung des empfangenen Paketes.

Eine detaillierte Ablaufbeschreibung ist in [Sievers 1996] enthalten.

4.2.3 Fehlererkennung

Im Datenaustausch über unsichere Medien, wie z.B. Telephonleitung, Funk und Disketten treten immer wieder Übertragungsfehler auf. Diese Fehler zu vermeiden hat sich die Kodierungstheorie zur Aufgabe gesetzt und liefert im wesentlichen zwei Kategorien von Fehlerbehandlungsstrategien.

Es kann grob zwischen der Fehlererkennung und der Fehlerkorrektur unterschieden werden. Die Fehlererkorrektur setzt allerdings eine vorhergehende Fehlererkennung voraus. Doch nun im Einzelnen:[68]

Forward Error Correction (FEC)

Diese Kategorie setzt die Anreicherung von Übertragungsdaten mit Zusatzdaten voraus, welche die Erkennung und die Behebung von Fehlern erlauben. Diese Kategorie erlaubt somit Simplex-Verbindungen zwischen Sender und Empfänger, d.h. fehlerhafte Daten werden nicht erneut beim Sender angefordert, sondern beim Empfänger automatisch korrigiert. Die Wahl dieses Verfahrens ist somit in Bereichen mit Einwege-Verbindungen sinnvoll. Auch Disketten, die nicht mehr im direkten Kontakt mit dem Sender (bzw. Erzeuger) stehen, sollten mit FEC-Kodes gesichert sein. Praktisch werden diese Verfahren für die Sicherung von Einwege-Funk-Verbindungen, Festplatten, Disketten, CD's und vielen weiteren Feldern genutzt. Oftmals ist es auch sinnvoll, bei Duplex-Verbindungen FEC-Kodes zu verwenden, um entweder Übertragungskosten oder Übertragungszeit zu minimieren.

Automatic Repeat Request (ARQ)

Diese Kategorie erlaubt nur die Fehlererkennung, die Fehlerbehebung muß über den Umweg des Senders erfolgen, d.h. erkannte Fehler bewirken eine Neuanforderung der Daten beim Sender. Hierfür wird eine Duplex-Verbindung zwischen Sender und Empfänger benötigt, welche es dem Empfänger ermöglicht, im Fehlerfall die Daten erneut anzufordern. Einsatzfelder für diese Kodes bieten sich im Bereich von lokalen Netzwerken und der Datenfernübertragung.

Grundsätzlich muß bedacht werden, welche Art von Verbindung zur Verfügung steht und wie die Kosten für eine ggf. erneute Übertragung bewertet werden.

In der vorliegenden Implementation der Modemkommunikation, welche auf der Server- als auch auf der Clientseite vorzufinden ist, wurde ein einfaches fehlerkorrigierendes Verfahren verwendet. Dieses Verfahren, Cyclic-Redundancy-Check (kurz CRC) nach CCITT, arbeitet auf beliebigen Blockgrößen und wird in vielen Bereichen der Datenübertragung eingesetzt. Steigende Blockgröße führen dabei allerdings zu steigenden Restfehlerraten.

Dieses Verfahren wird in der vorliegenden Implementation[69] nur zur Fehlererkennung genutzt, d.h. erkannte Fehler führen zur Neuanforderung des fehlerhaften Datenblocks beim Sender. Die

---

[68] siehe [Wong, Britland 1995]

theoretisch mögliche Fehlerkorrektur wurde, wie auch in vielen anderen Produkten (z.B. in GSM-Netzen und Modacom), aufgrund der hohen benötigten Rechenleistung und der komplexen Implementation nicht realisiert.[70]

In der Implementation nimmt dieses Modul neben dem Verschlüsselungsmodul eine Sonderstellung ein, die Fehlerkorrektur und die Verschlüsselung werden sowohl auf Server- als auch auf Clientseite benötigt. Die diesbezüglichen Module sind daher derart programmiert worden, daß eine Übersetzung auf beiden Plattformen möglich ist. Die Fehlererkennung wurde zudem noch in Assembler programmiert, um die Geschwindigkeit zu optimieren.

[69] siehe Anhang D

[70] siehe [Keller 1986]

4.3 Exemplarische Client-Implementation auf dem Philips Screen Phone P100

Exemplarisch, auf Basis des Screen Phones P100 der Philips Home Services, wurde eine konkrete MobileNotes-Client-Implementation geschaffen. Auf konkrete Entwicklungen aus dem 'AvALoN'-Projekt wurde dabei nicht zurückgegriffen, alle Komponenten sind von Grund auf neu entwickelt und implementiert worden. Durch die Modularisierung und die Programmierung in C wurden weitestgehend alle Systemabhängigkeiten beseitigt.

Die angestrebte Modularisierung des Clientprogramms kristallisierte sich in folgenden Modulen aus:

- Hauptprogramm (AVACMAIN, Ersteller: Nico Dirks)

  Steuerung des MobileNotes-Clients, bietet dem Benutzer die gesamte implementierte Funktionalität an und greift dabei auf die Funktionalität der anderen Client-Module zurück.

- Modemkommunikation (AVACCOM und AVACPROT, Ersteller: Dirk Sievers)

  Kapselung der Modemkommunikation und der damit verbundenen Betriebssystemabhängigkeiten vor dem Hauptprogramm.

- Datenhaltung (AVACDAL, Ersteller: Dirk Sievers)

  Bereitstellungen von Serviceroutinen zur Kapselung des lokalen und Servergestützen Betriebs. Dieses Modul bietet nach außen identische Schnittstellen für die Arbeit mit Notes-Daten, unterscheidet aber intern zwischen lokaler und servergestütztem Betrieb und führt dementsprechende Funktionen aus.

- Benutzungsoberfläche (AVACDISP, Ersteller: Nico Dirks)

  Servicefunktionen zur Darstellung einer dem jeweiligen Style-Guide genügenden Benutzungsoberfläche und Behandlung der Benutzerinteraktionen, greift hierzu auf das folgende Modul zurück.

- Benutzereingaben (AVACKEY, Ersteller: Nico Dirks)

  Nimmt die Benutzereingaben entgegen und reicht sie in vordefinierter Form an das aufrufende Modul zurück.

- Speicherverwaltung (AVACMEM, Ersteller: Nico Dirks)

  Kapselt systembedingte Besonderheiten vor den anderen Modulen und dient somit als deren Grundlage.

- Verschlüsselung (AVABDES, Ersteller: Bernd Altmiks)

  Verschlüsselung bzw. Entschlüsselung der übertragenen Daten, um Lauschangriffen vorzubeugen.

- Fehlererkennung (AVABCRC, Ersteller: Nico Dirks)
  Ersatz für die eventuell nicht im Modem vorhandene Fehlererkennung.

Abgebildet auf eine Hierarchie ergibt sich folgendes Bild:

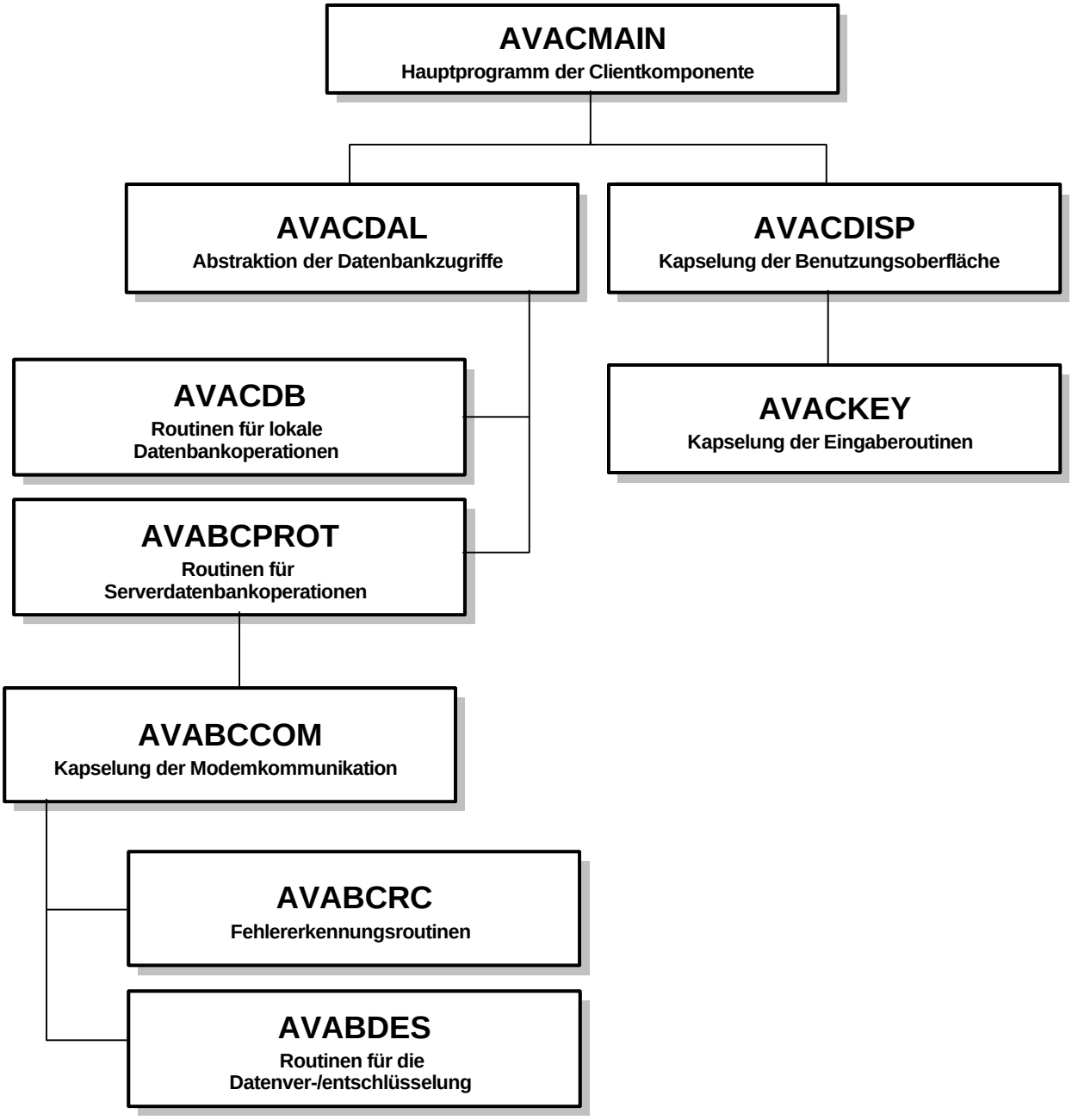

Abbildung 11: Modularisierung der Clientkomponente

Im folgenden werden die vom Verfasser dieser Arbeit erstellten Module näher betrachtet, die Implementation der Fehlererkennung wurde dabei schon in der Serverdokumentation angesprochen. Nähere Beschreibungen der anderen Module sind in [Sievers 1996] bzw. für die Verschlüsselung in [Altmiks 1996] enthalten.

4.3.1 Graphische Benutzungsoberfläche

Die graphische Benutzungsoberfläche integriert Funktionen zur Bildschirmdarstellung und Benutzerinteraktion. Das 'Look and Feel' des MobileNotes-Clients wird durch dieses Modul definiert und kann somit auch in anderen Projekten verwendet werden. Modular, auf der Grundlage vorher definierter Schnittstellen programmiert abstrahiert die Benutzungsoberfläche von dem zu Grunde liegenden Frontend, dessen Darstellungs- und Interaktionsmöglichkeiten. Dadurch steht einer Portierung dieser Schnittstelle[71] oder einer Veränderung des 'Look and Feel' innerhalb dieses Moduls nichts entgegen.

Durch eine Bedarfsanalyse kristallisierte sich eine Menge von Funktionen heraus, die den konkreten Bedarf des MobileNotes-Client abdecken, aber auch für viele weitere Aufgaben ausreichen dürften.

Ermittelte Funktionen:

- Textdarstellungsfunktion

  Darstellung von Texten, die evtl. über eine Bildschirmseite hinausgehen und daher mit Scrollfunktionalität versehen werden müssen. Wird z.B. zur Anzeige von Hilfetexten benötigt.

- Menüfunktion

  Darstellung von Auswahlmenüs und Behandlung der zugehörigen Benutzerinteraktionen. Wird z.B. zur Darstellung der Dokumente innerhalb einer Datenbank verwendet.

- Eingabemaskenfunktion

  Erstellt aus vorgegebenen Feldbeschreibungen eine Eingabemaske und bietet sie dem Benutzer zur Interaktion an. Wird z.B. zur Erstellung von Antwortdokumenten verwendet.

- Funktion zum hervorgehobenen Darstellen einer Zeichenkette

  Vergleichbar mit einer Dialogbox werden Texte auf einem dunklen rechteckigen Hintergrund dargestellt

- Funktion zum hervorgehobenen Einlesen einer Zeichenkette

  Vom Erscheinungsbild vergleichbar mit der vorangegangenen Funktion erlaubt diese Funktion dem Benutzer noch zusätzlich ein Eingabefeld auszufüllen.

Diese Funktionen werden in den unterschiedlichsten Zusammenhängen verwendet und müssen situationsabhängig auf Benutzerinteraktionen reagieren können (z.B. kann der Druck auf eine Funktionstaste abhängig vom konkreten Menü oder gar Untermenü unterschiedlichste Aktionen

[71] siehe Anhang E

auslösen). Die daher benötigte Variabilität der Funktionen muß sich in den Schnittstellen der angebotenen Funktionen widerspiegeln.

Es wird ein Mechanismus verwendet, der es erlaubt, die konkrete Reaktion nicht fest in die Benutzeroberfläche kodieren zu müssen, sondern durch die aufrufende Funktion vorzugeben. Ein solcher Mechanismus basiert auf Call-Backs. Call-Backs sind Funktionen die von außen vorgegeben und in bestimmten Fällen aufgerufen werden. In der Programmiersprache C basieren Call-Backs auf Funktionspointern. Im vorliegenden Fall führt z.B. der Druck auf eine Funktionstaste zur Ausführung des zugehörigen Call-Backs, da die Interaktionsroutine nicht wissen kann, welche Funktion sich hinter einer Funktionstaste (z.B. 'Hangup') verbirgt, die Steuerroutine (d.h. die die Interaktionsroutine aufrufende Funktion) diese Information aber sehr wohl zur Verfügung stellen kann. Call-Backs werden z.B. bei der Interaktion mit den frei definierbaren Funktionstasten, einer Menüpunktauswahl und der Menüdarstellung intensiv verwendet. Sie tragen somit ihren Teil dazu bei, die Interaktionsroutinen unabhängig vom konkreten Verwendungszweck zu halten. Jegliche Programmablaufinformation bleibt außen vor und muß durch die aufrufenden Routinen mitgeliefert werden. Im Resultat bleiben sämtliche Funktionen der Benutzungsoberfläche von der konkreten Verwendung unberührt.

Die auf dem P100 implementierte graphische Benutzungsoberfläche orientiert sich in ihrem Look&Feel an dem 'User Interface Style Guide'[72] des P100. Dieser Style Guide sieht eine grundsätzliche Trennung des Bildschirms in drei horizontal unterteilte Bereiche vor, welche jeweils die Darstellung bestimmter Informationen übernehmen. Der obere Bildschirmabschnitt, der sogenannte 'top header area', umfaßt 3 Zeilen und dient der Darstellung aktueller Status-, Modus- und Zeitinformationen. Der mittlere Bildschirmabschnitt, der sogenannte 'middle work area', umfaßt 11 Zeilen und dient der Interaktion mit dem Benutzer. Die verbleibenden 2 Zeilen bilden den sogenannten 'bottom softlabel area' und dienen der Beschriftungen für die darunterliegenden Funktionstasten.

[72] siehe [Philips 1994b] u. [Philips 1994c]

Schematisch stellt sich ein solchermaßen aufgebauter Bildschirm wie folgt dar:

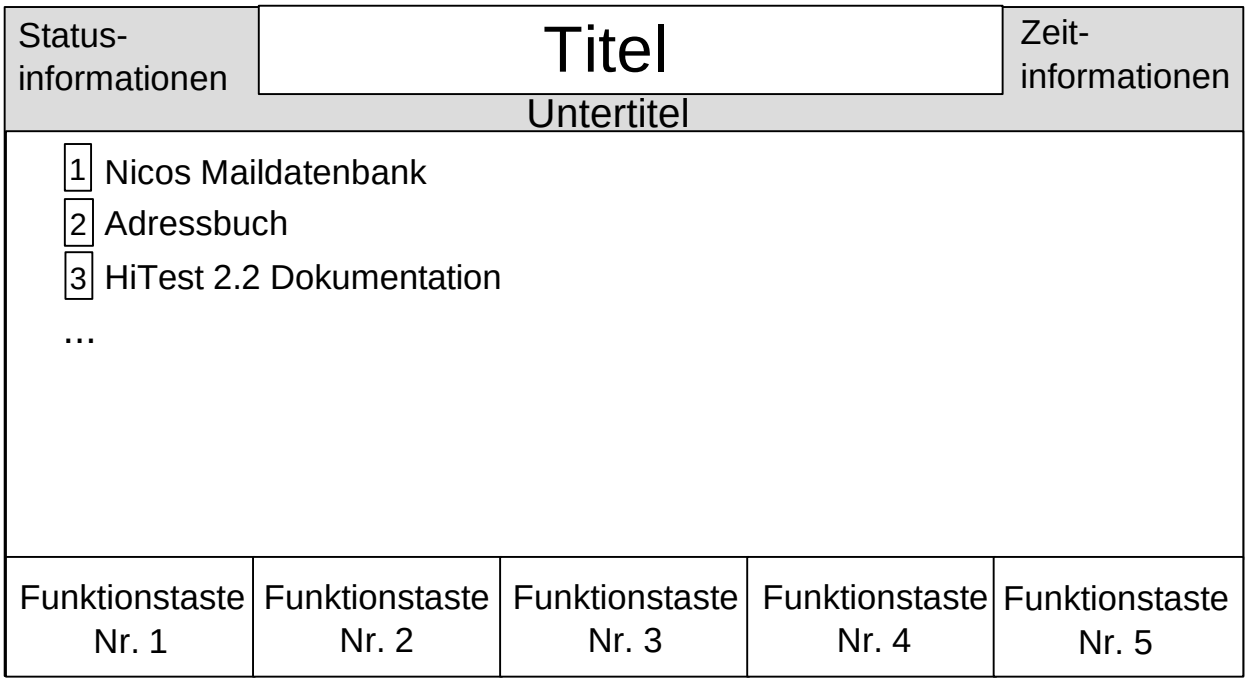

Abbildung 12: Exemplarischer Bildschirmaufbau des P100

Wurde ein Großteil der Funktionalität dieser Benutzungsschnittstelle durch die Verfolgung eines prozeduralen Konzeptes realisiert, so existieren doch Bereiche, die von besonderen Fähigkeiten des P100 Gebrauch machen. Die Ausgabe der Zeitinformationen und des Verbindungsstatus werden, ebenso wie der Bildschirmschoner[73], über einen Systemtaktgeber (Timer) angestoßen. In Abständen von 100ms wird vom POS eine Ausgaberoutine aufgerufen, welche die Ausgabe dieser Informationen übernimmt. Die Nutzung eines Timers für die Steuerung dieser Ausgaben entlastet das hier vorgestellte Modul von der fortwährenden Neuausgabe dieser Informationen. Somit kann auch während langwieriger Aktionen wie z.B. Verbindungsaufbau zum Server, in denen das Programm keinen Einfluß auf die Bildschirmausgaben nehmen kann, die zyklische Aktualisierung dieser Angaben gewährleistet werden.

Funktionalität

Der Nutzung der Benutzungsschnittstelle soll ein Aufruf der Initialisierungsfunktion vorangehen, die z.Zt. nur der oben angesprochene Timer einrichtet, aber in Zukunft oder auf anderen Frontends weitere Funktionalität beherbergen kann. Anschließend stehen dem Nutzer die oben angesprochenen Funktionen zur Verfügung, mit denen er alle notwendigen Interaktionen abwickeln kann.

[73] siehe Benutzereingabemodul, Kapitel 4.3.2.

Neben relativ einfachen Funktionen, die z.B. eine Textnachricht auf den Bildschirm bringen oder einen längeren Text durch Navigationsmöglichkeiten zugänglich machen, existieren auch anspruchsvollere Funktionen, von denen im folgenden zwei exemplarisch behandelt werden:

- Menüfunktion (usAVACDisp_Menu)

  Stellt auf dem Bildschirm ein Menü dar, inklusive der Titel- und Untertitelzeile, die Funktionstasten (bzw. die korrespondierenden Felder in der 'bottom softlabel area') werden über Funktionsparameter gesteuert mit Namen und Call-Backs belegt. Die darzustellenden Menüpunkte können direkt als Liste an die Funktion übergeben werden oder von dieser Funktion selbst über eine in den Funktionsparametern zu deklarierende Funktion angefordert werden. Die beiden Möglichkeiten sind alternativ und besitzen jeweils gewisse Vor- bzw. Nachteile. Die Listen-Variante besitzt den Vorteil, daß sie relativ einfach zu nutzen ist, doch die übergebene Liste ist statisch und kann sich neuen Gegebenheiten nicht anpassen. Sollen große Menüs dargestellt werden, z.B. eine Dokumentenansicht, so müssen vor dem Funktionsaufruf alle Menüpunkte vorliegen, auch wenn der Benutzer sofort den ersten Menüpunkt auswählt und die folgenden gar nicht einsieht. (Soll eine Ansicht bzw. Auswahl der Dokumente auf dem Server angeboten werden, so kann diese Aktion ziemlich schnell zur Geduldsprobe werden.)

  Die Call-Back-Variante bietet den Vorteil, daß sie dynamisch auf Benutzeranforderungen reagieren kann. Soll eine Dokumentenansicht eingeholt werden, so reicht es aus, so viele Dokumententitel einzuholen, bis der einsehbare Bereich gefüllt ist. Soll in der Ansicht navigiert werden, so müssen jeweils nur die wirklich notwendigen Dokumententitel eingelesen werden. Durch geschickte Programmierung der Call-Backs kann z.B. die Kommunikationslast zwischen Server und Client noch zusätzlich gesenkt werden. Statt jeden Dokumententitel neu vom Server anzufordern, werden einmal angeforderte Titel im Client zwischengespeichert, um zukünftige Anfragen direkt innerhalb des Clients befriedigen zu können.

  Als Nachteil erweist sich hier die wesentlich komplexere Implementation. Des weiteren kann der Programmierer auch die Art und Weise beeinflussen, mit der auf Menüpunktauswahl seitens des Benutzers reagiert wird. Im einfachsten Fall wird die Indexnummer des gewählten Menüpunktes als Funktionsergebnis zurückgeliefert. Soll die Menüfunktion nicht verlassen werden, da z.B. ein erneuter Menüaufbau zu viel Zeit kosten würde, kann auch hier auf eine Call-Back-Variante zurückgegriffen werden. Durch die Deklaration eines bestimmten Parameters wird nach Auswahl eines gültigen Menüpunktes eine Funktion (Call-Back) aufgerufen, welche mit der Indexnummer parametrisiert wird.

- Eingabemaskenfunktion (usAVACDisp_InputMask)

  Stellt auf dem Bildschirm eine Eingabemaske inklusive der Titel- und Untertitelzeile dar. Die Funktionstasten (bzw. die korrespondierenden Felder in der 'bottom softlabel area') werden über Funktionsparameter gesteuert mit Namen und Call-Backs belegt. Die Eingabefelderdefinitionen werden in einem definierten Format als Liste erwartet. Dieses Format umfaßt den Feldnamen, die maximale Eingabelänge, Vorgabewerte und den Datentyp des Feldes und dient der folgenden Darstellung bzw. Benutzerinteraktion als Grundlage. Anhand der Längeninformationen werden die Eingabefelder in ihrer Form und Position berechnet und auf dem Bildschirm dargestellt. In der Benutzerinteraktion stellt sich die Maske auf intuitive Art und Weise dar. Auf einfache Art und Weise kann zwischen den Feldern navigiert werden, Zeichen können in Felder aufgenommen und wieder gelöscht werden.

4.3.2 Behandlung von Benutzereingaben

Funktionen bezüglich Benutzereingaben werden ebenso wie die anderen Funktionsgruppen in einem separaten Modul gehandhabt. Abhängigkeiten von dem konkret zu Grunde liegenden Frontend bleiben durch die Verwendung definierter Schnittstellen[74] außen vor und stehen einer Portierung auf andere Systeme, sofern sich deren Eingabemechanismus nicht zu sehr von dem hier verwendeten Paradigma unterscheidet, nicht im Wege. Mit dem minimalen angebotenen Befehlssatz ist es dem MobileNotes-Client möglich, alle notwendigen Eingaben abzuhandeln. Die Spezialisierung auf die konkrete Client-Implementation trat aber auch hier zu Gunsten einer offenen und für andere Anwendungen sinnvollen Implementation zurück.

Da die Benutzereingaben bei dem konkret vorliegenden Frontend, dem Philips Screen Phone P100, und bei einer Vielzahl alternativer Frontends durch die Verwendung einer Tastatur mit mehr oder weniger vielen Funktionstasten geschieht, wird die folgende Funktionalität benötigt:

- Einlesen eines Tastenschlages
- Einlesen einer vorgegebenen Anzahl von Tastenschlägen

Mit dieser Basisfunktionalität kann das aufsetzende Modul, im Falle des MobileNotes-Clients die graphische Benutzungsoberfläche, die Behandlung von Benutzereingaben realisieren. Im Falle des P100 wurden zwei zusätzliche Funktionen implementiert, eine Initialisierungs- und Terminierungsfunktion, die vor bzw. nach Nutzung dieses Moduls aufgerufen werden müssen. Momentan richten sie einen Hintergrundtask ein bzw. beenden ihn, welcher in Zeiten ohne Interaktion den Bildschirm ausschaltet (Bildschirmschonerfunktion), bzw. bei Wiederaufnahme der Interaktion den Bildschirm wieder einschaltet. Zukünftig können sie auch für andere Aufgaben verwendet werden.

Die beiden alternativen Eingabemöglichkeiten, die Tastatur und der erweiterte Zehnerblock (in der Philips-Notation Keyboard und Keypad), werden dabei identisch gehandhabt. Die angebotenen Funktionen beachten beide Eingabemöglichkeiten und geben alle Tastendrücke zurück.

[74] siehe Anhang F

4.3.3 Speicherverwaltung

Die Speicherverwaltung abstrahiert von dem konkret zu Grunde liegenden Speichermodell, kapselt systembedingte Besonderheiten und bietet allen Client-Modulen eine definierte Speicherschnittstelle[75] an. Grundsätzlich werden Mechanismen angeboten, welche die temporäre und persistente Bereitstellung von Speicherbereichen unterstützen, den Speicher auf Konsistenz überprüfen können und einfache Speicheroperationen durchführen können. Im Falle des P100 wurde die systeminterne Speicherverwaltung sehr stiefmütterlich behandelt und mußte zu großen Teilen neu implementiert werden. Bis auf einen Speichermanager, der große PCMCIA-Speicherkarten nicht voll ausnutzen und Daten nicht dauerhaft speichern kann, bietet das POS keinerlei Unterstützung.

Die Anforderungen an eine eigene Speicherverwaltung wurden im MobileNotes-Projekt wie folgt festgelegt:

- Verwaltung großer Speicherbereiche (ggf. auf Speicherkarten) ermöglichen
- Unterstützung dauerhafter (persistenter) und temporärer Speicherbereiche
- Portierbarkeit auf andere Clients

Anforderungen die Speicherverwaltung mehrprozeßfähig zu machen existierten nicht, da das MobileNotes-Client-Programm von dem Multitasking des POS nur sehr spärlichen Gebrauch macht.

In der Standardkonfiguration ist das P100 mit vier 128KB Speicherchips bestückt. Zwei dieser Speicherchips sind als Flash-ROM ausgeführt und beinhalten die Systemprogramme. Ein Systemsoftwareupdate wird somit ohne Änderung der Hardwarekonfiguration, möglich. (Philips unterstützt die Möglichkeit über Modemverbindung neue Systemprogramme einzuholen und direkt in dem P100 zu installieren.) Die zwei verbleibenden Chips bilden das teilweise reservierte, für Applikationen nutzbare flüchtige RAM. Eine ggf. vorhandene PCMCIA-Speicherkarte wird in Einheiten von 256KB in den Adreßraum des Telephons eingeblendet. Maximal lassen sich zu einem Zeitpunkt zwei Speicherbänke in den Adreßraum einblenden, so daß maximal 512 KB Kartenspeicher adressierbar ist. Sollen weitere Speicherbänke auf der Karte angesprochen werden, so muß das Anwendungsprogramm einen expliziten Bankwechsel, d.h. Ausblenden einer z.Zt. gültigen Speicherbank und Einblenden einer weiteren Speicherbank, durchführen. Durch die Ausnutzung dieses Mechanismusses ist theoretisch die Ausnutzung beliebig großer Speicherkarten vorstellbar, praktisch existiert eine obere Schranke bei 4 MB.[76]

[75] siehe Anhang G

[76] siehe [Philips 1995d]

Aus diesen Einzelteilen ergibt sich folgendes Gesamtbild:

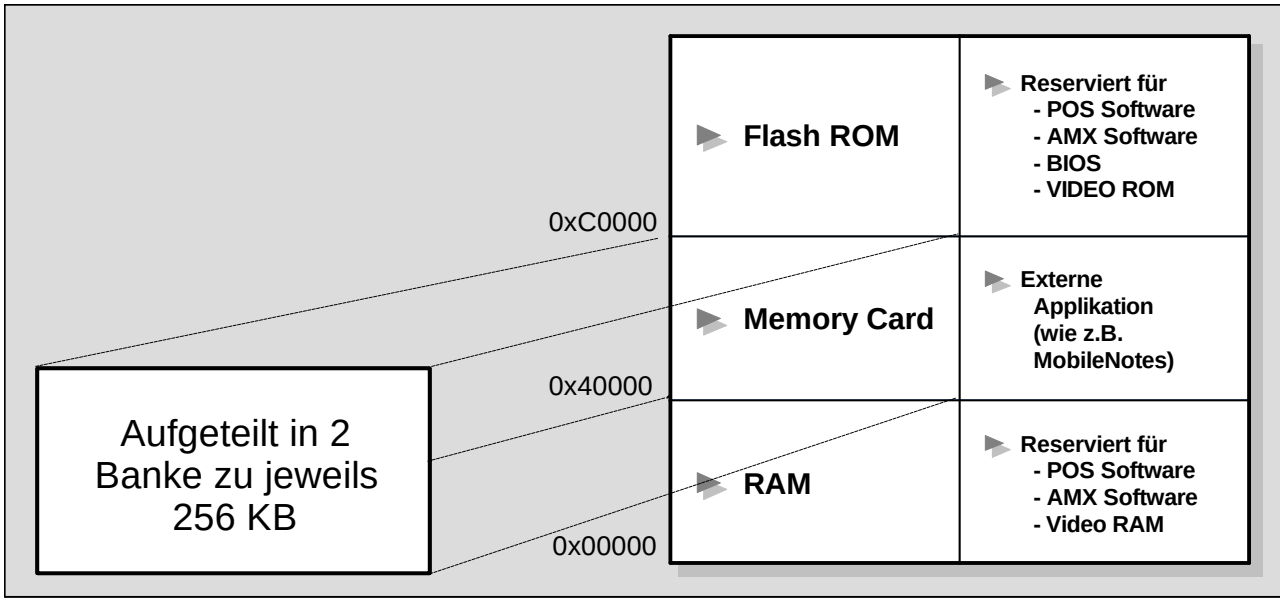

Abbildung 13: Speichermodell des P100

Verfahren

Die Speicherverwaltung ist als eigenständiges Modul aufgebaut, das nur über vorher definierte Schnittstellen angesprochen werden kann. Diese definierten Schnittstellen erlauben eine Abstraktion von dem konkreten unterliegenden Speicher. Zukünftige Portierungen werden durch diese Abstraktion und Kapselung vereinfacht, sofern die Zielplattform nicht ein völlig anderes Speicherparadigma verwendet (z.B. Objektorientierte Speicherungstechniken). Durch die Einführung eines virtuellen Speicherhandles und die konsequente Verwendung desselben, kann z.B. die Organisation des Speichers in Bänke verborgen werden. Die Auslegung aller Client-Schnittstellen auf diesen neuen Speicherreferenztyp führt z.B. bei der vorliegenden Implementation zur prinzipiellen Unterstützung beliebig großer bankweise organisierter PCMCIA-Speicherkarten.

Grundsätzlich basiert die neu implementierte Speicherverwaltung auf der Verkettung von freien und vergebenen Speicherbereichen zu einer linearen Liste. Wird ein Speicherblock angefordert (Allokation), so wird die lineare Liste nach einem freien Speicherbereich passender Größe durchsucht. Wird einer gefunden, kann die Speicheranforderung befriedigt und ein Handle für diesen Speicherbereich zurückgegeben werden. Wird kein freier Speicherbereich passender Größe gefunden, kann die Speicheranforderung nicht befriedigt werden. Die Unterscheidung, ob ein Element der linearen Liste einen belegten oder freien Speicherbereich repräsentiert, geschieht anhand eines Statusflags, welches die Zustände 'Belegt' und 'Frei' annehmen kann (Die Implementation differenziert hier noch etwas weiter, doch dazu später mehr).

Das Auffinden passender freier Speicherbereiche innerhalb der Liste ist durch mehrere Verfahren möglich, in der Literatur[77] werden z.B. folgende mehr oder minder gebräuchliche Verfahren behandelt:

- First Fit

  Der erste freie Speicherbereich in der linearen Liste, der die Anforderung erfüllt oder übererfüllt wird ausgewählt.

- Next Fit

  Die Suche setzt beim letzten identifizierten Speicherblock wieder auf und liefert den nächsten freien Speicherbereich in der linearen Liste, der die Anforderung erfüllt oder übererfüllt.

- Best Fit

  Die gesamte lineare Liste wird nach freien Speicherbereichen durchforstet und der kleinste noch ausreichende Bereich zurückgeliefert.

- Worst Fit

  Die gesamte lineare Liste wird nach freien Speicherbereichen durchforstet und der größte ausreichende Bereich zurückgeliefert.

Die Übererfüllung einer Speicheranforderung resultiert in der Teilung des Speicherbereiches. Der erste Teil in Größe der Speicheranforderung, wird als Anforderungsergebnis zurückgeliefert und fortan als belegter Speicherbereich in der linearen Liste geführt. Der verbleibende Rest wird als freier Speicherbereich weiterhin in der linearen Liste geführt. Vor diesem Hintergrund bekommt auch das scheinbar abstruse Suchverfahren 'Worst Fit' einen Sinn, da es nicht kleine, unbrauchbare freie Speicherbereiche hinterläßt, sondern ganz bewußt große freie Speicherbereiche produziert, die gut zukünftige Anforderungen befriedigen können.

Die Freigabe (Deallokation) eines als 'Belegt' gekennzeichneten Speicherblocks erfordert im einfachsten Fall nur das Setzen des Statusflag auf 'Frei'. Doch würde diese Vorgehensweise allein im Laufe der Zeit zu einer Vielzahl von kleinen freien Speicherbereichen führen. Daher werden bei der Deallokation eines Speicherbereiches die benachbarten Speicherbereiche (benachbarte Listenelemente) untersucht, ob eine Verschmelzung zu einem größeren freien Speicherblock möglich ist. Im schlechtesten Fall läuft dies dennoch auf das einfache Umsetzen des Statusflag hinaus. In den anderen Fällen werden zwei oder drei Speicherblöcke in ein Listenelement und somit zu einem großen Block verschmolzen. Aus Gründen der Sicherheit sollten freigegebene Speicherblöcke, wie es auch bei der vorliegenden Implementation geschieht, von der Freigaberoutine komplett überschrieben werden, um unbefugte Einsichtnahme zu verhindern.

[77] siehe [Tanenbaum 1994, S. 103ff]

Implementation

Die konkrete Implementation bildet das oben geschilderte Verfahren unter Berücksichtigung der P100 spezifischen Besonderheiten ab und nutzt zur Suche eines passenden Speicherbereiches das 'First Fit'-Verfahren. Ein Speicherhandle umfaßt in dieser Version die Nummer der Bank, in welcher der Speicherbereich liegt, und die Adresse innerhalb dieser Bank. Soll ein Speicherbereich angesprochen werden, so kann durch das Handle die zugehörige Bank ermittelt und mit einer speziellen Funktion, die mit dem Handle parametrisiert wird, in den aktuellen Adreßraum eingegliedert werden. Ist diese Fokussierung durchgeführt worden, kann mit normalen Zeigeradressierungstechniken auf den Speicherbereich zugegriffen werden. Eine erneute Fokussierung ist erst dann wieder notwendig, wenn zwischenzeitlich andere Handles verwendet wurden, und somit die Gefahr besteht, daß die Bankkonstellation verändert wurde. Ein Handle besitzt in der vorliegenden Implementation folgende Struktur und wird als Ergebnis einer Speicherallokation zurückgeliefert:

1 Byte	5 Bytes	6 Bytes
Banknummer	**Speicheradresse**	**Füllbyte**
Numer der zu adressierenden 256KB-Einheit.	Adresse in der Speicherbank	Byte beliebigen Inhalts, um die Struktur auf eine bestimmte Länge zu bringen. Kann im weiteren ignoriert werden. Es vereinfacht allerdings die Implementation, da diese Struktur somit 6 Bytes, eine gerade Anzahl von Bytes, groß ist.

Abbildung 14: Speicherhandle

Die Batteriepufferung der PCMCIA-Speicherkarte erlaubt die persistente Speicherung von Daten und bietet dem MobileNotes-Client eine Alternative zu dem fehlenden persistenten Sekundärspeicher. Durch die Fixierung einer bestimmten Struktur (im folgenden Anker, s.u. Static Memory Block) auf der Speicherkarte, ist es dem MobileNotes-Client bei jedem Programmstart möglich, die auf der Karte vorhandene lineare Liste zu identifizieren und weiterhin zu nutzen. Die Nutzung einer vorhandenen linearen Liste wird durch die bei jedem Programmstart notwendige Speicherverwaltungsinitialisierungsroutine ermöglicht. Durch geeignete Parametrisierung kann diese Routine angewiesen werden, eine ggf. vorhandene Liste aufzuspüren und weiterhin zu nutzen, oder den Kartenspeicher kategorisch in einer neuen linearen Liste zu organisieren. Kann die erste Möglichkeit keine Liste identifizieren oder ist die identifizierte Liste defekt, so wird, mit Einverständnis des Benutzers, der Kartenspeicher ebenfalls in einer neuen linearen Liste organisiert. Im Falle einer Neuorganisation gehen alle ehemals gespeicherten Informationen verloren.

Die Elemente der notwendigen linearen Liste werden den einzelnen Speicherbereichen direkt voran gestellt. D.h. statt einer separaten linearen Liste, deren Elemente auf die einzelnen Speicherbereiche verweisen, werden die Speicherbereiche, angereichert um einige Zusatzinformationen, selbst als Listenelemente genutzt. Als Anker für diese Speicherelementliste dient eine statische Struktur, die mit SMB (Static Memory Block) betitelt ist und eine fixierte,

durch das Client-Programm definierte Position im Speicher hat. Neben den Informationen zum Auffinden der Liste enthält der SMB noch weitere Verweise, die dem Client-Programm die Möglichkeit bieten den Wählstring, den Modeminitialisierungsstring und weitere Daten dauerhaft, über das Programmende hinaus, zu speichern. Eine aus dieser Implementation resultierende Speicherorganisation sieht wie folgt aus:

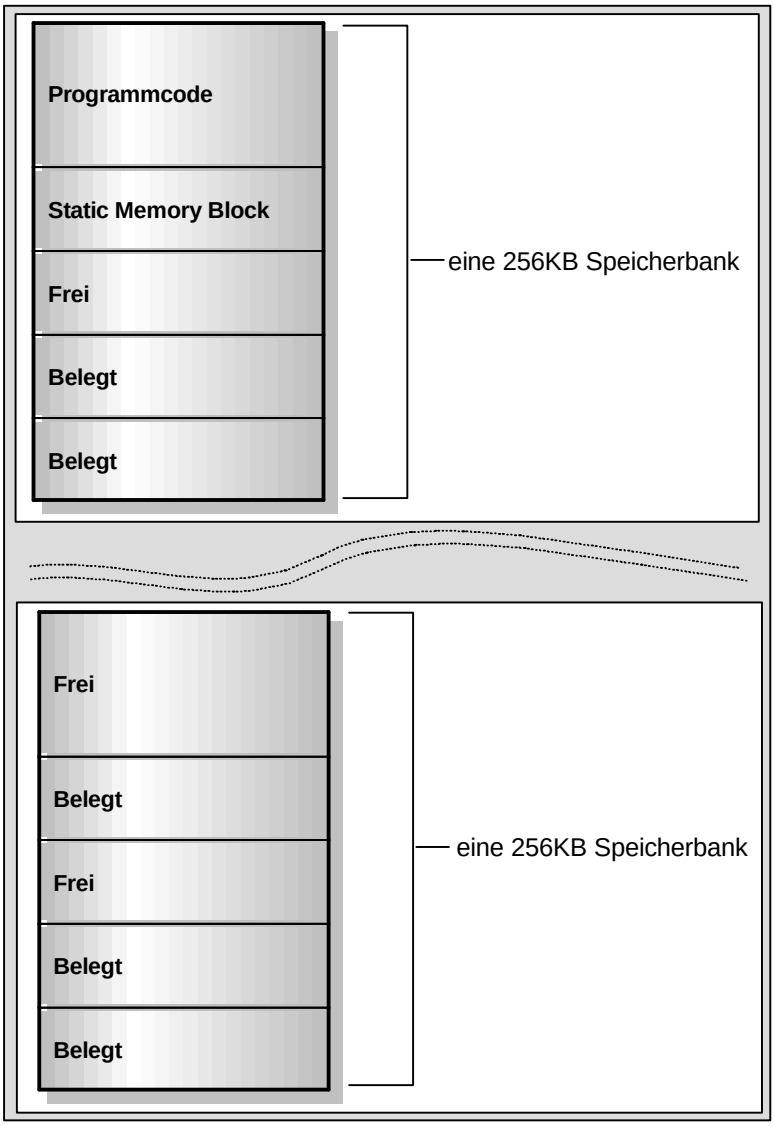

Abbildung 15: Typische Speicherorganisation eines MobileNotes-Clients

Die den Speicherbereichen angegliederten Zusatzinformationen werden als Präfix vor dem zur Verfügung stehenden Speicherbereich gehandhabt. Dieser Präfix, zusammengefaßt in einer Struktur, die den Titel 'Memory Control Block'[78] trägt, schließt Informationen zum Auffinden des vorigen und des nächsten Speicherbereichs, Informationen über die Größe und den Belegungsstatus des Speicherblocks ein.

[78] MCB, angelehnt an Microsofts MS-DOS-Notation

6 Byte	12 Bytes	14 Bytes	15 Bytes	16 Bytes
Speicherhandle	**Speicherhandle**	**Speichergröße**	**Speicherinhalt**	**Füllbyte**
Verweisauf den vorhergehenden MCB.	Verweis auf den vorhergehenden MCB.	Größe des folgenden Speicherbereiches wird hier in Vielfache von 16 Bytes (16Bytes entsprechen einem Paragraphen) angegeben.	Indikator für den Inhalt des folgenden Speicherbereiches. (Kann die Werte 'Frei','Temporär belegt' und 'Dauer- haft belegt' annehmen)	Ein Byte mit beliebigen Inhalt um diese Struktur auf eine bestimmte Länge zu bringen.

Abbildung 16: Memory Controll Block (MCB)

Mit dem Belegungsstatus ist es möglich, zwischen dauerhaft gespeicherten Speicherblöcken, die nach Programmende bestehen bleiben, temporären Speicherblöcken, die nach dem Programmende ihre Gültigkeit verlieren, und freien Speicherblöcken zu differenzieren. Über einen Zusatzparameter gesteuert kann die Allokationsroutine angewiesen werden, einen dauerhaften oder einen temporären Speicherblock zu allozieren.

Der Belegungsstatus kann die folgenden Werte annehmen:

- AVA_MEM_FREE (bzw. 0x00)
  Freier Speicherblock, der bei Allokationswünschen zurückgeliefert werden kann.
- AVA_MEM_ALLOC_NOT_PERSISTENT (bzw. 0x01)
  Allozierter Speicherblock, dessen Lebenszeit mit dem Programmende automatisch endet.
- AVA_MEM_ALLOC_PERSISTENT (bzw. 0x02)
  Allozierter Speicherblock, dessen Lebenszeit nicht mit dem Programm endet, es ist eine explizite Deallokation notwendig.

Die z.B. beim Programmstart aufgerufene Speicherverwaltungsinitialisierungsroutine, sofern sie zur Identifikation einer vorhandenen linearen Liste verwendet wird, durchsucht die Liste nach Speicherblöcken mit dem Status 'Nicht Persistent' und gibt diese gezielt frei. Dies wird notwendig, da durch die Natur der Speicherkarte alle Daten batteriegepuffert werden und auch als 'Nicht Persistent' gekennzeichnete Speicherblöcke bei einem erneuten Programmstart noch existent sind. Dieser Mechanismus schafft eine gewisse Fehlertoleranz gegenüber vergessenen Deallokationen und durch Programmunterbrechungen verhinderte Deallokationen, spätestens der nächste Programmstart bereinigt den Speicher wieder von 'Nicht Persistenten'-Blöcken.

4.3.4 Anwendungssteuerung

Die Anwendungssteuerung fügt sich wie alle anderen Komponenten des MobileNotes-Projektes nahtlos in das Konzept der Modularität ein. Prinzipiell unabhängig von dem konkret zu Grunde liegenden Frontend nutzt die Anwendungssteuerung (im folgenden auch Hauptprogramm genannt) nur Funktionen der anderen Module. Intern fixiert das Hauptprogramm den Programmfluß, wie sich das Programm dem Benutzer präsentiert. Andere auf dem Client vorhandenen Module bilden die für den Programmablauf notwendige Funktionsbibliotheken, nehmen aber keinen Einfluß auf den konkreten Programmablauf. Von der Abfrage des Benutzernamens beim Login bis hin zur Programmbeendigung geht jegliche Steuerung von diesem Modul aus. Das dadurch implementierte Verhalten, bzw. die in diesem Modul implementierten Möglichkeiten eines MobileNotes Client können somit sehr schnell auf andere Frontends übernommen werden, sofern die anderen Module an die konkreten Gegebenheiten angepaßt wurden.

Diese Eigenschaften lassen eine Beschreibung der Anwendungssteuerung in eine Beschreibung des MobileNotes-Programmflußes auf dem Client resultieren.

Durch die Intention dieses Projektes getrieben, den 'AvALoN'-Prototypen um zusätzliche Funktionalitäten zu bereichern, wurde das Hauptprogramm komplett neu entwickelt und spiegelt nun die neuen und erweiterten Möglichkeiten der unterliegenden Komponenten wider.

Ein typischer Programmablauf beginnt immer mit der Darstellung eines Startbildschirms, der dem Benutzer die Eingabe seines persönlichen Logins erlaubt. Zukünftige Versionen werden die Smart-Card eines P100 nutzen können und so die Identität eines Benutzers anhand eines eingegebenen und des auf der Smart-Card gespeicherten Paßwortes sicher bestimmen können. Derzeit stehen Smart-Cards für dieses Projekt nicht zur Verfügung und erlauben so die Anmeldung ohne Paßwort. Ist der Benutzername neu eingegeben oder der vom vorherigen Benutzer eingegebene und erneut dargestellte Benutzername bestätigt worden, so wird er mit dem auf der Speicherkarte abgelegten Namen des Karten-Eigentümers verglichen. Sind sie deckungsgleich, so unterliegt der komplette Speicherkarteninhalt der Kontrolle des Benutzers, andernfalls, Benutzer und Eigentümer der Speicherkarte sind nicht identisch, wird ein eingeschränkter Benutzungsmodus aktiviert.

Die beiden möglichen Nutzungsmodi, Benutzer mit eigener Karte und Benutzer mit fremder Karte, unterscheiden sich in den zur Verfügung stehenden Funktionen. Während der Kartenbesitzer die vollen Möglichkeiten des MobileNotes-Client ausschöpfen kann, unterliegt der Nutzer einer fremden Speicherkarte bestimmten Einschränkungen:

- Speicherkarte kann nicht neu initialisiert werden.
- Auf der Speicherkarte vorhandene Daten können nicht eingesehen werden.
- Persistente Datenspeicherung auf der Speicherkarte ist nicht möglich, jegliche lok erzeugte Information geht bei Programmabschluß verloren.

Diese Differenzierung erlaubt die gemeinsame Nutzung von Speicherkarten, ohne deren Inhalte unbefugt zu verändern oder einzusehen. Viele Situationen, in denen die eigene Speicherkarte nicht verfügbar ist, erlauben den sinnvollen Einsatz dieses eingeschränkten Benutzungsmodi, z.B. Mail-Abfrage, Adreßbuchabfrage, etc.

Ist die Benutzeridentifikation durchgeführt worden und der Benutzungsmodus festgelegt, so werden dem Benutzer die aktuell verfügbaren Datenbanken zur Auswahl angeboten. Durch die Auswahl einer dieser Datenbanken verzweigt der Benutzer in den Kontext derselben und bekommt die in der Datenbank enthaltenen Dokumente und die Möglichkeit zur Dokumentenerstellung zur Auswahl angeboten. Besteht eine Verbindung zum Server kann explizit zwischen der Anzeige lokal gespeicherter Dokumente (im Falle des P100 auf der Memory-Card) und der Anzeige auf dem Server liegender Dokumente umgeschaltet werden. Durch die Auswahl eines dargestellten Dokumentes verzweigt der Benutzer in den Kontext desselben und bekommt Möglichkeiten zur Modifikation, Beantwortung oder Löschung dieses Dokumentes angeboten. Wurde ein auf dem Server lokalisiertes Dokument ausgewählt, so wird ihm ermöglicht, das Dokument lokal auf dem Frontend, zwecks späterer Offline-Recherche zu speichern.

Bei all diesen Operationen beachtet das Frontend die dem Benutzer zugestandenen Rechte auf eine Datenbank, damit ungültige Operationen gar nicht erst angeboten werden und der Benutzer aus verwirrenden Situationen herausgehalten werden kann. Hinzu kommen die verschiedenen Nutzungsmodi für das Programm selbst, ob eine eigene Karte verwendet wird oder eine fremde, ob eine Serververbindung besteht oder nicht.

Schematisch stellt sich der Programmfluß wie folgt dar:

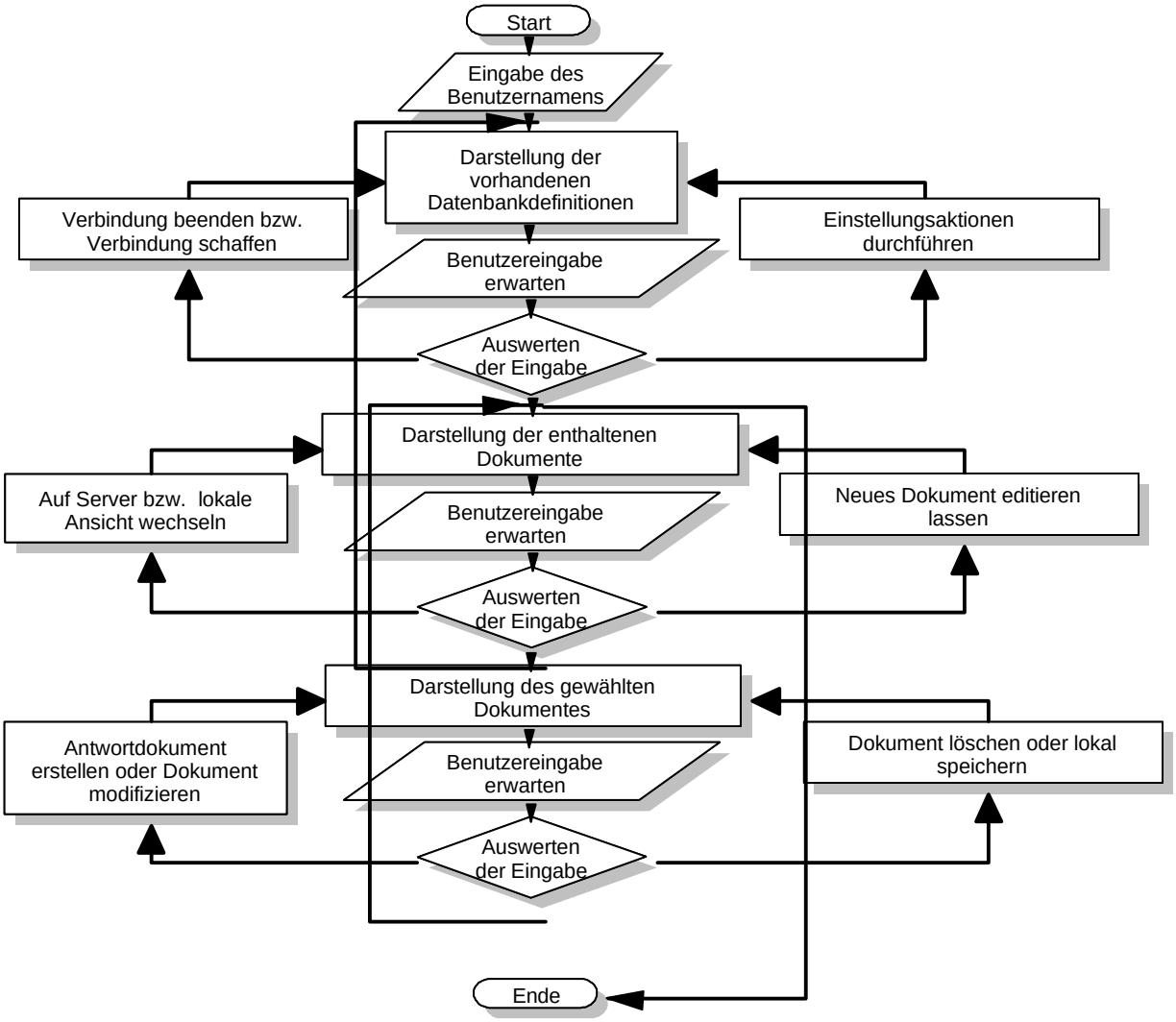

Abbildung 17: Schematischer Programmablauf eines MobileNotes-Clients

4.4 Problempunkte und zukünftige Erweiterungsmöglichkeiten

Obwohl die vorliegende praktische Arbeit 'MobileNotes' auf Vorarbeiten und dem Vorwissen der drei beteiligten Studenten aufsetzt, konnte auch diesmal kein ganz problemfreies Produkt erstellt werden. Es offenbarten sich, teilweise erst während der Programmierung, Grenzen, die von den Beteiligten mehr oder minder elegant behandelt wurden. Probleme mit den Entwicklungsumgebungen und ungenügende Dokumentationen kristallisierten sich dabei als Haupthindernisse heraus. Daher im folgenden ein Überblick über die Hauptproblempunkte:

Allgemein:

- Anmeldevorgang nur ungenügend abgesichert

  Vorgesehen war für den Anmeldevorgang die Nutzung des RSA-Verschlüsselungsverfahrens. Erschwerte Literatursuche, ungewisse Lizenzbestimmungen, hohe benötigte Rechenleistung und die komplexen Zusammenhänge führten letztendlich zur Implementation eines Anmeldeverfahrens auf Basis des DES-Verfahrens. Die offenen Struktur und definierte Schnittstellen ermöglichen allerdings die zukünftige Umstellung auf andere Verfahren.

Server:

- Die Serverkomponente von MobileNotes kann derzeit nur eine Verbindung bearbeiten

  Dieses Problem liegt darin begründet, daß Notes in der Version 3.x, bzw. die zugehörigen API's nicht multithreading-fähig sind. Unter Notes 3.x kann nur ein Thread eines Prozesses API-Funktionen nutzen, jede Nutzung von API-Funktionen in einem weiteren Thread resultiert in unvorhersehbaren Fehlersituationen. Vor dem Hintergrund, daß mit der Version 4.0 von Lotus Notes dieses Problem der Vergangenheit angehört, wurde dieses Problem nicht gesondert behandelt. Grundsätzlich wurde die Serverkomponente auf die Behandlung mehrerer gleichzeitiger Verbindungen ausgelegt.

Client:

- Relokation der Clientkomponente ist nicht implementiert

  Die Relokation wird mit der Einführung von Dualmode PCMCIA-Speicherkarten (Flash-ROM und RAM) notwendig, da die ausführbaren Komponenten des Programms nicht im Flash-ROM ausführbar sind. Derzeit wird dieses Problem nicht weiter betrachtet, da die Dokumentation zum P100 keine Vorgehensweise zur Lösung dieser Problematik offenbart.

- Wechsel zwischen P100-Hauptanwendung und MobileNotes ist nicht beliebig möglich

  Derzeit kann zwischen den beiden Programmen nicht beliebig umgeschaltet werden, bei jedem Wechsel wird die Beendigung der jeweils anderen Anwendung vorausgesetzt. Dieses Manko ist erst mit erweiterten Schnittstellen des POS zu lösen, derzeit ist keine Abhilfe möglich.

Aus diesen Problempunkten ergeben sich schon eine ganze Reihe an Erweiterungsmöglichkeiten, der Phantasie sind kaum Grenzen gesetzt. So können z.B. andere Frontends mit anderen Möglichkeiten der Informationswiedergabe durchaus für die Übertragung von Binärdaten geeignet sein. Bei Geräten mit Ton-Wiedergabe kann es sinnvoll sein Klangobjekte an den Client zu übergeben. Geräte mit Ton-Aufnahmemöglichkeit könnten sogar den umgekehrten Weg beschreiten und Klangobjekte an den MobileNotes-Server hochschicken. Auch die Übertragung von angehängten Dateien kann sinnvoll sein, Clients mit großem Sekundärspeicher können sie aufnehmen und an andere Verbraucher weiterreichen. Das zur Datenübertragung zwischen Client und Server genutzte Protokoll stellt sich dabei als große Hilfe heraus. Clienttypspezifisch können Erweiterungen vorgenommen und so die speziellen Möglichkeiten ausgenutzt werden.[79]

[79] siehe [Altmiks 1996] u. [Sievers 1996]

5. Zusammenfassung und Ausblick

In den ersten beiden Kapiteln wurden die derzeitigen Möglichkeiten der mobilen Kommunikation ausgeleuchtet. Schwerpunkt wurde dabei auf die Datenkommunikation vor dem Hintergrund der mobilen Nutzung von Groupware gelegt. Dazu wurde ein Überblick über die Entwicklungen, die derzeit erhältlichen Produkte und die Zukunftsvisionen im Bereich der mobilen Kommunikation gegeben und das weite Feld der Groupware ausgeleuchtet. Allerdings stellt sich bei dieser Betrachtung heraus, daß die Kombinationsmöglichkeiten dieser beiden Techniken oftmals angesichts der Hindernisse, die in vielen Fällen einer ernsthaften Nutzung entgegenstehen, verblassen. Unzureichende Übertragungsraten, hohe Übertragungskosten und mannigfaltige technische Einschränkungen führen nicht zum euphorischen Siegeszug der mobilen Kommunikation. Die Spezialisierung der Mobilfunksysteme auf die Sprachkommunikation und die beschränkte Anzahl der zur Verfügung stehenden Frequenzen wirken sich negativ auf die technisch möglichen Übertragungsraten aus und beschränken so die Nutzungsmöglichkeiten in Zeiten wachsender (multimedialer) Datenmengen. Während bei den leitungsgebundenen Übertragungsnetzen stetig steigende Übertragungsraten realisiert werden, beginnen sich die Grenzen des Mobilfunk schon heute abzuzeichnen. Mit vertretbarem Aufwand lassen sich nur noch geringe Leistungszuwächse erzielen, jeder Versuch darüber hinaus muß mit unverhältnismäßig hohen Kosten und enormen technischen Aufwand bezahlt werden. Trotz Standardisierungsbemühungen, technischen Innovationen und wachsender Konkurrenz auf den Telekommunikationsmärkten läßt sich daran nur wenig ändern. Allein die Rahmenbedingungen der mobilen Kommunikation werden kontinuierlich verbessert. Jedoch stellt sich die Frage, inwiefern sinkende Gerätepreise, grenzüberschreitende Gerätenutzung, internationales Roaming und weitere Errungenschaften über das Hauptmanko, die geringe Übertragungsleistung, hinwegspielen können.

Dies soll jedoch nicht in der kategorische Ablehnung der mobilen Kommunikation enden, eher im Aufruf, das Anwendungsgebiet, die Anforderungen und die Alternativen vor der Einführung konkreter Lösungen zu untersuchen und auch evtl. unbequemere, aber oftmals geeignetere Festnetz-Kommunikationsmöglichkeiten mit in den Prozeß der Lösungsfindung einzubeziehen. Die Problematik der geringen Übertragungsleistung darf zudem nicht überbewertet werden. Primär definiert sich der Nutzen der mobilen Kommunikation durch betriebswirtschaftliche und nicht durch technische Aspekte (oftmals wirken sogar nicht monetäre Aspekte, wie z.B. Imagegewinn positiv auf den Nutzen ein). Viele Anwendungsgebiete sind durchaus für den Betrieb über Mobilfunk geeignet, andere gar ohne Mobilfunk undenkbar, wiederum andere können durchaus ohne Mobilfunk effektiver sein.

Wie ebenfalls festgestellt bietet der Markt mittlerweile eine Vielzahl von Lösungen, die bei der Entscheidungsfindung Beachtung finden sollten. Lösungen zur digitalen Datenübertragung (z.B. M1 von AVM) ohne den Umweg über die Emulation eines Analogmodems gehören ebenso dazu,

wie Kombinationslösungen[80], die sowohl Verbindungen zum Festnetz, als auch Funkübertragung unterstützen. Ebenfalls ist zu beachten, daß mittlerweile mehrere Mobilfunknetze existieren, die mit besonderen Fähigkeiten, Abrechnungsmodi und Übertragungsleistungen auch spezielle Bedürfnisse befriedigen können.

Für die Zukunft können keine gravierenden Änderungen an dieser Situation prophezeit werden. Die in großen Teilen stiefmütterliche Behandlung der Datenkommunikation wird weiter anhalten. Ein Aufschluß an die Möglichkeiten der netzgebundenen Kommunikation wird wohl ein dauerhafter Wunschtraum bleiben. Die Rahmenbedingungen allerdings werden sich kontinuierlich verbessern. Kostengünstigere Endgeräte, bessere Versorgung mit Sende-/Empfangsanlagen, spezialisiertere Mobilfunknetze, Standardisierungsbemühungen und viele weitere Entwicklungen vermögen aber dennoch nicht die Begrenzung der Bandbreiten im Äther aufzuheben.

Die beiden folgenden Kapitel gehen auf das Projekt MobileNotes der Studenten Bernd Altmiks, Nico Dirks und Dirk Sievers ein. Einleitend mit einigen Projektgrundlagen im Kapitel 3 geht es im Kapitel 4 in die Beschreibung der von Nico Dirks realisierten Projektkomponenten über. Dabei kristallisiert sich die hinter MobileNotes stehende Idee, die Anbindung relativ leistungsschwacher mobiler Endgeräte an Groupware-Backend-Server, ggf. mit eingeschränkter Leistungsfähigkeit als durchaus attraktiv heraus. Sie wird am Beispiel der Anbindung des Philips Screen Phone P100 an das Groupware Produkt Lotus Notes aufgezeigt. Stellt auch die gewählte Clientplattform nicht das ideale mobile Kommunikationsendgerät dar, so erlaubt die Client-Implementation zukünftige Portierungen auf weitere Endgeräte und demonstriert die im Vergleich zu anderen Produkten (z.B. SkyNotes u. TeamAgent) größere Systemoffenheit. Dabei wurde sowohl von den Erfahrungen aller Beteiligten durch die Erstellung des Prototyps AvALoN im Rahmen einer Seminararbeit als auch von der Kooperation mit den beteiligten Unternehmen und der Lehr- und Forschungseinheit Wirtschaftsinformatik 2 der Universität Gesamthochschule Paderborn profitiert.

In einer Zeit, in der Geschwindigkeit, Informationen und fortwährende Erreichbarkeit den wirtschaftlichen Erfolg bestimmen und die Differenzierung von der Konkurrenz erlauben, wird die mobile Kommunikation, allen Widerständen zum trotz, weiter an Bedeutung zunehmen. Die Anbindung der im mobilen Betrieb verwendeten Computertechnik an die gewohnten Ressourcen des stationären Betriebs, seien es Datenbanken, Dateien oder Betriebsmittel erlaubt einen großen Schritt in Richtung Mobile Business.

Der Integrationsprozeß ortsgebundener Informationsnutzer in Unternehmen durch unternehmensübergreifende Datenmodelle, Computergestützte Gruppenarbeit und gezielten Geschäfts-

[80] z.B. FuryCard 19.2 Duo von Dr. Neuhaus u. GSM-Ready PCMCIA 50 von IME Marketing & Engineering

prozeßumgestaltungen wird dabei auf mobilen Anwender ausgedehnt. Die Anbindung der mobilen Nutzer an Groupware-Backend-Server ist somit konsequent.

Anhang A: Literaturverzeichnis und Informationsquellen

[Altmiks 1996]
Altmiks, B.: Mobile Computing und Serveranbindung;
Entwicklung einer generischen Serverkomponente für die Anbindung mobiler Kommunikationsfrontends an Groupware Backend Server;
Diplomarbeit an der Universität-Gesamthochschule Paderborn, Lehr- und Forschungseinheit Wirtschaftsinformatik 2, 1996

[Apple 1993]
o.V.: The Apple Open Collaborative Environment Technology;
Working With Others Has Never Been Easier;
Cupertino, CA, 1993

[Becker, Heim 1991a]
Becker, K.-F., Heim, G.: Unterwegs in der Welt mobiler Kommunikation;
Status quo und Visionen zum Funk;
In: telecom report 5/1991, S. 235-237

[Becker 1991b]
Becker, K.-F.: Vom "Fleckerlteppich" zum Einheitsnetz GSM900;
In: telecom-report 1/1991, S. 8-11

[Berlage, Schnöring 1995]
Berlage, M., Schnöring, T.: The Introduction of Competition in German Mobile Communications Markets;
In: Mobile Telecommunications: Emerging European Markets;
Hrsg. Schenk, K.-E., Müller, J., Schnöring, T., Artech House Publishers London, 1995

[Bucheit 1992]
Bucheit, M.: Windows Programmierbuch;
Verlag: Sybex-Verlag Düsseldorf, 1992

[Decker, Walke 1993]
Decker, P., Walke, B.: Mobile Datenkommunikation;
In: it+ti Mai 1993, S. 12 ff

[Encarnacao, Hornung, Noll 1994]
Encarnacao, J. L., Hornung, C., Noll, S.: Computer Supported Cooperative Work (CSCW): Stand und Perspektiven;
In: it+ti (4/5)/1994, S. 96-104

[Eberlen 1996]
Eberlen, L.: Mobile ISDN;
Remote Access für mobile Mitarbeiter;
In: Gateway 4/1996, Heinz Heise Verlag Hannover, S. 50-52

[Edge 1995]
o.V.: Lotus Notes HiTest C API;
Release 2.2;
In: Notesdatenbank, 1995

[Esser-Wellie 1996]
Esser-Wellie, M.: Regulierung (Teil 1 u. 2);
Was das neue Telekommunikationsgesetz bringt;
In: Gateway 3/1996, S. 66-71 und 4/1996, Heinz Heise Verlag Hannover, S. 86-90

[Feichtinger, Hofreiter 1995]
Feichtinger, H., Hofreiter, H.-P.: Modacom;
Daten funken;
In: c`t 6/1995, Heinz Heise Verlag Hannover, S. 252-256

[Finke 1992]
Finke, F.: Groupwaresysteme;
Basiskonzepte und Beispiele für den Einsatz im Unternehmen;
In: Information Management 1/1992, S. 24-30

[Flohr 1994]
Flohr, U.: Newton;
Aufbau, Funktion, Programmierung;
Verlag: Heinz Heise Verlag Hannover, 1994

[Forman, Zahorjan1994]
Forman, G. H., Zahorjan, J.: The Challenges of Mobile Computing;
In: Computer, April 1994, S.38ff.

[Geihs 1993]
Geihs, K.: Infrastrukturen für heterogene verteilte Systeme;
In: Informatik-Spektrum 1993, Springer Verlag Hamburg, S. 11-23

[Huber 1995]
Huber, J.-F.: Mobilkommunikation auf Expansionskurs;
Von lokaler zu weltweiter mobiler Erreichbarkeit - von Mobiltelefonie zu mobiler Daten- und Bildkommunikation;
In: telecom-report 2/1995, S. 52-55

[Hulten, Mölleryd 1995]
Hulten, S., Mölleryd, B. G.: Mobile Telecommunications in Sweden;
In: Mobile Telecommunications: Emerging European Markets;
Hrsg. Schenk, K.-E., Müller, J., Schnöring, T., Artech House Publishers London, 1995

[Jowett 1992]
Jowett, A.: Mobile Data Communications - Choices and Opportunities;
Verlag: NCC Blackwell Oxford, 1992

[Kadak]
o.V.: KADAX AMX 86 Multitasking Reference Manual für the 80x86 Family;
unbekanntes Erscheinungsdatum

[Kaiser 1994]
Kaiser, F.: Jetzt funkt's;
Konzepte und Angebote im D1-, D2- und E-Plus-Netz;
In: c`t 7/1994, Heinz Heise Verlag Hannover, S. 128-136

[Keller 1986]
Keller, R.: Frust für Fehlerteufel;
Von fehlererkennenden und fehlerkorrigierenden Codes;
In: c`t 2/1986, Heinz Heise Verlag Hannover, S. 54-56

[Krcmar 1992]
Krcmar, H.: Computerunterstützung für die Gruppenarbeit;
Zum Stand der Computer Supported Cooperative Work Forschung;
In: Wirtschaftsinformatik, 4/1992, Vieweg Verlag Wiesbaden, S. 425-437

[Lewe, Krcmar 1991]
Lewe, H., Krcmar, H.: Groupware;
In: Informatik-Spektrum 1991, S. 345-348

[Lobensommer 1994]
Lobensommer, H.: Die Technik der modernen Mobilkommunikation;
Grundlagen, Standards, Systeme und Anwendungen;
Verlag: Franzis´ Verlag München, 1994

[Lotus 1995]
o.V.: Die Lotus Kommunikationsstrategie;
Lotus Development GmbH München, Februar 1995

[Nastansky 1990a]
Nastansky, L.: Trends auf der Lotus Developer Conference `90 in Boston und Workgrou
orientiertes Informationsmanagement mit Notes;
In: LAC-Brief, 1990

[Nastansky 1990b]
Nastansky, L.: Softwarewerkzeuge für Endbenutzer;
In: Handbuch Wirtschaftsinformatik;
Hrsg. K. Kurbel, H. Strunz, Poeschel, 1990

[Nastansky 1994]
Nastansky, L.: Büroinformationssysteme;
In: Fischer, J., Herold, W., Dangelmaier, W., Nastansky, L., Wolff, R.: Bausteine der
Wirtschaftsinformatik, S+W Steuer- und Wirtschaftsverlag Hamburg, 1994

[Niemeier, Schäfer, Engstler, Kroll 1994]
Niemeier, J., Schäfer, M., Engstler, M., Kroll, P.: Mobile Computing;
Informationstechnologie ortsungebunden nutzen;
Technik - Einsatz - Wirtschaftlichkeit;
Verlag: Computerwoche Verlag München, 1994

[Niemeier, Schäfer 1993]
Niemeier, J., Schäfer, M.: Mobile Computing - Mobile Datenkommunikation;
In: Wirtschaftsinformatik, 35 3/1993, Vieweg Verlag Wiesbaden, S. 73-77

[Novell 1995]
o.V.: Novell BUYER's GUIDE und diverse Produktbroschüren zu GroupWare-Lösungen
von Novell;
1995

[Österle, Riehm 1994]
Österle, H., Riehm, R.: PDA;
In: Wirtschaftsinformatik, 36 3/1994, Vieweg Verlag Wiesbaden, S. 286-289

[Petrovic 92]
Petrovic, O.: Groupware - Systemkategorien, Anwendungsbeispiele, Problemfelder und
Entwicklungsstand;
In: Information Management 1/1992, S. 16-22

[Philips 1994a]
Tuttle, R.: P100 Screen Phone US Version 2.7 Functional Specification;
Philips Home Services Eindhoven, 1994

[Philips 1994b]
Johnson R.: P100 Screen Phone User Interface Style Guide;
Philips Home Services, 1994

[Philips 1994c]
o.V.: P100 Screen Phone User Interface Style Guide;
ADSI Implementation;
Philips Home Services Eindhoven, 1994

[Philips 1994d]
Johnson, R.: Country Module;
Interface Specification;
Philips Home Services Eindhoven, 7.12.1994

[Philips 1995a]
Johnson, R.: Hayes Module;
Interface Specification;
Philips Home Services Eindhoven, 15.1.1995

[Philips 1995b]
Johnson, R.: Phoneware Operating Services;
(POS) Interface Specification;
Philips Home Services Eindhoven, 18.1.1995

[Philips 1995c]
Johnson, R.: Basic Input/Output Module;
(BIOS) Interface Specification;
Philips Home Services Eindhoven, 23.2.1995

[Philips 1995d]
Johnson, R.: Application Developer's Guide;
Philips Home Services Eindhoven, 23.2.1995

[Preßmar 1994]
Preßmar, D. B.: Künftige Computeranwendungen für mobile Nutzer
In: it+ti (4/5)/1994, S. 105-109

[Reder 1996]
Reder, B.: Datenfunk;
Mobitex Flächendeckung bis 1997;
In: Gateway 4/1996, Heinz Heise Verlag Hannover, S. 58-61

[Robinson 1993]
Robinson, M.: Keyracks and computers: an introduction to "common artefact" in Computer Supported Cooperative Work (CSCW);
In: Wirtschaftsinformatik 2/1993, Vieweg Verlag Wiesbaden, S. 157-166

[Schneier 1996]
Schneier, B.: Applied Cryptography, Second Edition;
Protocols, Algorithms, and Source Code in C;
Verlag: John Wiley & Sons Inc. New York, 1996

[Schirmer 1992]
Schirmer, C.: Die Programmiersprache C;
Die ausführliche Beschreibung aller Sprachelemente;
Verlag: Carl Hanser Verlag München, 1992

[Sievers 1996]
Sievers, D.: Architekturen und Anwendungskonzepte von Groupware in der mobilen Kommunikation;
Generische Entwicklung von Kommunikationssteuerungsmodulen und Datenrepositories;
Diplomarbeit an der Universität-Gesamthochschule Paderborn, Lehr- und Forschungseinheit Wirtschaftsinformatik 2, 1996

[Tanenbaum 1994]
Tanenbaum, A. S.: Moderne Betriebssysteme;
Verlag: Coedition der Verlage Carl Hanser Verlag München und Prentice-Hall International London, 1994

[Toker 1995]
Toker, S.: The Integration of Mobile Telecommunications in Europe;
In: Mobile Telecommunications: Emerging European Markets;
Hrsg. Schenk, K.-E., Müller, J., Schnöring, T., Artech House Publishers London, 1995

[Weber 1996]
Weber, V.: Netzwerk im Visier;
IBMs neue Strategie: offen und plattformübergreifend ins Netz;
In: c`t 4/1996, Heinz Heise Verlag Hannover, S. 162-163

[Wong, Britland 1995]
Wong, P., Britland, D.: Mobile Data Communications Systems;
Verlag: Artech House Publishers London, 1995

[Vertegaal, Guest 1995]
Vertegaal, R., Guest, S.: Network Issues in the Growth and Adoption of Networked CSCW Services;
In: SIGCHI-Bulletin Vol. 27 No 4 Oct. 1995, S. 63-67

[Zimmermann, Bayard 1993]
Zimmermann, F., Bayard, B.: Daten funken;
Modacom - Telecom Datenfunkdienst;
In: c`t 6/1993, Heinz Heise Verlag Hannover, S. 62-66 Abkürzungsverzeichnis

Anhang B: Schnittstellen der Notes-Kapsel

Datenbankfunktionen

usAVASNotes_DbOpen-- Öffnen einer Datenbank.

Syntax:
```
unsigned short   usAVASNotes_DbOpen (szDbName, pusDbHandle);
char             *szDbName;         /* Input  */
unsigned short   *pusDbHandle;      /* Output */
```

Beschreibung: Öffnen der durch den Namen spezifizierten Notes-Datenbank, Erzeugung eines zugehörigen Kontextes und Rücklieferung des Kontexhandles.

Parameter:

szDbName
Vollqualifizierter Name der Notes-Datenbank.

pusDbHandle
Aufnahmepuffer für den Kontexthandle.

Ergebnis: unsigned short, ein MobileNotes-Fehlercode.

Beispiel:
```
unsigned short usHd  = AVA_NOTES_INVALID_HANDLE;
unsigned short usRet = AVA_GEN_OK;

usRet = usAVASNotes_DbOpen("PBWI2B!!C:\NAMES.NSF", &usHd);
```

Verweise: usAVASNotes_DbClose

usAVASNotes_DbClose-- Schließen einer geöffneten Datenbank.

Syntax:
```
unsigned short   usAVASNotes_DbClose (pusDbHandle);
unsigned short   *pusDbHandle;      /* Input, Output */
```

Beschreibung: Schließen einer geöffneten Datenbank, entfernen des zugehörigen Kontextes und Unkenntlichmachung des Kontexthandles.

Parameter:

pusDbHandle
Kontexthandle der zu schließenden Datenbank, wird mit Funktionsabschluß auf definierten 'Ungültig-Wert' gesetzt.

Ergebnis: unsigned short, ein MobileNotes-Fehlercode.

Beispiel:
```
unsigned short usRet = AVA_GEN_OK;

usRet = usAVASNotes_DbClose (&usHd);
```

Verweise: usAVASNotes_DbOpen

Ansichtfunktionen

usAVASNotes_ViewOpen-- Öffnen einer Ansicht

Syntax:
```
unsigned short   usAVASNotes_ViewOpen (pusDbHandle, sz  ViewName);
unsigned short   pusDbHandle;       /* Input */
char             *szViewName;       /* Input */
```

Beschreibung: Öffnen einer Ansicht in einem Kontext, eine ggf. geöffnete Ansicht wird vorher automatisch geschlossen.

Parameter:

pusDbHandle
Kontexthandle der betroffenen Datenbank.

szViewName
Originalname der Ansicht, der Name der Notes-GUI kann nicht verwendet werden.

Ergebnis: unsigned short, ein MobileNotes-Fehlercode.

Beispiel:
```
unsigned short usRet = AVA_GEN_OK;

usRet = usAVASNotes_ViewOpen (usHd,"Alles nach Datum");
```

Verweise: usAVASNotes_ViewGetInfo

usAVASNotes_ViewGetInfo-- Informationen zu einer Ansicht einholen

Syntax:
```
unsigned short   usAVASNotes_ViewGetInfo (pusDbHandle, pulDocNo,
                                          pulCategoryNo);
unsigned short   pusDbHandle;       /* Input  */
unsigned long    *pulDocNo;         /* Output */
unsigned long    *pulCategoryNo;    /* Output */
```

Beschreibung: Anzahl der Dokumente und Kategorien in einer Ansicht einholen.

Parameter:

pusDbHandle
Kontexthandle der betroffenen Datenbank.

pulDocNo
Aufnahmepuffer für die Anzahl der enthaltenen Dokumente.

pulCategoryNo
Aufnahmepuffer für die Anzahl der enthaltenen Kategorien.

Ergebnis: unsigned short, ein MobileNotes-Fehlercode.

Beispiel:
```
unsigned long  ulDoc = 0;
unsigned long  ulCat = 0;
unsigned short usRet = AVA_GEN_OK;

usRet   = usAVASNotes_ViewGetInfo (usHd,&ulDoc,&ulCat);
```

Verweise: usAVASNotes_ViewOpen

Dokumentfunktionen

usAVASNotes_DocFirst-- Navigation auf das erste Dokument einer geöffneten Ansicht

Syntax:
```
unsigned short   usAVASNotes_DocFirst (pusDbHandle);
unsigned short   pusDbHandle;       /* Input */
```

Beschreibung: Navigation auf das erste Dokument einer geöffneten Ansicht, eventuelle Veränderungen an einem vorher focussierten Dokument werden in der Datenbank fixiert.

Parameter:
pusDbHandle
Kontexthandle der betroffenen Datenbank.

Ergebnis: unsigned short, ein MobileNotes-Fehlercode.

Beispiel:
```
unsigned short usRet = AVA_GEN_OK;

usRet = usAVASNotes_DocFirst (usHd);
```

Verweise: usAVASNotes_DocLast, usAVASNotes_DocNext, usAVASNotes_DocPrev, usAVASNotes_DocById, usAVASNotes_DocByNo

usAVASNotes_DocLast-- Navigation auf das letzte Dokument einer geöffneten Ansicht

Syntax:
```
unsigned short   usAVASNotes_DocLast (pusDbHandle);
unsigned short   pusDbHandle;       /* Input */
```

Beschreibung: Navigation auf das letzte Dokument einer geöffneten Ansicht, eventuelle Veränderungen an einem vorher focussierten Dokument werden in der Datenbank fixiert.

Parameter:
pusDbHandle
Kontexthandle der betroffenen Datenbank.

Ergebnis: unsigned short, ein MobileNotes-Fehlercode.

Beispiel:
```
unsigned short usRet = AVA_GEN_OK;

usRet = usAVASNotes_DocLast (usHd);
```

Verweise: usAVASNotes_DocFirst, usAVASNotes_DocNext, usAVASNotes_DocPrev, usAVASNotes_DocById, usAVASNotes_DocByNo

usAVASNotes_DocNext-- Navigation auf das nächste Dokument einer geöffneten Ansicht

Syntax:
```
unsigned short   usAVASNotes_DocNext (pusDbHandle);
unsigned short   pusDbHandle;       /* Input */
```

Beschreibung: Navigation auf das nächste Dokument einer geöffneten Ansicht, eventuelle Veränderungen an einem vorher focussierten Dokument werden in der Datenbank fixiert.

Parameter:
pusDbHandle
Kontexthandle der betroffenen Datenbank.

Ergebnis: unsigned short, ein MobileNotes-Fehlercode.

Beispiel:
```
unsigned short usRet = AVA_GEN_OK;

usRet = usAVASNotes_DocNext (usHd);
```

Verweise: usAVASNotes_DocFirst, usAVASNotes_DocLast, usAVASNotes_DocPrev, usAVASNotes_DocById, usAVASNotes_DocByNo

usAVASNotes_DocPrev-- Navigation auf das vorhergehende Dokument einer geöffneten Ansicht

Syntax:
```
unsigned short   usAVASNotes_DocPrev (pusDbHandle)  ;
unsigned short   pusDbHandle;       /* Input */
```

Beschreibung: Navigation auf das vorhergehende Dokument einer geöffneten Ansicht, eventuelle Veränderungen an einem vorher focussierten Dokument werden in der Datenbank fixiert.

Parameter:

pusDbHandle
Kontexthandle der betroffenen Datenbank.

Ergebnis: unsigned short, ein MobileNotes-Fehlercode.

Beispiel:
```
unsigned short usRet = AVA_GEN_OK;

usRet = usAVASNotes_DocPrev (usHd);
```

Verweise: usAVASNotes_DocFirst, usAVASNotes_DocLast, usAVASNotes_DocNext, usAVASNotes_DocById, usAVASNotes_DocByNo

usAVASNotes_DocById-- Navigation auf ein Dokument anhand der Dokumenten-ID

Syntax:
```
unsigned short   usAVASNotes_DocById (pusDbHandle, ulDocId,
                                      pcUnId);
unsigned short   pusDbHandle;       /* Input */
unsigned long    ulDocId;           /* Input */
char             *pcUnId;           /* Input */
```

Beschreibung: Navigation auf ein Dokument in einer geöffneten Ansicht anhand der Dokumenten- oder Note-ID, eventuelle Veränderungen an einem vorher focussierten Dokument werden in der Datenbank fixiert.

Parameter:

pusDbHandle
Kontexthandle der betroffenen Datenbank.

ulDocId
Dokumenten ID des aufzusuchenden Dokuments, die Verwendung von ulDocId und pcUnId is alternativ. Im Falle einer Navigation nach der universellen Note-ID zwingend Null.

pcUnId
Universelle Note-ID des aufzusuchenden Dokuments, die Verwendung von ulDocId und pcUnId ist alternativ. Im Falle einer Navigation nach der Dokumenten-ID zwingend NULL.

Ergebnis: unsigned short, ein MobileNotes-Fehlercode.

Beispiel:
```
unsigned short   usRet = AVA_GEN _OK;

usRet = usAVASNotes_DocById (usHd,4711,NULL);
```

Verweise: usAVASNotes_DocFirst, usAVASNotes_DocLast, usAVASNotes_DocNext, usAVASNotes_DocLast, usAVASNotes_DocByNo

usAVASNotes_DocByNo-- Navigation auf ein Dokument anhand der Position in der Ansicht

Syntax:
```
unsigned short   usAVASNotes_DocByNo (pusDbHandle, ulDocNo);
unsigned short   pusDbHandle;       /* Input */
unsigned long    ulDocNo;           /* Input */
```

Beschreibung: Navigation auf ein Dokument in einer geöffneten Ansicht anhand der Position in der Ansicht, eventuelle Veränderungen an einem vorher focussierten Dokument werden in der Datenbank fixiert. Es ist zu beachten, daß Kategorien bei der Positionsbestimmung nicht mitgezählt werden.

Parameter:

pusDbHandle
Kontexthandle der betroffenen Datenbank.

ulDocNo
Position des aufzusuchenden Dokuments, wird ab 1 gezählt.

Ergebnis: unsigned short, ein MobileNotes-Fehlercode.

Beispiel:
```
unsigned short   usRet = AVA_GEN_OK;

usRet = usAVASNotes_DocByNo (usHd,815);
```

Verweise: usAVASNotes_DocFirst, usAVASNotes_DocLast, usAVASNotes_DocNext, usAVASNotes_DocLast, usAVASNotes_DocById

usAVASNotes_DocGetInfo -- Informationen zu dem aktuellen Dokument einholen

Syntax:
```
unsigned short   usAVASNotes_DocGetInfo (pusDbHandle, pulDocId,
                                         pcUnId, pusLength, szTitle);
unsigned short   pusDbHandle;       /* Input          */
unsigned long    *pulDocId;         /* Output         */
char             *pcUnId;           /* Output         */
unsigned short   *pusLength;        /* Input, Output */
char             *szTitle;          /* Output         */
```

Beschreibung: Informationen zu dem gerade aktuellen Dokument einholen, dies betrifft das Dokument auf da zuletzt navigiert wurde. Zu den Informationen zählen die Dokumeten-ID, die Note-ID und der Titel im der derzeit geöffneten Ansicht.

Parameter:

pusDbHandle
Kontexthandle der betroffenen Datenbank.

pulDocId
Aufnahmeplatz für die Dokumenten-ID, nimmt bei Kategoriezeilen nach Funktionsausführung den Wert Null an.

pcUnId
Aufnahmeplatz für die universelle Note-ID.

pusLength
Größe des für den Dokumententitel bereitgestellten Speicherplatzes in Bytes, mit 0 wird der Titel nicht eingeholt. Nimmt nach Funktionsausführung die Anzahl tatsächlich benötiger Bytes auf.

szTitle
Speicherplatz für die Aufnahme des Dokumententitels, muß die Größe von pusLength haben. Ein zurückgelieferter Titel ist mit '\0' terminiert.

Ergebnis: unsigned short, ein MobileNotes-Fehlercode.

Beispiel:
```
unsigned long    ulId        = 0;
unsigned short   usLenth     = 80;
char             szPuffer[80];
char             pcUnid  [AVA_NOTE_UNID_SIZE];
unsigned short   usRet       = AVA_GEN_OK;

usRet = usAVASNotes_DocGetInfo
    (usHd,&ulId,pcUnid,&usLength,szPuffer);
```

Verweise: usAVASNotes_DocNew, usAVASNotes_DocDelete

usAVASNotes_DocNew-- Erzeugen eines neuen Dokumentes

Syntax:
```
unsigned short   usAVASNotes_DocNew (pusDbHandle, szFormName,
                                          pulDocId, pcUnId);
unsigned short   pusDbHandle;      /* Input  */
char             *szFormName;      /* Input  */
unsigned long    *pulDocId;        /* Output */
char             *pcUnId;          /* Output */
```

Beschreibung: Erzeugen eines neuen Dokumentes auf Basis einer vorgegebenen Maske, die durch Navigation in der Ansicht erreichte Position wurde nicht verändert, aber dennoch befindet sich das neu erzeugte Dokument im Focus.

Parameter:

pusDbHandle
Kontexthandle der betroffenen Datenbank.

szFormName
Originalname der Ansicht, der Name der Notes-GUI kann nicht verwendet werden.

pulDocId
Aufnahmeplatz für die Dokumenten-ID des neuen Dokumentes.

pcUnId
Aufnahmeplatz für die universelle Note-ID des neuen Dokumentes.

Ergebnis: unsigned short, ein MobileNotes-Fehlercode.

Beispiel:
```
unsigned long    ulDocId;
char             pcUnId[AVA_NOTE_UNID_SIZE];
unsigned short   usRet;

usRet = usAVASNotes_DocNew (usHd,"Reply",&ulDocId,pcUnId);
```

Verweise: usAVASNotes_DocGetInfo, usAVASNotes_DocDelete

usAVASNotes_DocDelete-- Löschen eines Dokumentes

Syntax:
```
unsigned short   usAVASNotes_DocDelete (pusDbHandle);
unsigned short   pusDbHandle;      /* Input  */
```

Beschreibung: Löschen des aktuell focussierten Dokumentes.

Parameter:

pusDbHandle
Kontexthandle der betroffenen Datenbank.

Ergebnis: unsigned short, ein MobileNotes-Fehlercode.

Beispiel:
```
unsigned short   usRet;

usRet = usAVASNotes_DocDelete (usHd);
```

Verweise: usAVASNotes_DocGetInfo, usAVASNotes_DocNew

Feldfunktionen

usAVASNotes_FieldGetLength -- Länge des Inhaltes eines Feldes einholen

Syntax:
```
unsigned short   usAVASNotes_FieldGetLength (pusDbHandle,
                                             szFieldName, pulLength);
unsigned short   pusDbHandle;       /* Input  */
char             *szFieldName;      /* Input  */
unsigned long    *pulLength;        /* Output */
```

Beschreibung: Länge der in einem bestimmten Feld enthaltenen Daten einholen, diese Abfrage bezieht sich auf das aktuelle Dokument.

Parameter:

pusDbHandle
Kontexthandle der betroffenen Datenbank.

szFieldName
Name des abzufragenden Feldes.

pulLength
Aufnahmeplatz für die ermittelte Länge.

Ergebnis: unsigned short, ein MobileNotes-Fehlercode.

Beispiel:
```
unsigned long  ulLength      = 0;
unsigned short usRet         = AVA_GEN_OK;

usRet = usAVASNotes_FieldGetLength (usHd,"From",&ulLength);
```

Verweise: usAVASNotes_FieldGetContents, usAVASNotes_FieldSet

usAVASNotes_FieldGetContents -- Inhalt eines Feldes einholen

Syntax:
```
unsigned short   usAVASNotes_FieldGetContents (pusDbHandle,
                                 szFieldName, pulLength, szBuf);
unsigned short   pusDbHandle;       /* Input        */
char             *szFieldName;      /* Input        */
unsigned long    *pulLength;        /* Input,Output */
char             *szBuf;            /* Output       */
```

Beschreibung: Daten eines bestimmten Feldes einholen, diese Abfrage bezieht sich auf das aktuelle Dokument. Alle eingeholten Daten werden in dem Format 'ASCII-Text' zurueckgegeben.

Parameter:

pusDbHandle
Kontexthandle der betroffenen Datenbank.

szFieldName
Name des abzufragenden Feldes.

pulLength
Größe des für den Feldinhalt bereitgestellten Speicherplatzes in Bytes. Nimmt nach Funktionsausführung die Anzahl tatsächlich benötiger Bytes auf.

szBuf
Für den Feldinhalt bereitgestellter Puffer, muß mindestens die Größe pulLength aufweisen

Ergebnis: unsigned short, ein MobileNotes-Fehlercode.

Beispiel:
```
unsigned long  ulLength      = 80;
char           szPuffer [80];
unsigned short usRet         = AVA_GEN_OK;

usRet = usAVASNotes_FieldGetContents
                (usHd,"From",&ulLength,szPuffer);
```

Verweise: usAVASNotes_FieldGetLength, usAVASNotes_FieldSet

usAVASNotes_FieldSet-- Inhalt eines Feldes setzen

Syntax:
```
unsigned short   usAVASNotes_FieldSet (pusDbHandle, szFieldName,
                                       usLength, szBuf);
unsigned short   pusDbHandle;       /* Input  */
char             *szFieldName;      /* Input  */
unsigned short   usLength;          /* Input  */
char             *szBuf;            /* Output */
```

Beschreibung: Inhalt eines bestimmten Feldes besetzen, diese Aktion bezieht sich auf das aktuelle Dokument. Alle Daten werden aus dem Format 'ASCII-Text' automatisch in das Zielformat gewandelt.

Parameter:

pusDbHandle
Kontexthandle der betroffenen Datenbank.

szFieldName
Name des zu besetzenden Feldes.

pusLength
Größe des für den Feldinhalt bereitgestellten Speicherplatzes in Bytes.

szBuf
Neuer Feldinhalt.

Ergebnis: unsigned short, ein MobileNotes-Fehlercode.

Beispiel:
```
unsigned short usLength      = 6;
char           szPuffer[]    = "AvALoN";
unsigned short usRet         = AVA_GEN_OK;

usRet = usAVASNotes_FieldSet (usHd,"From",usLength,szPuffer);
```

Verweise: usAVASNotes_FieldGetLength, usAVASNotes_FieldGetContents

Mailfunktion

usAVASNotes_Mail- Erstellen einer Mail

Syntax:

```
unsigned short   usAVASNotes_Mail (szDb, szTo, szCc, szBcc,
                                   szSubject, szBody);
char             *szDb;            /* Input  */
char             *szTo;            /* Input  */
char             *szCc;            /* Input  */
char             *szBcc;           /* Input  */
char             *szSubject;       /* Input  */
char             *szBody;          /* Input  */
```

Beschreibung: Erstellen und Versenden einer Mail.

Parameter:

szDb
Vollqualifizierter Datenbankname.

szTo
Adressatenliste.

szCc
Carbon Copy.

szBcc
Black Carbon Copy.

szSubject
Betreff.

szBody
Nachrichtentext.

Ergebnis: unsigned short, ein MobileNotes-Fehlercode.

Beispiel:

```
unsigned short usRet = AVA_GEN_OK;

usRet = usAVASNotes_Mail ("mail.box","Nico
                          Dirks",NULL,NULL,"AVASNOTE","...");
```

Verweise: -

Anhang C: Schnittstellen für die Modemkommunikation

usAVASCom_Poll-- Anrufentgegennahme initiieren

Syntax:

```
unsigned short  usAVASCom_Poll(szComPort, szInitString,
                               ulReadTimeout, ulWriteTimeout);
char            *szComPort;                          /* Input */
char            *szInitString;                       /* Input */
unsigned long   ulReadTimeout;                       /* Input */
unsigned long   ulWriteTimeout;                      /* Input */
unsigned short  (*usHandleCom)(char *szComPort));    /* Input */
```

Beschreibung: Initialisierung des Modems. Thread erzeugen, welcher den COM-Port pollt und Aufrufen der angegebenen Funktion bei eingehendem Telefonanruf.

Parameter:

szComPort
Vollqualifizierter COM-Port-Name, ohne abschliessenden Doppelpunkt.

szInitString
Initialisierungsstring fuer das Modem.

ulReadTimeout
Maximalzeit (in ms) zum einlesen eines Datenblocks in der gewuenschten Groesse.

ulWriteTimeout
Maximalzeit (in ms) zum schreiben eines Datenblocks in der gewuenschten Groesse.

usHandleCom
Zeiger auf Funktion, die im Falle eines erfolgreichen Verbindungsaufbaus aufgerufen wird.

Ergebnis: unsigned short, ein MobileNotes-Fehlercode.

Beispiel:

```
unsigned short usRet = AVA_GEN_OK;

usRet = usAVASCom_Poll ("COM1","ATZ"1000,1000,usClientThread);
```

Verweise: usAVASCom_HangUp

usAVASCom_HangUp- Verbindung unterbrechen

Syntax:

```
unsigned short  usAVASCom_HangUp (szComPort);
char            *szComPort;  /* Input */
```

Beschreibung: Trennen einer bestehenden Verbindung.

Parameter:

szComPort
vollqualifizierter COM-Port-Name.

Ergebnis: unsigned short, ein MobileNotes-Fehlercode.

Beispiel:

```
unsigned short  usRet   = AVA_GEN_OK;

usRet = usAVASCom_HangUP("COM1");
```

Verweise: usAVASCom_Poll

usAVASCom_IsActive-- Verbindungsstatus überprüfen

Syntax:
```
unsigned short  usAVASCom_IsAct ive (szComPort);
char            *szComPort;  /* Input */
```

Beschreibung: Ueberpruefen, ob best. COM-Port aktuell eine Verbindung hält.

Parameter:

szComPort
vollqualifizierter COM-Port-Name.

Ergebnis: unsigned short, ein MobileNotes-Fehlercode.

Beispiel:
```
unsigned short usRet = AVA_GEN_OK;

usRet = usAVASNotes_IsActive ("COM1");
```

Verweise: -

usAVASCom_Send -- Senden

Syntax:
```
unsigned short  usAVASCom_Send (szComPort, pulLength, pBuffer,
                                nUseCrc, pulKey[32]);
char            *szComPort;       /* Input */
unsigned long   *pulLength;       /* Input, Output */
void            *pBuffer;         /* Input */
int             nUseCrc;          /* Input */
unsigned long   pulKey[32];       /* Input */
```

Beschreibung: Versenden aller Zeichen im Puffer.

Parameter:

szComPort
Vollqualifizierter COM-Port-Name.

pulLength
Pufferlänge, nimmt abschließend Anzahl abgesendeter Bytes auf.

pBuffer
Puffer.

nUseCrc
Indikator, ob CRC angewandt werden soll (0-Nein, sonst-Ja).

pulKey
Ggf. vorbehandelter DES-Schlüssel für Verschlüsselung, sonst NULL.

Ergebnis: unsigned short, ein MobileNotes-Fehlercode.

Beispiel:
```
unsigned short  usRet    = AVA_GEN_OK;
unsigned long   ulLength = 10;

usRet = usAVASCom_Send("COM1",&ulLength,"Nico Dirks",1,NULL);
```

Verweise: usAVASCom_Receive

usAVASCom_Receive- Empfangen

Syntax:

```
unsigned short  usAVASCom_Receive (szComPort, pulLength,
                                   pBuffer, pcStopChar, pulKey);
char            *szComPort;       /* Input  */
unsigned short  *pulLength;       /* Input, Output */
void            *pBuffer;         /* Output */
char            *pcStopChar;      /* Input  */
unsigned long   pulKey[32];       /* Input  */
```

Beschreibung: Empfangen von Daten bis Timer abgelaufen, Puffer voll oder Grenzzeichen empfangen.

Parameter:

szComPort
Vollqualifizierter COM-Port-Name.

pulLength
Pufferlänge, nimmt abschließend Anzahl empfangener Bytes auf.

pBuffer
Puffer der Länge pulLength.

pcStopChar
Grenzzeichen, bei dem der Empfang unterbrochen wird (NULL - kein Grenzzeichen).

pulKey
ggf. vorbehandelter DES-Schlüssel für Entschlüsselung, sonst NULL.

Ergebnis: unsigned short, ein MobileNotes-Fehlercode.

Beispiel:

```
unsigned short usRet    = AVA_GEN_OK;
unsigned long  ulLength = 20;
void           *pBuffer;

pBuffer   = malloc (ulLength);
usRet     = usAVASCom_Receive("COM1",&ulLength, pBuffer,
                                     NULL, NULL);
```

Verweise: usAVASCom_Send

Anhang D: Schnittstellen für die Fehlererkennung

usAVABCrc_Encode-- CRC anwenden

Syntax:
```
unsigned short   usAVABCrc_Encode (usBuf, usBufLen);
unsigned short   *usBuf;      /* Input */
unsigned short   usBufLen;    /* Input */
```

Beschreibung: Speicherblock via CRC-Verfahren kodieren.

Parameter:

usBuf
Zu kodierender Speicherblock

usBufLen
Anzahl zu kodierender Bytes (Vielfaches von Zwei).

Ergebnis: CRC-Code des Speicherblocks.

Beispiel:
```
unsigned short usCrc = 0;

usCRC = usAVABCrc_Encode (pBuffer,10);
```

Verweise: -

Anhang E: Schnittstellen der graphischen Benutzungsoberfläche

usAVACDisp_Init -- Initialisierung.

Syntax: `unsigned short usAVACDisp_Init (void);`

Beschreibung: Initialisieren der Bildschirmschnittstelle, richtet zyklisch Statusanzeige ein.

Parameter: -

Ergebnis: unsigned short, ein MobileNotes-Fehlercode.

Beispiel:
```
unsigned short usRet = AVA_GEN_OK;

usRet = usAVACDisp_Init ();
```

Verweise: usAVACDisp_Term

usAVACDisp_Term -- Terminierung.

Syntax: `unsigned short usAVACDisp_Term (void);`

Beschreibung: Terminieren der Bildschirmschnittstelle.

Parameter: -

Ergebnis: unsigned short, ein MobileNotes-Fehlercode.

Beispiel:
```
unsigned short usRet = AVA_GEN_OK;

usRet = usAVACDisp_Term ();
```

Verweise: usAVACDisp_Init

usAVACDisp_InputMask -- Eingabemaske.

Syntax:
```
unsigned short   usAVACDisp_InputMask (hmTitle, hmSubTitle,
    hmHelpText, nFields, hmFieldDef, nAllowEdit, nAnzFuncKey,
    hmszFuncKeyNames, funcFuncCalls);
HMEM              hmTitle;               /* Input */
HMEM              hmSubTitle;            /* Input */
HMEM              hmHelpText;            /* Input */
int               nFields;               /* Input */
HMEM              hmFieldDef;            /* Input, Output */
int               nAllowEdit;            /* Input */
int               nAnzFuncKey;           /* Input */
HMEM              hmszFuncKeyNames[];    /* Input */
AVACDISPMASKFUNC funcFuncCalls[];        /* Input */
```

Beschreibung: Stellt auf dem Bildschirm eine Maske dar und handelt ggf. die diesbezügl. Eingaben. Diese Maske kann sowohl zur Ansicht, Eingabe und Veränderung verwendet werden.

Parameter:

hmTitle
Titel der Maske.

hmSubTitle
Unteritel der Maske.

hmHelpText
Hilfetext zu dieser Maske.

nFields
Anzahl der folgenden Felddefinitionen.

hmFieldDef
Felddefinitionen (Zeiger auf Array vom Typ AVAFIELDDEF).

nAllowEdit
Anzeigemodus (1 - Editiermodus, 0 - Anzeigemodus).

nAnzFuncKey
Anzahl frei definierter Funktionstasten (max. 4 werden angezeigt).

hmszFuncKeyNames
Beschriftung der Funktionstasten

funcFuncCalls
Hinter den Funktionstasten liegende Funktionen.

Ergebnis: unsigned short, ein MobileNotes-Fehlercode.

Beispiel: -

Verweise: -

usAVACDisp_Menu-- Auswahlmenü.

Syntax:

```
unsigned short   usAVACDisp_Menu(hmTitle, hmSubTitle, hmHelpText,
          hmIndex, hmFuncKey, nAnzFuncKey, hmszFuncKeyNames,
          funcFuncCalls,( *nList), (*ucSelect));
HMEM              hmTitle;               /* Input */
HMEM              hmSubTitle;            /* Input */
HMEM              hmHelpText;            /* Input */
HMEM              hmIndex[];             /* Input */
HMEM              hmFuncKey;             /* Output */
int               nAnzFunc Key;          /* Input */
HMEM              hmszFuncKeyNames[];    /* Input */
AVACDISPMASKFUNC funcFuncCalls[];        /* Input */
int               (*nList)(...)          /* Input */
unsigned char     (*ucSelect)(...)       /* Input */
```

Beschreibung: Stellt auf dem Bildschirm eine Maske dar und handelt ggf. die diesbezügl. Eingaben. Diese Maske kann sowohl zur Ansicht, Eingabe und Veränderung verwendet werden.

Parameter:

hmTitle
Titel der Maske.

hmSubTitle
Unteritel der Maske.

hmHelpText
Hilfetext zu dieser Maske.

hmFuncKey
Nimmt ggf. die Nr. der Funktionstaste auf, welche gedrückt wurde (1-5). Allerdings nur dann, wenn dies zum Verlassen des Menüs führte.

hmIndex
Array der darzustellenden Menüpunkte.

nAnzFuncKey
Anzahl frei definierter Funktionstasten (max. 4 werden angezeigt).

hmszFuncKeyNames
Beschriftung der Funktionstasten

funcFuncCalls
Hinter den Funktionstasten liegende Funktionen.

Ergebnis: unsigned short, ein MobileNotes-Fehlercode.

Beispiel: -

Verweise: -

usAVACDisp_DisplayBlock-- Textdarstellung.

Syntax:
```
unsigned short   usAVACDis p_DisplayBlock(hmTitle, hmSubTitle,
                                          hmString, hmszHelpString);
HMEM        hmTitle;                /* Input */
HMEM        hmSubTitle;             /* Input */
HMEM        hmString;               /* Input */
HMEM       .hmHelpString;           /* Input */
```

Beschreibung: Darstellung eines Textblocks.

Parameter:

hmTitle
Titel der Maske.

hmSubTitle
Untertitel der Maske.

hmString
Darzustellender String.

hmszHelpStringt
Ggf. vorhandener Hilfetext, sonst hNULL.

Ergebnis: unsigned short, ein MobileNotes-Fehlercode.

Beispiel: -

Verweise: -

usAVACDisp_Message -- Nachrichtenbox.

Syntax:
```
unsigned short   usAVACDisp_Message(hmMessage);
HMEM              hmMessage;  /* Input */
```

Beschreibung: Stellt eine Statusinformation oder Nachricht auf dem Bildschirm dar, bzw. löscht sie wieder.

Parameter:

hmMessage
Nachricht die auszugeben ist oder hNULL. Bei hNULL wird die Nachricht wieder gelöscht.

Ergebnis: unsigned short, ein MobileNotes-Fehlercode.

Beispiel:
```
unsigned short   usRet = AVA_GEN_OK;

usRet = usAVACDisp_Message (AHANDLE("Server did not respond"));
```

Verweise: -

usAVACDisp_GetString -- Promptbox.

Syntax:
```
unsigned short   usAVACDisp_GetString(hmMessage, hmString,
                                                  hmLength, nShowMode);
HMEM       hmMessage;   /* Input */
HMEM       hmString;    /* Input, Output */
HMEM       hmLength;    /* Input, Output */
int        nShowMode;   /* Input */
```

Beschreibung: Stellt eine Statusinformation oder Nachricht auf dem Bildschirm dar und erlaubt dazu eine Benutzereingabe.

Parameter:

hmMessage
Nachricht die auszugeben ist.

hmString
Defaultwert und Puffer für eingegebene Zeichenkette.

hmLength
Nimmt bei Funktionsaufruf die Pufferlänge und bei Funktionsende die Eingabelänge auf.

nShowMode
Festlegen, ob Eingaben dargestellt werden (1-Ja, 0 - Nein, d.h Eingaben werden durch '*' dargestellt)

Ergebnis: unsigned short, ein MobileNotes-Fehlercode.

Beispiel:
```
unsigned short   usRet       = AVA_GEN_OK;
char       *szInput    = "Nico Dirks";
unsigned short   usLength    = 10;

usRet = usAVACDisp_GetString
(AHANDLE("Password"),AHANDLE(szInput), AHANDLE(&usLength),0);
```

Verweise: -

Anhang F: Schnittstellen der Tastaturbehandlung

usAVACKey_Init -- Initialisierung.

Syntax: `unsigned short usAVACKey_Init (void);`

Beschreibung: Initialisieren der Tastaturschnittstelle, richtet den Bildschirmschoner ein.

Parameter: -

Ergebnis: unsigned short, ein MobileNotes-Fehlercode.

Beispiel:
```
unsigned short usRet = AVA_GEN_OK;
usRet = usAVACKey_Init ();
```

Verweise: usAVACKey_Term

usAVACKey_Term -- Terminierung.

Syntax: `unsigned short usAVACKey_Term (void);`

Beschreibung: Terminieren der Tastaturschnittstelle.

Parameter: -

Ergebnis: unsigned short, ein MobileNotes-Fehlercode.

Beispiel:
```
unsigned short usRet = AVA_GEN_OK;
usRet = usAVACKey_Term ();
```

Verweise: usAVACKey_Init

usAVACKey_GetHit - Tastenanschlag holen.

Syntax: `unsigned short usAVACKey_GetHit (void)`

Beschreibung: Tastenanschlag einholen

Parameter: -

Ergebnis: Tastencode im P100-Format (d.h. High: Scan-Code/Low: Character)

Beispiel:
```
unsigned short usHit;
usHit = usAVACKey_GetHit();
```

Verweise: -

usAVACKey_GetString -- Zeichenkette einholen.

Syntax:
```
unsigned char   ucAVACKey_GetString (ucColors, ucLength,
                                     cStopChar, hmBuf, nShowMode)
unsigned char   ucColors;   /* Input */
```

```
unsigned char    ucLength;    /* Input */
char             cStopChar;   /* Input */
HMEM             hmBuf;       /* Input, Output */
int              nShowMode;   /* Input */
```

Beschreibung: Einholen eines Strings über die Tastatur, mit gleichzeitiger Darstellung der Eingaben.

Parameter:

ucColors
Farbencode für Hinter-/Vordergrundfarbe.

ucLength
Maximale Eingabelänge, d.h. Länge des Puffers.

cStopChar
Grenzzeichen, bis zu dem eingelesen wird.

hmBuf
Enthält zunächst den Default-Wert für den Eingabepuffer, enthält abschließend die eingegebene Zeichenkette.

nShowMode
Gibt an, ob Eingaben in Klartext angegeben oder durch '*' symbolisiert werden. (0 für die Verschlüsselung)

Ergebnis: reale Eingabelänge.

Beispiel:

```
unsigned short ucLength = 10;

ucLengh = ucAVACKey_GetString(0x47,ucLength,"Nico Dirks",0);
```

Verweise: -

Anhang G: Schnittstellen der Speicherverwaltung

usAVACMem_Init -- Initialisierung.

Syntax:
```
unsigned short   usAVACMem_Init (int nMode);
```

Beschreibung: Initialisieren des Speichersystems.

Parameter:

nMode
Bestimmt den Initialisierungsmodus:
AVA_MEM_GARBAGE_COLLECTION - bestehende Speicherstrukturen übernehmen
AVA_MEM_INIT - Speicher von Grund auf neu initialisieren

Ergebnis: unsigned short, ein MobileNotes-Fehlercode.

Beispiel:
```
unsigned short usRet = AVA_GEN_OK;

usRet = usAVACMem_Init (AVA_MEM_INIT);
```

Verweise: -

usAVACMem_malloc -- Allokation.

Syntax:
```
unsigned short   usAVACMem_malloc (ulSize, hMem, ucMode);
size_t           ulSize;     /* Input */
HMEM             *hMem;      /* Output */
unsigned char    ucMode;     /* Input */
```

Beschreibung: Speicherblock gewünschter Größe allozieren.

Parameter:

ulSize
Größe des gewünschten Speicherblocks in Anzahl Bytes.

hMem
Aufnahmecontainer für das zurückgelieferte Speicherhandle.

ucMode
Allokationsmodus, ALLOC_PERSISTENT für persistente Speicherblöcke, ALLOC_NON_PERSISTENT für temporäre Speicherblöcke.

Ergebnis: unsigned short, ein MobileNotes-Fehlercode.

Beispiel:
```
unsigned short usRet = AVA_GEN_OK;
HMEM           hMem;

usRet = usAVACMem_malloc (1000,&hMem,ALLOC_PERSISTENT);
```

Verweise: usAVACMem_free

usAVACMem_free -- Deallokation.

Syntax:
```
unsigned short   usAVACMem_free(hMem);
HMEM       *hMem;       /* Input, Output */
```

Beschreibung: Speicherblock wieder freigeben.

Parameter: **hMem**
Handle des freizugebenden Speicherbereiches.

Ergebnis: unsigned short, ein MobileNotes-Fehlercode.

Beispiel:
```
unsigned short uRet = AVA_GEN_OK;
usRet = usAVACMem_free (&hMem);
```

Verweise: usAVACMem_malloc