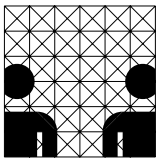
Diplomarbeit

Gestaltung von Anwendungssoftware nach dem WAM-Ansatz auf mobilen Geräten

Universität Hamburg
Fachbereich Informatik
Vogt-Kölln-Straße 30
D-22527 Hamburg



Juli 2003

Joachim Sauer

Matr.# 4920432

6sauer@informatik.uni-hamburg.de

Erstbetreuung: Prof. Dr. Heinz Züllighoven
Zweitbetreuung: Prof. Dr. Horst Oberquelle

Betreuung:

Prof. Dr. Heinz Züllighoven (Erstbetreuer)
Arbeitsbereich Softwaretechnik (SWT)
Fachbereich Informatik
Universität Hamburg
Vogt-Kölln-Straße 30
D-22527 Hamburg

Prof. Dr. Horst Oberquelle (Zweitbetreuer)
Arbeitsbereich Angewandte und sozialorientierte Informatik (ASI)
Fachbereich Informatik
Universität Hamburg
Vogt-Kölln-Straße 30
D-22527 Hamburg

„How do you fit a mountain in a teacup?“ – If your first reaction is to try to figure out a clever way to shrink the mountain, you're taking the PC approach: assume *we want to stuff everything in this product.*
The answer to the riddle is, Dig for the diamond and put that *in the teacup. After all, do you really need all the dirt and rocks?*

Rob Haitani, Produktmanager des original Palm Pilots

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1

Einleitung

1.1 Kontext und Motivation

Mobile Geräte wie Mobiltelefone und Personal Digital Assistants (PDAs) sind mittlerweile weit verbreitet. Noch werden sie bevorzugt für ihre ursprünglichen Verwendungen wie Telefonate, Versand von Kurznachrichten und typische Aufgaben wie Termin- und Adreßverwaltung genutzt. Doch zukünftig werden ihnen immer mehr auch andere Bereiche erschlossen werden. Dies wird ermöglicht durch eine rasante Weiterentwicklung der Technik, die sowohl die Verarbeitungs- und Speicherleistung der Geräte erhöht, als auch eine bessere Anbindung an bestehende Infrastrukturen, wie insbesondere das Internet, durch den Aufbau mobiler Netze mit deutlich größeren Übertragungsgeschwindigkeiten ermöglicht. Dadurch wird es möglich, auch entfernt vom stationären Arbeitsplatz auf Geschäftsdaten zuzugreifen und diese sinnvoll zu bearbeiten, ohne auf einen verhältnismäßig schweren und in der Akkulaufzeit sehr beschränkten Laptop angewiesen zu sein. Unter den Stichwörtern „Ubiquitous Computing“, „Pervasive Computing“ und mehreren anderen untersucht die Wissenschaft diese neuen Möglichkeiten.

Während die Programmierung verschiedener mobiler Geräte – sofern überhaupt möglich – bisher sehr aufwendig war, da jede Plattform über eigene Sprachen, Bibliotheken, Bedienschnittstellen und Konventionen verfügte, ändert sich diese Situation zusehends mit dem Aufkommen plattformunabhängiger Sprachen. Was bei der Verwendung dieser jungen Technologien fehlt, sind Erfahrungen im Design größerer Systeme und überzeugende Konzeptions- und Entwurfsmodelle für mobile geschäftliche Anwendungen. Diese werden umso wichtiger, als immer mehr Projekte in Unternehmen mobile Endgeräte für die Aufgabenerledigung im Außendienst einsetzen.

1.2 Zielsetzung

Der am Arbeitsbereich Softwaretechnik der Universität Hamburg entwickelte Werkzeug & Material-Ansatz (WAM-Ansatz) und seine Implementierung in dem objektorientierten Rahmenwerk JWAM haben als ein Leitbild den „Arbeitsplatz für eigenverantwortliche Expertentätigkeit“. Bisher manifestierte sich dieser Arbeitsplatz meist am stationären PC, hauptsächlich als Desktopapplikation, aber auch als Webseite im Browser.

Es stellt sich die Frage, inwieweit der WAM-Ansatz mit seinen Leitbildern und Metaphern auch für verhältnismäßig beschränkte mobile Geräte anwendbar ist und wie

er ggf. erweitert oder angepaßt werden kann. Sofern der WAM-Ansatz übertragbar ist, sind auch die Besonderheiten der Konstruktion mobiler Anwendungen zu untersuchen. Hierfür möchte ich Hilfestellungen entwickeln, insbesondere für den häufigen Fall, daß eine mobile Anwendung als Teil eines großen Anwendungssystems entwickelt wird. Dabei stellt sich die Frage, wieviel der Architektur und der Programmtexte gemeinsam genutzt werden können und ob ein solches Vorgehen überhaupt sinnvoll ist.

Eng zusammenhängend damit ist der in den letzten Jahren verstärkt zu beobachtende Trend, Geschäftslogik zu zentralisieren und über verschiedene Kanäle und Schnittstellen an unterschiedliche Endgeräte anzubinden. Dies wird als Multi-Channeling bezeichnet. In dieser Arbeit untersuche ich, wie sich mobile Endgeräte in dieses Schema einfügen und wie es in einem WAM-System realisiert werden kann.

Ich möchte auch darstellen, welche Anforderungen aus der Sicht der Software-Ergonomie bei der Erstellung der Benutzungsmodelle und Bedienoberflächen mobiler Anwendungen zu berücksichtigen sind und wie mit den, im Vergleich mit der Nutzung von PC und Workstation veränderten, Erwartungen von Benutzern an mobile Geräte umgegangen werden kann.

1.3 Methodik

Ich gehe das Thema sowohl theoretisch als auch praktisch an.

Konzepte des Mobile Computing, wie sie sich aus der einschlägigen Literatur ergeben, stellen den Ausgangspunkt meiner Untersuchungen dar. Anhand von Fachliteratur und unter Berücksichtigung aktueller Forschungsprojekte beziehe ich für die Entwicklung mobiler Anwendungen wichtige software-ergonomische Ergebnisse mit ein, um allgemeine Erkenntnisse für die Entwicklung mobiler Anwendungen den weiteren Untersuchungen zugrunde zu legen.

Der WAM-Ansatz bietet schon Konzepte für Multi-Channeling-Anwendungen unter Verwendung von Fachlichen Services. Ich betrachte die entsprechenden Veröffentlichungen und Erfahrungen der Praxis, um darauf aufbauend den WAM-Ansatz für mobile Anwendungen konzeptionell zu erweitern. Dies dient mir als Basis für eine konstruktive Umsetzung.

Die so gewonnenen Erkenntnisse wende ich in praktischen Projekten an. Feedback von anderen Entwicklern und Autor-Kritiker-Zyklen mit Anwendern sollen die Tauglichkeit im realen Einsatz testen. Insbesondere Fragen der Konstruktion mobiler Anwendungen zu bestehenden stationären Anwendungssystemen untersuche ich genauer.

Einen Überblick über wichtige Literaturquellen zu den vorgestellten Themen gebe ich in den Kapiteln, in denen ich die jeweiligen Konzepte behandle.

1.4 Inhalt der Arbeit

Zur Einführung gebe ich in Kapitel 2 einen Überblick über das Gebiet des Mobile Computing. Ich stelle bedeutende Konzepte und Forschungsrichtungen vor und beschreibe wichtige Klassen von mobilen Geräten mit ihren Charakteristika und Möglichkeiten.

Im folgenden Kapitel stelle ich wichtige software-ergonomische Grundlagen zusammen. Zuerst betrachte ich Empfehlungen allgemeiner Art. Anschließend gehe ich auf die Besonderheiten der Bedienung mobiler Geräte ein und beschreibe die Folgen der begrenzten Hardware und der besonderen Erwartungen an Consumer Devices. In diesem Zusammenhang stelle ich auch das Usage Spaces-Modell von Marcus und Chen vor. Ein- und Ausgabemöglichkeiten mobiler Geräte werden aufgrund ihrer Wichtigkeit ebenfalls betrachtet.

Die nächsten beiden Kapitel stellen den Hauptteil der Arbeit dar. In Kapitel 4 beschreibe ich die konzeptionelle Verwendung des WAM-Ansatzes zur Entwicklung mobiler Anwendungen. Dazu stelle ich wesentliche Elemente des Ansatzes kurz vor. Ich betrachte wichtige etablierte WAM-Leitbilder in bezug auf ihre Eignung für mobile Anwendungen und passe sie daran an. Genauso gehe ich auch mit den auf ihnen aufbauenden Metaphern vor. Anschließend stelle ich die für den WAM-Ansatz entwickelte Multi-Channeling-Architektur vor und erläutere Optionen der Integration mobiler Geräte darin.

Möglichkeiten der konstruktiven Umsetzung sind Thema des darauf folgenden Kapitels 5. Dabei unterscheide ich zwei Fälle. Zum einen die Entwicklung mobiler Anwendungen als Teile großer Anwendungssysteme mit stationären Komponenten, zum anderen den Fall, daß eine mobile Anwendung im Mittelpunkt des Entwicklungsprozesses steht. Ich untersuche, inwiefern im ersten Fall eine Parallelentwicklung von stationärer und mobiler Anwendung möglich ist und orientiere mich dabei eng an Entwurfsmustern des WAM-Ansatzes. Anschließend untersuche ich, welche Unterschiede im zweiten Fall auftreten können.

In Kapitel 6 stelle ich mit der Java 2 Micro Edition eine Plattform für mobile Anwendungen vor, die in unterschiedlichen Ausprägungen auf einer Vielzahl von Geräteklassen eingesetzt werden kann. Dabei steht insbesondere ihre Eignung für leistungsschwächere mobile Geräte wie intelligente Mobiltelefone, Smartphones und PDAs im Vordergrund.

Die Vorstellung von Beispielprojekten in Kapitel 7 zeigt die praktische Umsetzung wichtiger in den vorherigen Kapiteln entwickelter Konzepte.

Im abschließenden Schlußwort folgen eine Zusammenfassung der Arbeit, eine kurze Bewertung und ein Ausblick.

1.5 Danksagung

An dieser Stelle möchte ich allen danken, die mit Rat, Anregungen und Kritik zum Gelingen dieser Diplomarbeit beigetragen haben. Prof. Dr. Heinz Züllighoven danke

ich für die Erstbetreuung dieser Arbeit und viele wertvolle Ideen und Hinweise. Bei Prof. Dr. Horst Oberquelle möchte ich mich für die Zweitbetreuung und insbesondere für hilfreiche Anmerkungen zum Kapitel über Software-Ergonomie bedanken.

Martin Lippert vom Arbeitsbereich Softwaretechnik danke ich für viele Diskussionen und Feedback. Meine Kolleginnen und Kollegen von der it-wps GmbH haben sich einen Dank durch die gute Zusammenarbeit in der praktischen Umsetzung einiger hier beschriebener Konzepte verdient. Ein großer Dank gebührt auch meiner Familie, die mich jederzeit liebevoll unterstützt hat.

Kapitel 2

Überblick über Mobile Computing

2.1 Wichtige Begriffe und Konzepte

Im Bereich des Mobile Computing gibt es eine ganze Reihe von verschiedenen Forschungsrichtungen, die sich teilweise nur durch unterschiedliche Akzentuierungen unterscheiden. In diesem Abschnitt sollen historisch wichtige und zukunftsweisende davon vorgestellt werden.

Verweise auf wichtige Literaturquellen gebe ich direkt an den Stellen, an denen ich die darin beschriebenen Konzepte betrachte.

Unter Mobile Computing als Oberbegriff versteht man in der Regel die Benutzung von tragbaren Computern, die über eine drahtlose Netzanbindung verfügen (siehe [FormanZahorjan 94, S. 2]). Diese elementare Definition schränkt die mobilen Geräte schon auf solche ein, die eine online-Verbindung ermöglichen. In dieser Arbeit werden auch mobile Anwendungen betrachtet, die nur zu bestimmten Zeitpunkten mit stationären Systemen zur Synchronisation Daten austauschen. Daher wird folgende Definition verwendet:

Definition Mobile Computing:

Benutzung tragbarer Computer, die über eine drahtlose Netzanbindung verfügen oder Daten in regelmäßigen Abständen über eine direkte Verbindung mit stationären Systemen synchronisieren.

2.1.1 Ubiquitous Computing

Der zentrale Begriff des Ubiquitous Computing wurde von Mark Weiser in seinem 1991 im Scientific American erschienenen, wegweisenden Artikel „The Computer for the 21st Century“ geprägt (siehe [Weiser 91]). Weiser, der am Xerox PARC arbeitete und 1999 verstarb, sah Computer der Zukunft als ubiquitous, also allgegenwärtig. Während heutzutage Computer meist noch als solche wahrzunehmen sind, werden uns nach seiner Vision zukünftig viele unsichtbare, miteinander vernetzte Computer umgeben. Den verschwindenden Computer findet man auch unter Begriffen wie „Invisible Computing“ (nach Don Norman, [Norman 98]), „Calm Computing“ oder „Disappearing Computing“.

Beim Ubiquitous Computing (auch: UbiComp) sind Computer nicht nur miteinander vernetzt, sondern besitzen auch Informationen über ihren momentanen Ort und ihre

Umgebung. Sie passen sich den Menschen an und nicht umgekehrt. Wie Weiser schreibt ([Weiser 91, S. 104]): „Machines that fit the human environment, instead of forcing humans to enter theirs, will make using a computer as refreshing as taking a walk in the woods."

Nach der Vorstellung Weisers werden uns ubiquitäre Computer verschiedener Größe umgeben, die keine universellen Computer mehr sind, sondern jeder für einen speziellen Zweck zuständig sein werden. Er beschreibt drei mögliche Formen dieser Computer: Tabs, Pads und Boards. Tabs (dt. Etiketten, Schilder) sind die kleinsten Komponenten. Sie werden als persönliche Geräte von Personen getragen und dienen zur drahtlosen Identifikation des Besitzers und zur Erledigung vieler persönlicher Aufgaben durch Bereitstellung von Informationen und Diensten. Pads (dt. Blöcke) sind wie digitales Papier, das nach Bedarf genommen und überall benutzt werden kann. Boards (auch: yard-size displays; dt. Tafeln) können wie Videoschirme, Pinnwände oder Flip Charts zu Hause oder im Büro genutzt werden. Sie können zur gemeinsamen Arbeit mit anderen verwendet werden. Weiser sieht diese Realisierungen als Anfang des Ubiquitous Computing, dessen Bedeutung in der Interaktion der Geräte liegt, nicht in den Gegenständen an sich.

In einem recht aktuellen Überblick über wichtige Entwicklungen auf dem Gebiet des Ubiquitous Computing sehen Gregory Abowd und Elizabeth Mynatt einige Visionen von Weiser schon in erste konkrete, verbreitete Produkte umgesetzt (siehe [AbowdMynatt 00]): PDAs als persönliche Geräte, Laptops mit größerer Leistungsfähigkeit für anspruchsvollere Aufgaben und digitale elektronische Wandtafeln für gemeinschaftliches Arbeiten. Sie zeigen drei wichtige Forschungsrichtungen auf:

- **Natürliche Schnittstellen** für eine einfachere und direktere Mensch-Maschine-Interaktion. Althergebrachte Eingabeformen wie Tastaturen und Mäuse sollen durch für den Menschen natürlichere wie Handschrift, Sprache und Gesten abgelöst werden. Hier existiert noch großer Verbesserungsbedarf und es darf bezweifelt werden, daß jemals eine optimale technische Umsetzung erreicht werden kann.

- Durch **kontextsensitive Anwendungen** wird eine bessere Anpassung an die Umgebung erreicht. Erste Schritte in diese Richtung sind schon mit der Auswertung von einfachen Umgebungsinformationen wie Standort und Identität gemacht worden. Aber vor der Integration komplexerer Sensorinformationen und der Kombination von Daten aus mehreren Umgebungsquellen stehen noch einige Probleme.

- Die **automatische Aufnahme** von Erlebnissen und ein Zugriff auf diese ist ebenfalls Gegenstand intensiver Forschungen. Dadurch soll das Gedächtnis der Menschen entlastet werden, so daß sich diese auf andere Dinge, wie z. B. die Zusammenfassung und Interpretation der Daten, konzentrieren können. Die Speicherung von Daten und der Zugriff auf diese ist heutzutage noch problematisch, doch gibt es erste vielversprechende Ergebnisse auf dem Gebiet.

2.1.2 Everyday Computing

Abowd und Mynatt sehen auch einen interessanten neuen Forschungsbereich entstehen: Die Gegenwart von Computern wird nicht nur allgegenwärtig sein, sondern auch kontinuierlich. Sie bezeichnen dies als Everyday Computing (siehe [AbowdMynatt 00, S. 42ff]). Dabei stehen insbesondere folgende Besonderheiten des täglichen Umgangs mit Computern im Mittelpunkt:

- Aktivitäten haben keinen klaren Beginn und kein definiertes Ende.
- Eine Unterbrechung und folgende Wiederaufnahme ist jederzeit möglich.
- Mehrere Aktivitäten passieren zeitgleich; ein Wechsel zwischen ihnen ist möglich.
- Zeitinformationen, wie z. B. die Dauer zwischen zwei Ereignissen, spielen eine wichtige Rolle.
- Informationen können nicht in strikte Hierarchien geordnet werden, sondern müssen aus einer Vielzahl von Perspektiven assoziativ zugreifbar sein.

Das eigentliche Ziel ist ihrer Ansicht nach die Verknüpfung von vielen einzelnen Aktivitäten zu einer umfassenden und fortlaufenden Interaktion zwischen Menschen und computerunterstützten Diensten.

2.1.3 Pervasive Computing

Der Begriff des Pervasive Computing (pervasive = dt. durchdringend, überall vorhanden) wird of synonym zu Ubiquitous Computing gebraucht. Er wurde von der Industrie später als dieser geprägt und wird überwiegend im wirtschaftlichen Bereich verwendet (siehe [Mattern 01]). Die allgegenwärtigen Computer sollen eher kurzfristig für elektronischen Handel (E-Commerce) und zum Zugriff auf webbasierte Geschäftsprozesse dienen.

Der kommerzielle Handel mit Gütern und Dienstleistungen über drahtlose mobile Geräte wird mit dem Begriff **M-Commerce** bezeichnet. Beispiele dafür sind Mobile Banking für den Zugriff auf Konten, Mobile Shopping für das Einkaufen in elektronischen Kaufhäusern und Mobile Payment für bargeldloses Bezahlen mit dem Mobiltelefon und anderen mobilen Geräten.

Dabei lassen sich wie beim E-Commerce zwei Fälle unterscheiden:

- **B2C (Business to Consumer)** für eine Firmen–Endkunden Beziehung und
- **B2B (Business to Business)** für den Handel zwischen Firmen, z. B. im Beschaffungswesen.

Beim M-Commerce spielen neben den Möglichkeiten miteinander vernetzter, eingebetteter Rechner insbesondere auch Fragen der Sicherheit, des Datenschutzes, der Entdeckung zur Verfügung stehender Dienste und der Abrechnung eine Rolle.

Auf der technischen Seite geht es um die Einbindung mobiler Geräte in große Anwendungssysteme, den mobilen Zugriff auf angebotene Dienste, z. B. über Web Services, den Aufruf entfernter Programme und Techniken zur drahtlosen Kommunikation.

2.1.4 Nomadic Computing

Leonard Kleinrock hat 1995 mit dem Aufkommen des Nomadic Computing (auch: Nomadicity) einen Paradigmenwechsel im Gegensatz zum herkömmlichen Computereinsatz angekündigt. Dieses trägt viele Züge des Ubiquitous Computing, betont aber den Menschen als soziales Wesen, der von überall, auch beim nomadischen Umherstreifen außerhalb des festen Büros, Zugriff auf Daten und Dienste haben möchte (siehe [Kleinrock 95]).

Dabei ist eine automatische, im Hintergrund ablaufende Anpassung an geänderte Bedingungen nötig. Die Benutzung soll unabhängig sein vom Aufenthaltsort, vom verwendeten Gerät und der Betriebsplattform, der Art des Netzzugriffs und der Bandbreite, und davon, ob der Benutzer sich an einem Ort aufhält oder sich in Bewegung befindet.

Dazu sind sowohl neue Systeme mit passender Architektur und Protokollen als auch drahtlose Netzwerke, die mit geringer Bandbreite und unterbrochenen Verbindungen umgehen können, nötig. Nach Kleinrock müssen die Systeme von Anfang an auf ein Nomadic Computing ausgelegt sein, da ein Anpassen vorhandener Systeme nicht möglich ist. Wichtig ist insbesondere die Interoperabilität heterogener Systeme.

Kleinrock sieht das Nomadic Computing mit heutigen Notebooks, PDAs und Mobiltelefonen schon im Ansatz möglich, doch betont er, daß noch viele wichtige Probleme unter Einbeziehung mehrerer wissenschaftlicher Disziplinen zu lösen sind, um seine Vision zu verwirklichen.

2.2 Klassen mobiler Geräte

Eine Darstellung verschiedener Klassen mobiler Geräte ist nicht einfach, da viele unterschiedliche Arten mobiler Geräte existieren und oft verschiedenartige Funktionalität in einem Gerät integriert ist.

Hier wird die Kategorisierung von Jörg Roth aus [Roth 02a] verwendet. Er unterscheidet mobile Standardcomputer, Bordcomputer, Handhelds, Wearables und Chipkarten. Darüber hinaus unterteilt er diese Kategorien noch in **Universalgeräte**, die vom Hersteller nicht für einen bestimmten Zweck vorgesehen worden sind und beliebige Anwendungen ausführen können, und **Spezialgeräte**, die für einen Einsatzzweck optimiert wurden und auf denen sich keine anderen Anwendungen installieren lassen. In Tabelle 2.1 ist die so entstandene Matrix mit einigen typischen und verbreiteten Geräten dargestellt. Im folgenden werden die Kategorien im einzelnen erläutert.

Kategorie	Universalgerät	Spezialgerät
Mobile Standard-computer	Notebook	Spezielle mobile Computer, z. B. in der Vermessungstechnik und in der Kartographie
Bordcomputer	–	Bordcomputer in Fahr- und Flugzeugen, Satellitensteuerung
Handhelds	PDA	Elektronischer Kalender (nicht programmierbar), Lesestift, E-Book, Web-Pad, GPS-Empfänger, Mobiltelefon, Pager, Digitalkamera
	Smartphone, Communicator, Mobile Spielkonsole, Programmierbarer Taschenrechner	
Wearables	Programmierbares Wearable	Armbanduhr, Pulsmesser
Chipkarten	Smart Card	SIM-Karte, Geldkarte, Telefonkarte

Tabelle 2.1: Klassifikation mobiler Geräte (nach Jörg Roth, [Roth 02a, S. 339])

2.2.1 Mobile Standardcomputer

Diese Klasse umfaßt Geräte, die so kompakt sind, daß sie unterwegs mitgeführt werden können, dabei aber fast so leistungsfähig wie Standardcomputer sind. Universalgeräte dieser Klasse werden als **Notebooks** (auch: Laptops) oder **Subnotebooks** (kleiner und leichter als Notebooks) bezeichnet. Sie verfügen in der Regel über Desktop-Betriebssysteme und können dieselben Anwendungen ausführen.

Zugeständnisse sind an Bildschirme, Tastaturen und Zeigegeräte zu machen. Diese sind in die Geräte integriert und kleiner als die herkömmlichen. Statt Mäusen findet man Trackpoints oder Touchpads. Große Peripheriegeräte wie Laufwerke können aus Platzgründen oft nur extern angeschlossen werden. Kleine Peripheriegeräte wie Netzwerkkarten und Modems sind entweder schon integriert oder können über Einsteckkarten (Standard ist momentan das PC-Card Format) nachgerüstet werden.

Die Stromversorgung ist noch ein limitierender Faktor im mobilen Einsatz. Auch die Verwendung stromsparender Komponenten, wie spezieller Mobilausführungen von Prozessoren, die Abschaltung nicht benutzter Komponenten und die Integration eines Energiemanagements ermöglichen momentan noch keine längeren Laufzeiten als wenige Stunden.

Eine recht neue Gerätekategorie sind **Tablet PCs**. Dieses sind Notebooks, welche zusätzlich zur Tastatur oder statt Tastatur über einen Touchscreen verfügen, der mit einem speziellen Stift (einem elektromagnetischen Digitizer) bedient werden kann. Daran angepaßte Betriebssysteme ermöglichen die Eingabe von Handschrift und die Zeichenerkennung und Umwandlung in digitalen Text, der dann weniger Speicherplatz

benötigt, nach Stichwörtern durchsucht und beliebig weiterverarbeitet werden kann.

2.2.2 Bordcomputer

Bordcomputer sind Spezialgeräte, die fest in Fahrzeuge, Schiffe, Flugzeuge und andere bewegliche Transportmittel integriert sind. Sie sind auf Steuerungs- und Regelaufgaben zugeschnitten und sind nicht frei programmierbar.

Einen großen Anwendungsbereich finden Bordcomputer in Autos. Die meisten modernen Fahrzeuge verfügen über eine Vielzahl an digitalen Systemen, die gesteuert werden müssen, z. B. Airbags, Antiblockiersysteme und Motorsteuerungen. Eine sich immer weiter verbreitende Anwendung sind auch Navigationssysteme, welche über Satelliten und Fahrzeugsensoren die Position bestimmen und den Fahrzeugführer mit akustischen und visuellen Hinweisen zu seinem Ziel leiten können. Dabei fließen bei neueren Systemen aktuelle Verkehrsinformationen in die Routenberechnung mit ein.

In dieser Geräteklasse spielt der Schutz vor Ausfällen und Fehlern eine große Rolle, da Fehlfunktionen aufgrund der direkten Kopplung an sicherheitsrelevante Systeme schwerwiegende Folgen haben können.

2.2.3 Handhelds

Geräte dieser Klasse benötigen keine Unterlage oder Einbau. Sie können in der Hand gehalten und mit ein oder zwei Händen bedient werden. Diese Kategorie mobiler Geräte umfaßt viele verschiedene Varianten.

Personal Digital Assistants (PDAs) sind Universalgeräte. Ursprünglich waren sie für die Verwaltung persönlicher Daten bestimmt. Dazugehörige Anwendungen wie Adreßbücher, Kalender und Aufgabenlisten faßt man unter dem Begriff Personal Information Management (PIM) zusammen. Da auf PDAs beliebige Anwendungen installierbar sind, haben sie sich auch andere Bereiche erschlossen; beispielsweise gibt es E-Mail-Programme und Web-Browser, elektronische Bücher, mathematische Anwendungen, digitale Stadtpläne, Multimedia-Anwendungen, Spiele u. v. m. Über eine Synchronisation mit Desktop-Rechnern können Daten mit diesen ausgetauscht werden (siehe Abschnitt 5.3.2). Mit Hilfe von Mobiltelefonen, mit denen über Kurzstreckenfunk oder Infrarot kommuniziert wird, oder über integrierte Modems kann eine Netzverbindung für einen online-Zugriff aufgebaut werden.

PDAs werden auch als Organizer oder Taschencomputer (Pocket PCs) bezeichnet, wobei Organizer weniger leistungsstark sind und schlanke, ressourcenschonende Betriebssysteme besitzen, während Taschencomputer stärkere Prozessoren und mehr Speicher haben und in der Regel einen größeren Formfaktor aufweisen. Momentan sind zwei Betriebssysteme vorherrschend: Palm OS (für den Organizer-Bereich) und Microsoft Windows CE bzw. Pocket PC (für Taschencomputer), siehe Abschnitt 3.2.3.

Eine weitere Einteilung ist die in tastatur- oder stiftbasierte PDAs. Tastatur-PDAs werden entweder für den Betrieb aufgeklappt und geben dann eine kleine Tastatur frei oder besitzen (in der Regel noch kleinere) Gummitasten. Mit stiftbasierten PDAs ist

eine Eingabe von vereinfachten Buchstaben möglich. Beide Varianten verfügen in der Regel über Touchscreens.

Im Vergleich mit mobilen Standardcomputern besitzen PDAs deutlich reduzierte Prozessorleistungen, Speicherkapazitäten und Bildschirme. Sie sind ohne lange Startphase einsatzbereit und für kurze, aber häufige Arbeiten gedacht. Dementsprechend ermöglichen ihre Akkus eine längere Betriebsdauer von mehreren Wochen bei mäßiger Nutzung.

Mobiltelefone waren als Spezialgeräte gedacht. Mittlerweile hat sich die Technik so weiterentwickelt, daß bei vielen Geräten beliebige Anwendungen installiert werden können. Diese Geräte bezeichnet man als **intelligente Mobiltelefone**. Mit ihnen können u. a. PIM-Anwendungen zur Verwaltung persönlicher Daten genutzt werden. Sie verfügen über Web-Browser im WAP- oder anderen geeigneten Formaten, ermöglichen den Versand von Text- und Multimediakurznachrichten und bieten Programmierschnittstellen zur Ausführung neuer Anwendungen.

Geräte, welche die Technik von PDAs und Mobiltelefonen in sich vereinen, werden als **Smartphones** bezeichnet. Während der Absatz reiner PDAs momentan rückläufig ist, können Smartphones zulegen.

Interessant sind ebenfalls **Tablett-PCs** (auch: SimPads). Im Gegensatz zu den oben als mobile Standardcomputer vorgestellten Tablet PCs verfügen sie nicht über leistungsfähige Komponenten und Standard-Betriebssysteme sondern über PDA-typische Hardware. Allerdings haben sie gegenüber diesen einen deutlich größeren Bildschirm mit besserer Auflösung. Als Betriebssystem kommt momentan häufig Windows CE zum Einsatz.

Desweiteren gibt es eine große Vielfalt von Spezialgeräten. Hierzu zählen beispielsweise

- **Zwei-Wege-Pager** zum Empfang und Versand von kurzen Textnachrichten,
- **E-Books** zum Lesen elektronischer Bücher in einem handlichen Format, geschützt durch Digital Rights Management (DRM)-Techniken,
- **Digitalkameras** für digitale Fotos und
- **GPS-Empfänger** zur Positionsbestimmung und Navigation.

2.2.4 Wearables

Diese Geräte werden am Körper getragen, entweder durch Integration in die Kleidung oder durch Befestigung direkt am Körper. Dabei gibt es viele Möglichkeiten: Diese reichen von auf dem Kopf getragenen Helmen mit integrierten Sichtbrillen über am Gürtel befestigte Geräte bis zu Kleinstcomputern, welche in Knöpfe oder Ringe eingebettet werden. Alle bieten den Vorteil, daß die Hände freibleiben.

Das Gebiet ist im Moment Gegenstand intensiver Forschungen.

2.2.5 Chipkarten

Chipkarten verfügen über einen nichtflüchtigen Speicher und einen Prozessor, benötigen aber externe Lesegeräte für ihren Einsatz. Hauptsächlich werden sie zur Benutzeridentifikation eingesetzt, beispielsweise zur Zugriffskontrolle in gesicherten Bereichen oder für sichere Überweisungen im Bankenbereich. Bekannte Typen von Chipkarten sind SIM-Karten für GSM-Mobiltelefone und Telefonkarten.

Interessant sind **Smart Cards**, da sie programmiert werden können. Dazu stehen eine Reihe von bewährten Techniken zur Verfügung.

2.2.6 Bewertung

Der Fokus dieser Arbeit liegt auf beschränkten mobilen Geräten. Darunter fallen in erster Linie die Universalgeräte, welche oben in der Kategorie Handhelds vorgestellt wurden. Aufgrund ihrer weiten Verbreitung und universellen Einsetzbarkeit werden vor allem PDAs, intelligente Mobiltelefone, Smartphones und Tablett-PCs betrachtet. Mobile Standardcomputer stehen weniger im Mittelpunkt, da sie für Desktopsysteme geschriebene Anwendungen ausführen können. Die fehlende permanente Netzverbindung macht diese Klasse trotzdem interessant.

Die besonderen Eigenschaften von Bordcomputern, Wearables und Chipkarten werden im folgenden nicht betrachtet.

In Tabelle 2.2 sind zusammenfassend Vor- und Nachteile ausgewählter Typen mobiler Endgeräte aus Benutzersicht aufgeführt.

Typ	Vorteile	Nachteile
Mobiltelefone	- weit verbreitet - preisgünstig - Kommunikationsmöglichkeiten integriert	- umständliche Bedienung - kleiner Bildschirm - wenig leistungsfähig
PDAs	- handlich und trotzdem leistungsfähig - einfache Bedienung durch Touchscreen - viele Anwendungen erhältlich	- teurer als Mobiltelefone - recht kleiner Bildschirm
Tablett-PCs	- großer Bildschirm - einfache Bedienung durch Touchscreen - mehr Platz für Erweiterungen	- recht schwer, unhandlich - hoher Preis
Notebooks	- leistungsfähig wie stationäre Rechner - Anwendungen ohne Anpassung ausführbar - viele Hardware-Erweiterungen erhältlich	- hohes Gewicht - hoher Preis - geringe Akkulaufzeit

Tabelle 2.2: Vor- und Nachteile verschiedener Typen mobiler Endgeräte

2.3 Programmierplattformen für mobile Geräte

Für Entwickler von mobilen Anwendungen sind neben den Gerätecharakteristika insbesondere die Möglichkeiten der Geräte als Plattformen für Anwendungen entscheidend. Diese bestimmen weitgehend, welcher Art und wie mächtig die zu entwickelnden Anwendungen sind.

2.3.1 Eigenständige Anwendungen

Anwendungen werden direkt auf dem Gerät ausgeführt. So lassen sich Berechnungen durchführen und vielfältige Interaktionen realisieren. Entwickler können weitgehend die Funktionalität und Handhabung ihrer Produkte bestimmen. Dadurch, daß Daten lokal gehalten werden können, sind sie schnell im Zugriff und die Netzwerkbelastung wird minimiert.

Auf Desktop-Systemen stellt diese Klasse aufgrund der geschilderten Vorteile die überwiegende Mehrzahl aller Anwendungen dar. Insbesondere auf leistungsschwächeren mobilen Geräten wie Mobiltelefonen war es lange Zeit nicht möglich, eigene Anwendungen zu installieren. Dort war man auf browserbasierte Anwendungen angewiesen. Mit dem Fortschreiten der Technik werden jedoch immer mehr dieser Geräte mit entsprechenden Möglichkeiten versehen.

Viele Hersteller mobiler Geräte und Betriebssysteme stellen **Software Development Kits** (SDKs) zur Entwicklung eigenständiger Anwendungen zur Verfügung. Üblicherweise sind diese in weitverbreiteten Sprachen wie C und C++ verfügbar. Sie haben den Vorteil, daß in der Regel alle wichtigen Gerätefunktionen angesprochen werden können, aber den Nachteil, daß der Programmcode nur für eine eingeschränkte Klasse von Geräten benutzt werden kann. Dort setzen **plattformunabhängige Programmiersprachen** an.

Für kleine mobile Geräte haben sich hauptsächlich zwei dieser Sprachen durchgesetzt:

J2ME: Java hat sich in anderen Umgebungen schon als weitgehend plattformunabhängige Sprache bewährt. Die Java 2 Micro Edition erlaubt die Entwicklung von Anwendungen für kleine mobile Geräte, die nicht die volle Funktionalität der Standardausgabe unterstützen können. In Kapitel 6 wird J2ME ausführlich beschrieben.

BREW: Mit dem Binary Runtime Environment for Wireless von Qualcomm[1] können Anwendungen in C und C++ für intelligente Mobiltelefone und Smartphones entwickelt werden. Darüber hinaus wird durch die Systemarchitektur der gesamte Prozeß der Entdeckung kompatibler Anwendungen, deren Kauf, Download, Verwaltung und Bezahlung abgewickelt.

[1] siehe `http://www.qualcomm.com/brew/`

J2ME und BREW schließen sich nicht gegenseitig aus. Es ist möglich, J2ME-Programme über eine BREW-Infrastruktur auf mobile Geräte zu überspielen und dort auf Java Virtuellen Maschinen unter BREW-Systemen laufen zu lassen.

2.3.2 Browserbasierte Anwendungen

Bei diesem Typ erzeugen entfernte Server Inhalte und Anwendungen und übertragen sie über das Netzwerk auf mobile Geräte, welche die Daten in Browsern anzeigen. Die Interaktionsmöglichkeiten sind beschränkt: Durch direkte Eingabe der Adresse oder durch die Anwahl von Hyperlinks können neue Seiten angezeigt werden und über Formulare können Inhalte eingegeben und an den Server geschickt werden. Auch das Netzwerk wird vergleichsweise stark belastet, da darüber alle Daten verschickt werden.

Für Desktop-Systeme wurden die Möglichkeiten durch Skriptsprachen, Java-Applets u. a. erweitert, so daß auf dem Client mehr Dynamik verwendet werden kann. Auf leistungsschwächeren mobilen Geräten ist dies in der Regel nicht möglich.

Vorteilhaft beim browserbasierten Ansatz ist, daß die Geräte nicht über viel Prozessorleistung verfügen müssen und daß prinzipiell Anwendungen über Browser auf vielen verschiedenen Geräten ohne spezielle Anpassung bedient werden können. Störend sind jedoch Inkompatibilitäten von Browsern.

Auf Desktop-Systemen ist momentan HTML (Hypertext Markup Language) als Auszeichnungssprache vorherrschend. Zunehmend wird es mit Multimediainhalten für große Bandbreiten und leistungsfähige Endgeräte optimiert, so daß viele leistungsschwächere mobile Geräte damit nicht umgehen können. Daher wurden abgespeckte Versionen wie WML (Wireless Markup Language) für WAP definiert, die nicht über alle Funktionen von HTML verfügen, sich dafür aber einfacher verarbeiten und darstellen lassen.

Aufgrund der eingeschränkten Interaktion, der nicht ganz einfachen Bedienung und hoher Verbindungspreise wurden diese Anwendungen von den Konsumenten nicht so angenommen wie erhofft. In [Buchanan et al. 01] werden drei Hauptprobleme der Benutzbarkeit von WAP genannt, wie sie von den Benutzern wahrgenommen werden: Der zu kleine Bildschirm, die unklare Seitenstruktur und Navigation sowie die umständliche Texteingabe (siehe dazu auch Abschnitt 3.3: Eigenschaften der Interaktion mit mobilen Geräten).

2.3.3 Austausch von Textnachrichten

In diesem Fall wird keine Anwendung eigenständig ausgeführt oder im Browser angezeigt. Stattdessen kommunizieren Geräte als Klienten mit entfernten Servern über Textnachrichten. Dabei werden nach einer festgelegten Form Kommandos in Kurznachrichten oder E-Mails eingebettet. Die Server werten sie aus und schicken Antworten auf dem gleichen Wege zurück.

Mit diesem einfachen Verfahren lassen sich durchaus komplexere Aufgaben erledigen. Bekannte Beispiele sind Mailinglistenverwaltung durch Marjordomo über E-Mails

oder Fahrplanauskünfte per SMS. In [StajanoJones 98] beschreiben Frank Stajano und Alan Jones weitere mögliche Anwendungen und passende Infrastrukturen. Unter anderem kann man sich damit für personalisierte Dienste (z. B. einen Börsendienst) anmelden und erhält bei bestimmten Ereignissen (z. B. Preisverfall einer Aktie unter eine bestimmte Schwelle) eine Kurznachricht zugeschickt.

Bei diesem Verfahren liegen die Vorteile darin, daß einfache, kostengünstige Mobiltelefone verwendet werden können und keine hohen Kosten entstehen. Sehr vorteilhaft ist auch, daß Benutzer in der Regel schon die nötigen Endgeräte besitzen, auf ihre Erfahrungen mit Kurznachrichten zurückgreifen können und keine neuen Programme einrichten und verstehen müssen. Negativ muß die eingeschränkte Interaktionsmöglichkeit angemerkt werden. Auch müssen sich Benutzer die verwendbaren Kommandos und Parameter merken.

Das Hauptaugenmerk dieser Arbeit liegt auf eigenständigen Anwendungen, da sich auf mobilen Geräten oft nur mit ihnen die benötigte Funktionalität und Handhabung realisieren lassen, die typisch für Anwendungen nach dem WAM-Ansatz sind.

Kapitel 3

Software-ergonomische Betrachtungen

Nachdem ein Schwerpunkt des vorangegangenen Kapitels auf den technischen Besonderheiten mobiler Geräte lag, wird in diesem Abschnitt betrachtet, was bei der Entwicklung mobiler Anwendungen in bezug auf ihre spätere Benutzbarkeit beachtet werden muß.

Ausgehend von allgemeinen software-ergonomischen Grundlagen werden besondere Anforderungen für mobile Geräte zusammengestellt. Anschließend wird darauf eingegangen, wie der geänderte Umgang mit mobilen Geräten die Anwendungsentwicklung beeinflußt. Es werden Forschungsprojekte vorgestellt, die versuchen, die Ein- und Ausgabemöglichkeiten zu verbessern.

Da der Fokus dieser Arbeit auf der Anwendung des WAM-Ansatzes liegt, beschränkt sie sich darauf, die wichtigsten Entwicklungen zusammenzustellen, ohne im Detail auf sie einzugehen.

3.1 Allgemeine software-ergonomische Grundlagen

Die Software-Ergonomie dient dazu, die Benutzbarkeit (engl. usability) von Softwaresystemen so zu gestalten, daß der Anwender seine Arbeit optimal erledigen kann. Dabei geht es insbesondere um die Gebrauchstauglichkeit interaktiver Software. Diese sollte im Mittelpunkt der Entwicklung stehen, nicht technische Aspekte. Es gibt auch nicht *den* Benutzer. Nach Shneiderman ist die Diversität der Benutzergruppe eine große Herausforderung (siehe [Shneiderman 97]). Physische und kognitive Fähigkeiten sind genauso zu beachten wie unterschiedliche Persönlichkeiten, Kulturen und Sprachen. Insbesondere Benutzer mit Behinderungen und ältere Menschen benötigen besondere Unterstützung.

Die ISO 9241, Teil 10: „Ergonomische Anforderungen für Bürotätigkeiten mit Bildschirmgeräten: Grundsätze der Dialoggestaltung“ ([ISO9241-10 96]) definiert sieben Prinzipien der Software-Ergonomie:

- **Aufgabenangemessenheit**: Der Benutzer soll seine Arbeitsaufgabe effektiv und effizient erledigen können.
- **Selbstbeschreibungsfähigkeit:** Jeder Dialogschritt ist für den Benutzer unmittelbar verständlich oder wird auf Anfrage erklärt.

- **Steuerbarkeit:** Der Ablauf der Interaktion läßt sich vom Benutzer steuern. Die Richtung und Geschwindigkeit ist beeinflußbar, bis das Ziel erreicht ist.

- **Erwartungskonformität:** Dialoge entsprechen den Kenntnissen, der Ausbildung und der Erfahrung der Benutzer sowie gebräuchlichen Konventionen.

- **Fehlerrobustheit:** Der Aufwand zur Korrektur von erkennbar fehlerhaften Benutzereingaben ist gering.

- **Individualisierbarkeit:** Dialoge sind an die individuellen Benutzerwünsche und -fähigkeiten und an ihre Arbeit anpaßbar.

- **Lernförderlichkeit:** Das Erlernen des Systems wird unterstützt und angeleitet.

Nach Oberquelle (vgl. [Oberquelle 99]) liegen Hauptmängel der Gebrauchstauglichkeit aktueller Softwaresysteme darin, daß die Nutzung der Systeme in konkreten Kontexten durch konkrete Benutzer zu wenig antizipiert wird. Eine anwendungsorientierte Globalmodellierung verhindert individuelle Anpassungen. Ein ständiges Anwachsen der Funktionalität führt dazu, daß viel Geld unnötig für Schulungen und Support ausgegeben werden muß.

Menschen als soziale, lernende Individuen stehen zu wenig im Mittelpunkt. Ihre Aufgabenerledigung wird als mechanisches, vorherplanbares Verhalten modelliert. Oberquelle fordert deswegen: „Benutzerorientierte Gestaltung muss ihren eigenständigen und frühen Platz in der Software-Konstruktion erhalten und sich in allen Qualitätssicherungsmaßnahmen widerspiegeln.“ ([Oberquelle 99, S. 20])

3.2 Besonderheiten mobiler Geräte

Die allgemeinen software-ergonomischen Anforderungen sollten auch für Anwendungen auf mobilen Geräten gelten. Die Besonderheiten mobiler Geräte und ihr von den Voraussetzungen der ISO-Norm („Bürotätigkeiten mit Bildschirmgeräten“) abweichender Einsatz läßt das nicht selbstverständlich erscheinen. Dazu werden in diesem Abschnitt einige Prinzipien und Konzepte vorgestellt.

3.2.1 Grenzen der Hardware

Die geringe Prozessorleistung und der limitierte Speicherplatz mobiler Geräte bedürfen einer besonderen Beachtung. Software-ergonomische Verbesserungen müssen immer daraufhin untersucht werden, ob sie nicht unverhältnismäßig viele Berechnungen benötigen und Speicherplatz verbrauchen.

Die begrenzte Ausgabefläche und unkonventionelle Eingabemethoden stellen neue software-ergonomische Herausforderungen dar. Aufgrund ihrer Wichtigkeit werden Ein- und Ausgabemöglichkeiten in Abschnitt 3.3 ausführlicher behandelt.

3.2.2 Mobile Kommunikation

Forman und Zahorjan führten in [FormanZahorjan 94] schon 1994 aus, welche Einschränkungen bei der drahtlosen Netzanbindung mobiler Geräte im Gegensatz zu stationären Systemen zu beachten sind:

Verbindungsunterbrechungen: Mobile Anwendungen, die auf das Netz zugreifen wollen, dürfen nicht erwarten, daß immer eine Verbindung besteht. Mobile Netze sind zum einen anfälliger für Ausfälle und zum anderen wird aus Kostengründen nicht immer eine permanente Netzverbindung aufrechterhalten.

Geringe Bandbreite: Mobile Geräte verfügen über eine um Größenordnungen geringere Bandbreite als stationäre Systeme. Anwendungen sollten diese deshalb möglichst effizient nutzen, u. a. durch Komprimierung der zu übertragenden Daten und durch Zusammenfassung mehrerer kleiner Anfragen zu größeren. Bei längeren, die restliche Anwendung blockierenden Operationen sollte der Benutzer durch Fortschrittsbalken o. ä. informiert werden.

Große Durchsatzschwankungen: Je nach Verfügbarkeit des Netzes kann der Durchsatz stark sinken oder steigen. Der Unterschied zwischen drahtloser und drahtgebundener Kommunikation (z. B. ein PDA über ein Modem im GSM-Netz oder über eine Dockingstation am PC) ist noch viel größer. Die Anwendung muß darauf angemessen reagieren.

Heterogene Netzwerke: Stationäre Rechner sind im Regelfall immer an das gleiche Netz angebunden. Mobile Rechner können sich in verschiedene Netze auf unterschiedliche Wege einwählen (z. B. über GSM, Infrarot, Bluetooth, ...).

Sicherheitsprobleme: Weil das illegale Abhören und Manipulieren von drahtlosen Netzen für Unbefugte leichter ist als das von innerhalb geschlossener Gebäude verlegten Kabeln, sind besondere Schutzmaßnahmen zu treffen.

Diese Faktoren führen meist zu einer Verzögerung in der Kommunikation aufgrund nochmaliger Übertragungen, Verbindungsaussetzern, Abarbeitung von Fehlerprotokollen oder aufwendigen Sicherheitsprotokollen. Dies macht es schwieriger, gut bedienbare Anwendungen zu konstruieren, bei denen sich der Benutzer nicht mit technischen Fragen beschäftigen muß.

3.2.3 Design von Consumer Devices

Eine weitere große Besonderheit besteht in den Zielgruppen. Viele mobile Geräte sind als **Consumer Devices**[1] ausgelegt. [Sun 00, S. 3] macht Unterschiede deutlich: „A person who uses a PC at work is effectively being paid to put up with the system's idiosyncrasies – it comes with the territory. [...] At home it's a different story. [...]

[1] Hierfür gibt es leider keine treffende deutsche Übersetzung. „Geräte für Privatnutzer“ wäre eine direkte Übertragung; meist wird aber auch im Deutschen der englische Begriff verwendet.

Psychologically, the fact that consumers spend their own money to purchase a TV, cell phone, or new game box makes them far less tolerant of products that demand that the user adapt to its mode of operation, rather than the other way around."

Während sich auf stationären Systemen wenige Betriebssysteme mit ihren vorherrschenden Paradigmen und graphischen Oberflächen durchgesetzt haben, findet man bei mobilen Geräten viele sehr unterschiedliche Bedienkonzepte. Daher ist eine einheitliche Bedienung auf einer Plattform und ein konsistentes Look & Feel entscheidend. Ebenfalls wichtig ist die Möglichkeit, das Consumer Device und seine Anwendungen an die eigene Arbeit anzupassen, damit ein wirklich persönliches Gerät entsteht.

Es gibt einige Artikel, die auf Grundlage der in den letzten Abschnitten angeführten Forschungsergebnisse und eigenen empirischen Studien Empfehlungen zur Gestaltung von Anwendungen auf mobilen Geräten aussprechen. Diese sind sowohl genereller Art als auch spezifisch für bestimmte Geräte.

Interessant ist die unterschiedliche Designphilosophie bei PDAs. Dieser Markt wird momentan von Palm OS-basierten Geräten und Geräten mit Microsoft-Betriebssystemen (Windows CE, Pocket PC 2002) dominiert. Beide Firmen geben unterschiedliche Empfehlungen ab:

Im sogenannten „Zen of Palm" (siehe [Palm 01]) werden drei grundlegende Prinzipien von Geräten mit **Palm OS** und Anwendungen dafür benannt:

1. Der Benutzer steht im Mittelpunkt.
2. Die Handhelds können überall und jederzeit benutzt werden.
3. Die Applikationen haben einen sehr speziellen Fokus.

Deutlich wird der Unterschied zum stationären Computer. PCs werden universell eingesetzt. Software-Hersteller erweitern ihre Produkte in jeder neuen Version um mehr Möglichkeiten durch immer neue Funktionen (engl. features). Da der Speicherplatz, die Rechengeschwindigkeit und andere Faktoren keine so große Rolle spielen, lassen sich Produkte mit mehr Funktionen im allgemeinen besser verkaufen, auch wenn viele Benutzer nur die Grundfunktionalitäten nutzen.

Mobile Geräte sollten dagegen schnell und einfach zu benutzen sein. Sie müssen günstig im Batterieverbrauch, klein und leicht sein. Zuviele Funktionen behindern diese Ziele. Daher ist es laut Palm wichtig, die notwendigen Funktionen herauszuarbeiten, diese einfach zu realisieren und den Rest wegzulassen. Aufwendige Berechnungen sollten nicht auf dem mobilen Gerät getätigt werden, sondern – falls sie wirklich benötigt werden – nach Datenabgleich auf dem PC vorgenommen werden.

Die mobile Anwendung sollte die Erledigung der häufigsten Aufgaben mit möglichst wenigen Tastendrücken ermöglichen. Die Oberfläche sollte übersichtlich sein, weshalb seltener oder nur von fortgeschrittenen Benutzern benötigte Funktionen nicht auf dem Hauptfenster der Anwendung sondern über Menüs, Shortcuts o. ä. zur Verfügung gestellt werden sollten.

Demgegenüber steht die Philosophie hinter **Microsofts Betriebssystemen** für PDAs. Zuberec schreibt in [Zuberec 00, S. 104] über die Microsoft-Anwendungen: „Each product was designed to be a desktop companion. To that end, consistency with desktop applications and operating system has played a significant role in defining the user interface for each product.“ Die Zielgruppe sind hier Geschäftsleute, die Microsoft-Systeme aus dem Büroalltag kennen und auch unterwegs auf ihre Daten zugreifen wollen. Für sie wollte man ein Umlernen vermeiden, indem möglichst viele der Funktionen und Umgangsformen der Desktopsysteme für mobile Geräte adaptiert wurden.

Der Designprozeß sah so aus, daß von den bestehenden Systemen ausgegangen wurde. Die Oberflächen der mobilen Anwendungen wurden dabei modifiziert, so daß sie auch auf kleinen Bildschirmen nutzbar waren. Man versuchte, die meisten wichtigen Funktionen zu übernehmen. Die Entwickler merkten jedoch, daß dadurch zur Erledigung häufig vorkommender Aufgaben teilweise sehr viele Schritte nötig waren. Außerdem waren einige Methoden nicht so einfach auf Handhelds übertragbar (z. B. ist die Aktivierung kleiner Symbole per Doppelklick mit einem Stift auf einem kleinen Touchscreen umständlich). Diese wurden deshalb angepaßt. In einem iterativen Prozeß entstanden so die Betriebssysteme und Anwendungen für die Handhelds.

Für **Mobiltelefone** mit ihren noch beschränkteren Möglichkeiten sind ebenfalls einige Besonderheiten bei der Entwicklung von Anwendungen zu beachten. Man findet unter ihren Benutzern viele, die nur wenige oder gar keine Computerkenntnisse besitzen. Daher steht hier der Benutzer noch mehr im Mittelpunkt. Die Anwendungen müssen einfach, intuitiv und zuverlässig sein. Wie in [Sun 00] festgestellt wird, sind z. B. Neustarts während eines Telefonats nicht tolerabel. Vielmehr erwarten Benutzer, daß die Geräte leistungsfähig sind und sich unkompliziert bedienen lassen. Sie sollten erwartungskonform und zuverlässig arbeiten, ohne daß eine lange Einarbeitungszeit erforderlich ist.

Dabei sollte auch immer beachtet werden, welche Auswirkungen die Benutzung mobiler Geräte auf das soziale Verhalten hat. Beispielsweise haben Palen, Salzman und Young in [Palen et al. 00] durch empirische Untersuchungen festgestellt, daß sich das kommunikative Verhalten neuer Benutzer von Mobiltelefonen schnell ändert und dadurch andere Möglichkeiten und Tätigkeiten entstehen.

3.2.4 Usage Spaces

Von Aaron Marcus und Eugene Chen stammt das Konzept der Usage Spaces (auf deutsch vielleicht übersetzbar mit Gebrauchsbereiche), siehe [MarcusChen 02]. Ihr Ziel war es, ein innovatives, wirklich persönliches Gerät zu entwickeln, das der Benutzer stets mit sich führen und für viele Aufgaben verwenden kann. Dabei gingen sie anwenderzentriert vor, indem sie technologisch sehr unterschiedlich vorgebildete Benutzer bei ihren Routinetätigkeiten beobachteten. Aus einer Strukturierung der daraus gewonnenen Erkenntnisse entstanden Usage Spaces als ein analytisches Rahmenwerk für die Möglichkeiten der Unterstützung durch mobile Geräte. Sechs Bereiche sind unterschieden (siehe Abbildung 3.1):

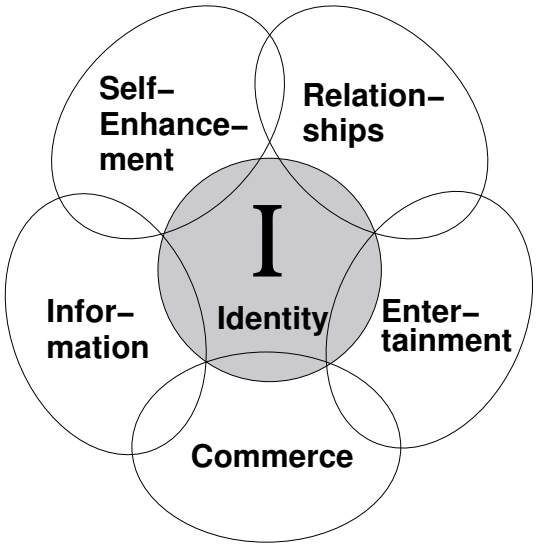


Abbildung 3.1: Usage Spaces nach Aaron Marcus und Eugene Chen, in Anlehnung an: [MarcusChen 02, S. 40]

Identity: Der Benutzer steht im Mittelpunkt. Das große I in der Mitte der Abbildung steht für den Gebrauch des Gerätes als Speicher von Informationen des Benutzers.

Information: Enthält statische Informationen zum Nachschlagen wie Wettervorhersagen, klassische PIM (personal information management) - Funktionen wie Adressen und Termine und geschäftliche Informationen wie beispielsweise spezielle Lexika.

Self-Enhancement: Beinhaltet Möglichkeiten, die eigenen Fähigkeiten zu erweitern. Es gibt z. B. Funktionen zur Ergänzung des Gedächtnisses und zum Lernen.

Relationships: Dient dem Kontakt zu anderen Personen und dem Aufbau und Erhalt sozialer Beziehungen.

Entertainment: Kann Musik abspielen, bietet Computerspiele an und weitere Dinge, die der Entspannung und der Unterhaltung dienen.

M-Commerce: Enthält elektronisches Geld und Dienstleistungen, die von anderen dem Benutzer zur Verfügung gestellt werden.

Es gibt zwei grundsätzliche Philosophien für die Entwicklung solcher mobiler Geräte. Der **Informationsgeräte-Ansatz** (engl. information-appliance) hebt den Wert des Geräts für spezialisierte Funktionen hervor. Dagegen soll die Metapher des **Schweizer Armee-Messers** (engl. Swiss Army knife) die breite Einsetzbarkeit eines Gerätes in vielen verschiedenen Funktionen symbolisieren. Marcus und Chen vertreten die Meinung, daß der richtige Weg irgendwo zwischen diesen Extremen liegt und genau geprüft werden muß, welche Funktionen mobil verfügbar sein müssen.

3.3 Eigenschaften der Interaktion mit mobilen Geräten

Aufgrund der besonderen Hardware und Nutzung mobiler Geräte kommt der Mensch-Maschine-Interaktion eine spezielle Bedeutung zu. Die begrenzten Möglichkeiten der Ein- und Ausgabe spielen hierbei eine entscheidende Rolle.

3.3.1 Eingabemöglichkeiten

Es lassen sich grundsätzlich zwei Aufgaben unterscheiden: Die Eingabe von Texten und Graphiken und die Navigation und Auswahl von Elementen.

Zur Bewertung der **Texteingabe** sind zwei Faktoren wesentlich: Die Geschwindigkeit und die Fehlerhäufigkeit. Erstere wird in Zeichen pro Sekunde (cps) oder Wörter pro Minute (wpm) gemessen.[2] Vergleichswerte sind 16 bis 33 wpm bei Handschrift und 50 (durchschnittlicher Büroarbeiter) bis 150 wpm (Experten) bei Desktoptastaturen mit QWERTY-Layout (nach [Shneiderman 97, S. 307f]).

Bei der Fehlerhäufigkeit sind vier Fehlertypen zu unterscheiden: Eingabe eines falschen Zeichens, Auslassung eines Zeichens, Hinzufügen eines unnötigen Zeichens und Transposition zweier Zeichen. Geschwindigkeit und Fehlerhäufigkeit sind dabei nicht gleichzeitig optimierbar. [MacKenzieSoukoreff 02] stellt fest: „Subjects can enter text more quickly if they are willing to sacrifice accuracy. For subjects to perform with high accuracy, they must slow down."

Die **Eingabe von Graphiken** spielt in den meisten Anwendungen keine große Rolle und wird in der wissenschaftlichen Literatur nur begrenzt behandelt. Deshalb geht diese Arbeit darauf nicht weiter ein.

Die **Navigation** umfaßt das Anwählen von Elementen auf dem Bildschirm (Fenster, Textfelder, Tasten) und die Positionierung einer Marke (Textcursor, Zeiger).

Bei **Desktopsystemen** herrscht die Dateneingabe mittels Tastatur und Maus vor. Dabei können mit der Tastatur Texte eingegeben werden. Die Navigation erfolgt bei Systemen mit graphischer Oberfläche über die Maus und über Steuerungstasten oder Tastenkombinationen der Tastatur. Bei mobilen Geräten stößt man mit diesen Mitteln auf Probleme, da ergonomische Tastaturen genauso wie Mäuse eine gewisse Minimalgröße benötigen. Außerdem müßten Mäuse zusätzlich zum mobilen Gerät mitgeführt werden und ihre Benutzung erfordert eine ebene, nicht zu kleine Unterlage.

Im folgenden werden die gebräuchlichen Eingabevarianten der Geräteklassen Mobiltelefon, PDA und Laptop betrachtet und wichtige Forschungsergebnisse zur Verbesserung der Bedienbarkeit aufgezeigt.

Mobiltelefone

Standard bei Mobiltelefonen ist die ITU-T Tastatur mit zwölf Tasten, den Ziffern 0 bis 9, * und #. Zusätzliche Tasten dienen besonderen Aufgaben, z. B. dem Zugriff auf das

[2] Dabei werden durchschnittlich fünf Zeichen pro Wort angenommen. Zur Umrechnung gilt folglich: [wpm] = [cps] * 60 / 5.

Menü und Navigation in demselben, der Wahl einer angegebenen Nummer oder zum Abbruch eines Anrufes.

Für die **Texteingabe** werden traditionell die Zifferntasten verwendet. Die Tasten 2 bis 8 sind dabei fortlaufend mit drei Buchstaben des Alphabets belegt, die 9 enthält die letzten vier Buchstaben des 26-teiligen Alphabets. Die 0 steht für das Leerzeichen, die 1 für diverse Sonderzeichen. Jede Zifferntaste steht auch für die entsprechende Ziffer und evtl. weitere Zeichen.

Das klassische Eingabeverfahren, als Multitap bezeichnet, erfordert mehrere Tastendrücke auf eine Taste. Beim ersten wird der erste Buchstabe ausgewählt (zum Beispiel a bei der Taste 2), bei weiteren die nächsten (zweimal Taste 2 wählt b, dreimal wählt c usw.). Falls zwei im Text aufeinanderfolgende Buchstaben derselben Taste zugeordnet sind, ist zwischen beiden eine spezielle vorwärts-Taste zu drücken oder eine gewisse Zeit (bei den meisten Geräten etwa 1,5 Sekunden) zu warten.

Eine Alternative hierzu ist die Verwendung eines Lexikons der häufigsten Wörter (siehe [DunlopCrossan 99]). Diese Methode wird englisch als „Dictionary-based Disambiguation“ bezeichnet. Jede Taste, die einem Buchstaben des einzugebenden Wortes entspricht, wird nur einmal gedrückt. Basierend auf der Wahrscheinlichkeit des Vorkommens der mit dieser Kombination zu erzeugenden Wörter ändert sich die Eingabe. Wenn das gewünschte Wort nicht erscheint, können mit einer speziellen Taste auch Wörter geringerer Wahrscheinlichkeit nacheinander gewählt werden. Falls das gewünschte Wort nicht im Lexikon vorhanden ist, muß der Benutzer auf das Multitap-Verfahren zurückgreifen. Eine verbreitete Implementation dieser Technik ist T9 (text on nine keys).

Eine neuere Technik ist die „Prefix-based Disambiguation“ (LetterWise, siehe dazu [MacKenzie et al. 01]). Hierbei wird nicht ein komplettes Wörterlexikon mit Wort-Wahrscheinlichkeiten gespeichert, sondern die Wahrscheinlichkeit von Wortpräfixen bis zu einer bestimmten Länge (z. B. drei). Wie bei der Lexikonmethode existiert eine vorwärts-Taste, die inkorrekte Vermutungen auflöst. Diese wird naturgemäß am häufigsten beim ersten Buchstaben eines Wortes benötigt werden, wenn noch kein Präfix vorliegt.

Ein empirischer Vergleich dieser Eingabemöglichkeiten (aus [MacKenzie et al. 01]) zeigt, daß sich mit Training bei allen Verfahren die Eingabegeschwindigkeit signifikant steigern und die Fehlerrate etwas senken läßt. Multitap ist das langsamste Verfahren. Mit T9 kann die Geschwindigkeit erhöht werden, insbesondere wenn nur gebräuchliche Wörter verwendet werden, die im Lexikon vorhanden sind. LetterWise ist die schnellste Methode, die noch dazu im Gegensatz zu lexikonbasierten Verfahren deutlich weniger Speicherplatz benötigt. Die Spanne der Eingabegeschwindigkeiten für alle Verfahren und alle Trainingsstufen der Benutzer reicht von 6 bis 21 wpm.

Die **Navigation** bei Mobiltelefonen erfolgt durch hierarchisch gegliederte Menüs, Auswahl von Alternativen durch zugeordnete Tasten und durch Tasten für spezielle Funktionen. Für direktere Interaktionen fehlen Zeigegeräte oder ein Touchscreen. Jeder Mobiltelefonhersteller verfolgt ein eigenes Bedienkonzept, das innerhalb einer Modell-

reihe einheitlich gestaltet ist, aber bei fremden Geräten ein Umlernen beim Benutzer erfordert. Dies betrifft beispielsweise den Aufbau der Menüs und spezielle Tasten und ihre Belegung.

PDAs

Die traditionellen **Texteingabemöglichkeiten** der Personal Digital Assistants können in drei Gruppen eingeteilt werden: Hardwaretastaturen, Softwaretastaturen und Handschrifterkennung.

Die eingebauten (Mini-)Tastaturen (engl. auch Thumboards genannt, da sie am besten mit den Daumen bedient werden) weisen nur eine geringe Größe auf. Die Tasten sind klein und stehen eng beieinander. Beim Schreiben ohne Unterlage muß eine Hand das Gerät halten. Deshalb lassen sich keine guten Eingaberaten erreichen. Für die gängigen PDAs gibt es eine recht große Auswahl an externen Tastaturen. Diese lassen wesentlich höhere Geschwindigkeiten zu, erfordern aber eine feste Unterlage.

Softwaretastaturen (auch Virtuelle Tastaturen genannt; engl. Soft Keyboards) bilden eine Tastatur auf der Bildschirmoberfläche der PDAs ab. Die Tasten lassen sich mit einem Stift anwählen. Die Geschwindigkeit liegt hier etwa um den Faktor drei niedriger als bei Hardwaretastaturen. Daher gibt es Forschungsansätze und daraus entstandene Produkte, die die Benutzbarkeit verbessern wollen. In [MacKenzieZang 99] wird ein Tastaturlayout beschrieben, das von der traditionellen QWERTY-Anordnung abweicht (u. a. liegen häufig verwendete Tasten zentral und es gibt vier gut erreichbare Leertasten) und nach einigem Training höhere Geschwindigkeiten erlaubt.

Softwaretastaturen haben im Gegensatz zu ihren materiellen Abbildern auch die Möglichkeit, das Layout dynamisch zu ändern und zusätzliche Informationen einzublenden. In [Masui 98] wird eine Tastatur beschrieben, bei der nach Eingabe eines Buchstabens wahrscheinliche Wörter mit dem bisher eingegebenen Präfix vorgeschlagen werden. Nach vollständiger Eingabe oder Auswahl eines Wortes werden mögliche Folgewörter angezeigt. Insbesondere für ostasiatische Sprachen mit ihren mehreren tausend Zeichen ist dieses Verfahren vorteilhaft. In [MacKenzieSoukoreff 02] werden weitere Tastenanordnungen von Softwaretastaturen vorgestellt und bzgl. der möglichen Eingabegeschwindigkeiten verglichen.

In nahezu alle PDAs ist eine Texterkennung für Handschrift integriert. Aufgrund der begrenzten Rechenleistung ist es für einige Systeme erforderlich, vereinfachte Zeichen einzugeben, um eine zuverlässige Erkennung noch in Echtzeit zu ermöglichen (z. B. Unistrokes von Xerox, Graffiti von Palm). Fortlaufende Handschrift wird nicht immer richtig behandelt. Einige Systeme können vom Benutzer trainiert werden, um die Erkennungsrate zu steigern. Trotzdem ist die Leistung um Größenordnungen geringer als z. B. auf Tablet PCs mit Windows XP Tablet Edition.

Es existieren eine Reihe von Vorschlägen für alternative Texteingaben. Beispielsweise werden im MOHOLT-System (MObile HOchLeistungs-Texteingabe) von Magerkurth ([Magerkurth 02]) sechs Hardwaretasten des Gerätes mit einbezogen. Fünf von ihnen sind Vokalen direkt zugeordnet. Konsonanten werden durch Kombination der Hardwaretasten mit vier virtuellen Tasten auf dem Bildschirm eingegeben. Die Einga-

begeschwindigkeit kommt an die der Zeichenerkennung und der vollständig virtuellen Tastatur nicht heran, doch liegt die Fehlerzahl unter der dieser Methoden.

Die Firma Senseboard entwickelt ein Verfahren, bei dem keine reale Tastatur notwendig ist.[3] Sensoren an beiden Händen nehmen die beim Zehn-Finger-Schreiben entstehenden Handbewegungen auf. Eine kleine Elektronik wandelt diese in das entsprechende Zeichen um und sendet es drahtlos an das mobile Gerät. Die israelische Firma VKB forscht an Laserprojektionen, die eine Tastatur auf dem Tisch abbilden.[4] Getippte Tasten werden per Infrarotsensor erkannt. Beide Verfahren haben den Nachteil, daß der Benutzer keinerlei taktile Rückkopplung erhält.

Dank eines Touchscreens kann die **Navigation** komfortabel erfolgen. Elemente können direkt auf dem Bildschirm angewählt werden. Auch direkte Manipulation[5] ist möglich, soweit es der begrenzte Bildschirm zuläßt.

Es gibt auch Forschungsansätze für ungewöhnlichere Mensch-Maschine Schnittstellen. [Hinckley et al. 00] beschreibt ein Projekt, bei dem PDAs mit Sensoren ausgestattet wurden: Berührungs-, Neigungs- und Entfernungssensoren, die den Kontext einer Aktion feststellen können. Beispielsweise kann automatisch eine Sprachaufnahme gestartet werden, sobald folgende Bedingungen zutreffen: 1) Der Benutzer hält das Gerät. 2) Die Entfernung beträgt weniger als 8 cm. 3) Der Benutzer neigt den PDA zu sich. Die Autoren kommen zu dem Ergebnis, daß diese Techniken in zukünftigen Benutzungsschnittstellen sehr nützlich sein können.

Smartphones stellen auch aufgrund ihrer Eingabemöglichkeiten eine Kombination aus Mobiltelefonen und PDAs dar. Sie verfügen in der Regel über einen Touchscreen und eine ITU-T Mobiltastatur.

Laptops

Laptops besitzen eingeschränkte Tastaturen. Meist fehlt der Ziffernblock und die anderen Tasten sind platzsparender angeordnet. Dies erschwert die Bedienung ebenso wie die manchmal geringere Tastengröße. Mäuse können angeschlossen werden. Zusätzlich ist fast immer ein Trackpoint oder Touchpad vorhanden, das ebenfalls eine Bewegung des Mauszeigers und Betätigen der Tasten ermöglicht, aber mit weniger Platz auskommt.

Eine neuere Entwicklung sind die sogenannten Tablett-PCs. Hier ist zusätzlich eine Bedienung per Touchscreen möglich. Neben der direkteren Positionierung des Zeigers ist auch Texteingabe durch die ins Betriebssystem integrierte, im Gegensatz zu PDAs viel leistungsfähigere OCR-Software möglich.

[3] Siehe `http://www.senseboard.com/`

[4] Siehe `http://www.vkb.co.il/`

[5] Direkte Manipulation ersetzt das Eintippen von Befehlen durch den Umgang mit graphischen Repräsentationen von Objekten und Aktionen mit unmittelbarer Rückkopplung (nach [Shneiderman 97, S. 185ff]).

Mit diesen Möglichkeiten gestaltet sich die Dateneingabe recht komfortabel, so daß im Regelfall keine besonderen Hilfsmittel in der Anwendungssoftware nötig sind.

In [Rosenberg 98] ist ein ausführlicherer Überblick über Eingabegeräte für mobile Geräte zu finden.

3.3.2 Ausgabemöglichkeiten

Die größte Herausforderung der Ausgabe bei kleinen mobilen Geräten wie Mobiltelefonen und PDAs ist die geringe Größe und Auflösung des Bildschirms. Erik Sparre schreibt in [Marcus et al. 98] im Rahmen der CHI 1998: „With small displays, the need for navigation by the user is increased, but it becomes difficult to the designer to provide an overview, to supply a context, and to speak the user's language." Im selben Artikel wird der englische Begriff **Baby Faces** für diese Art von Bedienschnittstelle geprägt. An der kleinen Größe des Bildschirms wird sich voraussichtlich auch zukünftig nichts ändern, da diese Geräte gut in der Hand liegen und in Taschen passen sollen.

Besonderheiten

Aaron Marcus führt Gründe für die Schwierigkeiten beim Design für Baby Faces an (siehe [Marcus 01]):

- Der zur Verfügung stehende Platz für Daten und Funktionen ist sehr eingeschränkt.
- Die Benutzer sind extrem heterogen und verfügen über verschiedene Nutzungskontexte.
- Viele Benutzer sind Neulinge in bezug auf Inhalte und Technologien der Geräte.
- Die internationale Ausrichtung erfordert die Berücksichtigung lokaler kultureller Hintergründe und Sprachlokalisierungen.
- Die Benutzer erwarten viele verschiedene Eigenschaften und Funktionen in portablen, preisgünstigen Geräten.
- Besondere rechtliche Regelungen müssen berücksichtigt werden, beispielsweise das Verbot des Telefonierens im Auto ohne Freisprecheinrichtung.

Diese Punkte betreffen hauptsächlich Consumer Devices. Für geschäftliche Anwendungen, die auf einen bestimmten Bereich zugeschnitten sind und von Anwendungsexperten bedient werden, sieht einiges anders aus.

Widgets

Von Herstellern mobiler Geräte und den Entwicklern ihrer Betriebssysteme gibt es detaillierte Angaben, wie die Bedienoberflächen aussehen sollten, damit sie insgesamt ein konsistentes Bild abgeben. Dies ist wichtig, damit der Benutzer den grundlegenden Bildaufbau und die Eingabemöglichkeiten nicht bei jeder Anwendung von Grund auf neu erlernen muß. Im Gegensatz zu Desktopsystemen wird dieses Wissen vorausgesetzt, da Hilfstexte zu den einzelnen Funktionen (wie z. B. Tooltips und Statuszeile) durch den begrenzten Platz nicht gleichzeitig mit dem sonstigen Inhalt angezeigt werden können und dem Benutzer sein momentaner Kontext klar sein muß. Außerdem erfolgt die Benutzung typisch für Consumer Devices u. U. nicht mit voller Aufmerksamkeit.

Bekanntere Beispiele dieser Style Guides sind die Palm OS User Interface Guidelines[6], die Style Guides für WAP-Seiten von Openwave[7] oder der Style Guide von Sun für J2ME/MIDP[8].

Es gibt eine Reihe von Forschungsprojekten, die an der Verbesserung der Benutzbarkeit an dieser Stelle arbeiten. Sie versuchen, die vorhandenen Widgets besser zu gestalten oder zu positionieren oder suchen nach neuen Rückkopplungsmechanismen. Ein Beispiel für ersteren Ansatz ist die Verwendung von halbtransparenten Widgets, die hinter anderem Inhalt wie Text oder Bildern angezeigt werden und so Platz sparen (siehe [Kamba et al. 96]). Wenn der Benutzer Elemente benutzen möchte, entscheidet die Zeitdauer der Interaktion (z. B. kurzer oder langer Druck mit dem Stift auf den Bildschirm), welche Ebene angesprochen wird. Eine Möglichkeit, die vorhandenen Interaktionsformen zu erweitern, ist die Verwendung von Tönen (siehe [BrewsterCryer 99]). Diese werden abgespielt, wenn der Benutzer eine Taste auf dem Bildschirm mit dem Stift herunterdrückt, losläßt oder abrutscht. Brewster und Cryer haben empirisch untersucht, daß dadurch die Größe von Tasten auf dem Bildschirm reduziert werden kann, so daß mehr Elemente dargestellt werden können.

Aufbereitung von Texten und Bildern

Wichtig für eine Darstellung auf kleinen Bildschirmen ist die Aufbereitung von Texten und Bildern.

Die Schriftart und -größe von **Text** sind so zu wählen, daß er gut lesbar ist. Längerer Text muß geeignet auf mehrere Bildschirmseiten umgebrochen und durch Navigationsmöglichkeiten verbunden werden.

In [Gomes et al. 01] wurde untersucht, inwieweit vorhandener Text inhaltlich komprimiert werden kann, um noch verständlich zu sein. Die dabei eingesetzten Techniken umfassen das Weglassen für das Verständnis unwichtiger Wörter wie Pronomen und

[6] PalmSource Developer Documentation, Document Number 3101-001-HW, `http://www.palmos.com/dev/support/docs/ui/UIGuide_Front.html`

[7] Openwave Graphical Browser Application Style Guide, `http://demo.openwave.com/pdf/51/style_guide.pdf`

[8] Sun MIDP 1.0 Style Guide 1.0a, `http://java.sun.com/j2me/docs`

Adjektiven, die Verwendung von gebräuchlichen Abkürzungen und das Unterdrücken von Vokalen. Das Fazit ist, daß im Portugiesischen 40% der Zeichen ohne große Beeinträchtigung des Verständnisses weggelassen werden können.

Bei neu zu erstellendem Text für mobile Anwendungen ist auf eine knappe und eindeutige Sprache zu achten.

Bei **Bildern** bestehen neben der Verkleinerung oft auch noch die Notwendigkeiten, die Anzahl ihrer Farben zu reduzieren, z. B. für die Darstellung auf schwarz/weiß-Bildschirmen oder um ihren Speicherplatzbedarf zu verringern, und eine Transformation zur Anpassung an die Höhe und Breite des Ausgabegerätes.

Rist und Brandmeier unterscheiden in [RistBrandmeier 02] drei mögliche Umformungen:

Uniforme Transformationen: Hier werden nur wenige Bildinformationen (wie z. B. Größe und Farbtiefe) verwendet und die gleiche Transformation auf alle Bilder angewendet. Diese Methode hat den Vorteil, daß sie leicht anwendbar und schnell ist. Sie erzeugt aber im Regelfall keine besonders brauchbaren Bilder.

Informierte Transformationen: Anhand einer syntaktischen oder semantischen Klassifikation wird ein Quellbild eingeordnet und dann mit passenden Methoden transformiert. Beispielsweise können Personenbilder anders als Landschaftsbilder behandelt werden oder Innen- und Außenaufnahmen unterschieden werden.

Erzeugung neuer Bilder: Ein neues Bild wird auf Basis von aus dem Quellbild extrahierten semantischen Informationen generiert. Ein Beispiel ist die Gewinnung von Straßendaten aus einer Umgebungskarte und ihre Umsetzung in Bilder von Richtungspfeilen zur mobilen Navigation.

Anzustreben ist eine Automatisierung dieser Umformungen, so daß eine für die mobilen Geräte adäquate Ausgabe schnell erzeugt werden kann.

3.4 Zusammenfassung und Auswertung

In diesem Kapitel wurden software-ergonomische Grundlagen erläutert. Es wurde zusammengefaßt, wie die Besonderheiten mobiler Geräte zu berücksichtigen sind, insbesondere in ihrer Rolle als Consumer Devices. Das Usage Spaces-Modell von Marcus und Chen, das die Unterstützungsmöglichkeiten durch persönliche mobile Geräte darstellt, wurde vorgestellt. In den Abschnitten zur Interaktion mit mobilen Geräten wurden Ein- und Ausgabemöglichkeiten ausführlich diskutiert.

Es hat sich gezeigt, daß aufgrund der Einschränkungen mobiler Geräte Fragen der Benutzbarkeit sehr wichtig sind. Die Ein- und Ausgabemöglichkeiten mobiler Geräte sollten mit Bedacht genutzt werden.

Von den allgemeinen software-ergonomischen Anforderungen spielen auf mobilen Geräten insbesondere die Individualisierbarkeit, Erwartungskonformität und Fehlerrobustheit eine Rolle. Mobile Anwendungen müssen sich intuitiv bedienen und an die

Fähigkeiten und Vorlieben ihrer Benutzer anpassen lassen. Fehler sind noch weniger als auf stationären Systemen tolerierbar. Auch die anderen software-ergonomischen Anforderungen sind für Anwendungen auf mobilen Geräten wichtig.

In den folgenden Abschnitten wird immer wieder auf die hier zusammengestellten Erkenntnisse zurückgegriffen, um der Wichtigkeit ergonomischer Fragen Rechnung zu tragen.

Kapitel 4

WAM-Ansatz bei mobilen Geräten

4.1 Kurze Vorstellung des WAM-Ansatzes

An dieser Stelle soll ein erster Überblick über den WAM-Ansatz gegeben werden. Genauere Erläuterungen zu den einzelnen Bestandteilen erfolgen in der weiteren Arbeit dort, wo sie benötigt werden.

Der WAM-Ansatz (Werkzeug & Material-Ansatz) ist eine Softwareentwurfs- und Konstruktionstechnik (siehe [Züllighoven 98]). Ausgehend vom Ziel einer anwendungsorientierten Software mit hoher Gebrauchsqualität umfaßt er verschiedene Elemente wie ein Leitmotiv mit Entwurfsmetaphern, eine Anforderungsermittlung mit spezifischen Dokumenttypen und ein evolutionäres, kundenorientiertes Vorgehen. Diese werden im nächsten Abschnitt kurz beschrieben.

JWAM ist eine Realisierung des WAM-Ansatzes in der Programmiersprache Java. Es wird im darauffolgenden Abschnitt vorgestellt.

4.1.1 Grundlagen

Zentrale Idee des WAM-Ansatzes ist die Herstellung einer **Strukturähnlichkeit** zwischen Gegenständen des realen Anwendungsbereiches und ihrer Realisierung im Softwaresystem. Die Entwickler besitzen dadurch eine Entscheidungsgrundlage für den Systementwurf und Anwender können sich besser im System orientieren.

Der WAM-Ansatz definiert **Leitbilder**, die Entwicklern und Anwendern helfen sollen, das Softwaresystem zu entwickeln und zu verstehen. Ein Leitbild ist dabei eine grundsätzliche Sichtweise. Ein häufig verwendetes Leitbild ist der Arbeitsplatz für qualifizierte und eigenverantwortliche Expertentätigkeit. Ein Leitbild wird durch **Entwurfsmetaphern** konkretisiert, die Elemente und Konzepte der Anwendung durch bildhafte Vorstellungen von realen Gegenständen beschreiben.

Grundlegende Metaphern des WAM-Ansatzes sind Werkzeug, Material, Automat, Arbeitsumgebung und Fachlicher Service. **Werkzeuge** arbeiten interaktiv auf Materialien. Sie können vom Benutzer flexibel gebraucht und wieder abgelegt werden und geben keine festen Abläufe vor. **Materialien** werden zum Arbeitsergebnis. Sie verhalten sich passiv und können nicht direkt, d. h. ohne Werkzeuge, bearbeitet werden. Materialien besitzen an ihrer Schnittstelle fachliche Operationen, über die ihr Zustand verändert werden kann. **Automaten** erledigen Routineaufgaben für den Anwender. Sie können über Einstellungen konfiguriert werden und arbeiten dann selbständig.

Gegenstände wie diese können in der **Arbeitsumgebung** abgelegt und angeordnet werden. Man unterscheidet persönliche Arbeitsplätze der einzelnen Benutzer und für alle zugängliche Räume der Arbeitsumgebung, über die eine Kooperation stattfinden kann. **Fachliche Services** stellen als Dienstleister Funktionalitäten zur Verfügung. Sie können Materialien verwalten und zur Bearbeitung herausgeben oder fachliche Funktionen zur Arbeit auf von ihnen gehaltenen Materialien anbieten. Im Gegensatz zu Werkzeugen verfügen sie über keine eigene Präsentation und geben auch keine Handhabung vor.

Zur Hilfe bei der Analyse des Systems sieht der WAM-Ansatz anwendungsorientierte Dokumenttypen vor. In **Szenarios** werden die realen Arbeitsabläufe, die zukünftig vom System unterstützt werden sollen, niedergelegt. Dabei auftretende Fachbegriffe werden ebenso wie durch das entwickelte System geprägte in einem **Glossar** festgehalten und erläutert. **Systemvisionen** beschreiben, wie die zukünftige Arbeitserledigung mit dem System erfolgen kann. Sie dienen für die Entwicklung als Leitlinie.

Der WAM-Ansatz fördert eine **evolutionäre Systementwicklung**, bei der ein Produkt in fortwährenden Verfeinerungen entsteht (vgl. [Floyd 93]). Das System kann in ein **Kernsystem**, das zuerst entwickelt wird, und **Ausbaustufen**, die modular in das Kernsystem eingefügt werden können, gegliedert werden. Wichtig bei der Entwicklung sind **Autor-Kritiker-Zyklen**, bei denen die Entwickler Rückmeldung von den späteren Anwendern des Systems bekommen. Grundlage hierfür sind **Prototypen** – lauffähige Programme, die bestimmte Eigenschaften des Systems vermitteln.

Zur Gliederung des Projektablaufs werden **Projektetappen** definiert, die Zeitpunkte festlegen, an denen bestimmte Teilziele des Projekts erfüllt sein sollen. Die Feinplanung geschieht mit Hilfe von **Referenzlinien**, die kleine Arbeitspakete bestimmen, die in einer festgelegten Qualität erstellt werden sollen.

4.1.2 JWAM

JWAM ist ein objektorientiertes Rahmenwerk für die Konstruktion von Anwendungssystemen nach dem WAM-Ansatz in der Programmiersprache Java ([JWAM 03]). Es ist entstanden aus Arbeiten am Arbeitsbereich Softwaretechnik des Fachbereichs Informatik der Universität Hamburg. Es wird in verschiedenen kommerziellen Projekten eingesetzt und dort sowie in Studien- und Diplomarbeiten an der Universität weiterentwickelt.

Das Rahmenwerk enthält softwaretechnische Umsetzungen der WAM-Elemente. Alle oben angesprochenen Entwurfsmetaphern finden sich darin wieder und bieten die Grundlage für konkrete Realisierungen in Anwendungssystemen. JWAM sieht verschiedene Abstraktionsebenen vor, um den Softwareentwurf zu vereinfachen, Komponenten wiederverwendbar und austauschbar zu machen und die Fehlerfindung zu beschleunigen. Wichtige Konstruktionsmuster sind beispielsweise die Aufteilung von Werkzeugen in weitgehend entkoppelte Komponenten sowie Fachwerte.

Bei Werkzeugen ist eine Trennung von Funktion und Interaktion möglich (siehe [Bleek et al. 99a]). Funktion bedeutet in diesem Fall die fachliche Funktionalität des

Werkzeugs, insbesondere die Bearbeitung der Materialien. Sie wird in der Funktionskomponente (FK) realisiert. Der Umgang des Werkzeugs ist Aufgabe der Interaktionskomponente (IAK). Sie reagiert auf Benutzereingaben, ruft an der FK Operationen auf und stellt die Ergebnisse dem Benutzer dar. Eine FK kann von mehreren Interaktionskomponenten gleichzeitig benutzt werden. Sie kennt diese nicht. Die Rückkopplung geschieht vielmehr über ein Callback System, bei dem sich die IAKs bei ihrer FK für bestimmte Ereignisse (wie z. B. die Änderung eines Materials) registrieren können und von ihr über deren Eintreten informiert werden.

Eine andere Komponente sind Fachwerte (siehe [Bleek et al. 99b]). Dieses sind anwendungsfachliche Werte wie z. B. Kontonummern oder Geldbeträge. Diese fachlichen Werte sollten nicht als Basiswerte realisiert werden, da diesen die fachlichen Umgangsformen fehlen, sie beliebige Manipulationen zulassen und keine Typsicherheit gegeben ist. Aber auch nicht als gewöhnliche Objekte, da bei ihnen wichtige Eigenschaften der Wertsemantik fehlen. Fachwerttypen bringen eine eigene Fabrik mit, an der sich konkrete Fachwerte erzeugen lassen und stellen die Wertsemantik und eine performante Implementierung sicher. Für die Erzeugung von Fachwerten gibt es verschiedene Möglichkeiten. [Sauer 01] gibt einen Überblick.

Die bei Erstellung dieser Arbeit aktuelle JWAM-Version ist 1.8.0. In Entwicklung befindet sich JWAM 2, das eine stärkere Aufteilung in Komponenten, die je nach Einsatzkontext eingesetzt werden können, aufweist.

4.2 Bekannte WAM-Leitbilder

Beim WAM-Ansatz steht die Kundenorientierung im Zentrum aller Überlegungen. Um Zufriedenheit beim Kunden zu erreichen, ist eine möglichst gute Gebrauchsqualität wichtig. Diese wird durch ein Zusammenspiel mehrerer Faktoren erreicht.

Eine wichtige Rolle spielen die Leitbilder und Entwurfsmetaphern. Ein **Leitbild** hat dabei nach [Züllighoven 98, S. 73] folgende Funktion: „Ein Leitbild in der Softwareentwicklung gibt im Entwicklungsprozeß und für den Einsatz einen gemeinsamen Orientierungsrahmen für die beteiligten Gruppen. Es unterstützt den Entwurf, die Verwendung und die Bewertung von Software und basiert auf Wertvorstellungen und Zielsetzungen.“

Leitbilder und dazu passende Metaphern definieren das **Benutzungsmodell** der Software. Dieses fachlich orientierte Modell umfaßt eine Vorstellung von der Handhabung und Präsentation der Software sowie unterstützte Konzepte und Abläufe (vgl. [Züllighoven 98, S. 71]).

Das klassische Leitbild des WAM-Ansatzes ist der **Arbeitsplatz für eigenverantwortliche Expertentätigkeit**. Ziel dabei ist die Unterstützung qualifizierter menschlicher Arbeit. Der fachliche Experte erledigt häufig wechselnde, komplexe Aufgaben, die eine hohe Qualifikation voraussetzen. Ihm sollen Mittel an die Hand gegeben werden, die ihm gestellten Aufgaben mit Hilfe des Systems zu lösen.

Es ist nicht sinnvoll, ein universelles Leitbild vorzugeben und auf alle Anwendungsbereiche zu übertragen. Vielmehr sollte der jeweilige Kontext berücksichtigt werden. Daher gibt es neben diesem Leitbild, das aus der Arbeit von Softwareentwicklern und Büroarbeit im Banken- und Dienstleistungsbereich entstanden ist, weitere WAM-Leitbilder, die für andere Anwendungsbereiche geeigneter sein können:

Funktionsarbeitsplatz für eigenverantwortliche Expertentätigkeit:
Die Eigenverantwortlichkeit des Anwenders bleibt bestehen. Allerdings ist nur eine Aufgabe oft durchzuführen. Dafür müssen dem fachlichen Experten sehr spezielle Mittel an die Hand gegeben werden. Mensch und Maschine sind in der Arbeitserledigung enger miteinander gekoppelt. Die vom Menschen benutzten Werkzeuge können auf eine gut bekannte Arbeitssituation zugeschnitten werden.

Gruppenarbeitsplatz für eigenverantwortliche, kooperative Aufgabenerledigung:
Die Kooperation mehrerer Mitarbeiter steht hier im Mittelpunkt. Ein Austausch von gemeinsamen Gegenständen in der realen Welt (z. B. Akten) wird dabei im Rechner virtuell nachgebildet. Ziel ist, den gemeinsamen Arbeitsablauf zu unterstützen.

Selbstbedienungsautomat:
Ein Beispiel für Selbstbedienungsautomaten sind Kundenselbstbedienungsautomaten im Bankenbereich. Ohne daß der Kunde ein Experte auf diesem Gebiet sein muß, kann er Standardfunktionen nutzen. Die Benutzerführung muß dafür einfach und unzweideutig sein. Die Bedienung des Anwendungssystems und die fachliche Aufgabe, die damit erledigt werden kann, müssen für den Benutzer klar ersichtlich sein, z. B. durch erläuternde Informations- und Hilfetexte.

4.3 Leitbilder für mobile Arbeit

Mobile Geräte weisen andere Charakteristika als stationäre Arbeitsplatzrechner auf. Folglich werden sie anders zur Aufgabenerledigung eingesetzt als Desktopsysteme. Daher muß geprüft werden, ob neue Leitbilder erforderlich sind oder ob die oben beschriebenen angepaßt und übernommen werden können.

Das Benutzungsmodell mobiler Anwendungen unterscheidet sich von dem von Desktop-Anwendungen in einer Reihe von Punkten, wie in Kapitel 3 genauer ausgeführt wurde. Der veränderte Einsatzkontext, unterschiedliche Erwartungen der Benutzer, Einschränkungen der Leistungsfähigkeit der Geräte, andere Ein- und Ausgabemethoden u. ä. erfordern eine Anpassung der Handhabung und Präsentation der Software.

Wichtige Anforderungen zusammengefaßt:

- Einfache, nicht zu komplexe Bedienung muß ermöglicht werden.
- Komplexere Berechnungen sind nicht möglich.

- Mobiles Gerät und Anwendungen sollten ohne lange Startphase einsetzbar sein.
- Fehlerfälle sollten vermieden werden.

Um damit umzugehen, sollten auch die Leitbilder den mobilen, veränderten Einsatzkontext widerspiegeln. Sie erinnern Anwendungsentwickler explizit an die Besonderheiten mobiler Geräte und ermöglichen von Beginn an eine Entwicklung, welche diese berücksichtigt, um funktionelle Anwendungen für den mobilen Einsatz zu erstellen. Auch für die Benutzer sind sie sinnvoll, damit diese die Entwicklung und den Einsatz der mobilen Anwendung besser verstehen und ihre Tauglichkeit beurteilen können. Sie sollten keine Anwendungen mit dem Funktionsumfang und der Leistungsfähigkeit von Desktopapplikationen erwarten.

Dabei ändert sich der Typ der mit dem WAM-Ansatz erstellten Anwendungen nicht grundlegend. Auf ihrem Gebiet erfahrene Benutzer können in eigener Verantwortung Aufgaben erledigen. Was sich ändert, ist der Umgang mit dem Computer. Dieser bildet nicht nur einen Arbeitsplatz ab, sondern wird vielmehr zu einem Teil eines generellen Arbeitsplatzes. Der Benutzer nimmt den Computer nicht mehr als Rechner war, auf dem in Software realisierte Werkzeuge ablaufen. Das *gesamte mobile Gerät* wird nach Aufruf eines Programms zu seinem Arbeitsgerät. Dies ist maßgeblich durch zwei Faktoren bedingt: Zum einen durch die handliche Form der Geräte, die in einigen Fällen auch speziell an eine zu erledigende Aufgabe angepaßt ist und den Computer als Maschine nicht offensichtlich werden läßt. Zum anderen durch die Verwendung außerhalb einer festen Arbeitsumgebung, was die Umwelt in die Arbeit einbezieht. Diese Entwicklung ist schon von Mark Weiser so vorausgesehen worden (siehe Abschnitt 2.1.1).

Die oben bei der Definition von Leitbildern als Basis angegebenen Wertvorstellungen und Ziele ändern sich ebenfalls. Daher sind eigene bzw. angepaßte Leitbilder gerechtfertigt, insbesondere unter Berücksichtigung des oben beschriebenen Gedankens der Vermeidung universeller Leitbilder für alle Bereiche.

Nach den Untersuchungen des Autors werden mobile Geräte hauptsächlich für zwei Einsatzzwecke benutzt: Als mobile Informationsgeräte, die wichtige persönliche und berufliche Informationen jederzeit und schnell abfragbar machen und zur einfachen Kommunikation mit anderen benutzt werden können, und als spezielle Arbeitsgeräte, die bei der Aufgabenerledigung vor Ort Daten bereitstellen und aufnehmen können sowie einfache Berechnungen mit ihnen ermöglichen.

Hieraus hat der Autor zwei Leitbilder abgeleitet: Das **mobile Arbeitsgerät für eigenverantwortliche Expertentätigkeit** und das **Funktionsgerät für eigenverantwortliche Expertentätigkeit im mobilen Einsatz**.

4.3.1 Leitbild „Mobiles Arbeitsgerät für eigenverantwortliche Expertentätigkeit“

Das mobile Arbeitsgerät stellt dem fachlichen Experten eine Umgebung zur Verfügung, in der er auch außerhalb seines Büros auf seine wichtigsten Daten zugreifen kann. Dies sind zum einen persönliche Informationen, die für die Organisation seiner Arbeit wichtig

sind, wie z. B. Termine, Adressen und Aufgaben. Zum anderen benötigt er Daten für die Erledigung seiner Aufgaben, die er mit seinem mobilen Gerät unabhängig vom Standort einsehen und bearbeiten kann. Beispielsweise können dies der Zustand der Server und des Rechnernetzes bei Systemadministratoren sein. Oder ein Versicherungsvertreter kann im Hause des Kunden Informationen zu gewünschten Dienstleistungen abfragen, wie z. B. ihre aktuelle Verfügbarkeit.

Geschäftliche Informationen stehen nicht nur zum Nachschlagen zur Verfügung. Daten lassen sich in speziell geeigneten Anwendungen digital erfassen, z. B. die von einem Kunden neu bestellten Versicherungspakete mit seinen persönlichen Daten. So werden Arbeitsschritte gespart und Kopierfehler bei der Übertragung in das Anwendungssystem vermieden. Je nach Bedarf stehen auch andere Anwendungen geringeren Umfangs zur Verfügung, so daß unterwegs jederzeit kleine Aufgaben erledigt werden können.

Hinzu kommt die Möglichkeit zur Kommunikation mit anderen. Wenn der Experte weitere Informationen benötigt oder weitergeben möchte, die nicht über Programmfunktionen zur Verfügung stehen, kann er darauf zurückgreifen. So kann er beispielsweise telefonisch einen Termin beim Kunden machen oder mit einer kurzen Textnachricht eine Zugreservierung für die Rückfahrt ins Büro tätigen.

Diese Visionen folgen weitgehend der Philosophie des mobilen Gerätes als Schweizer Armee-Messer (vgl. Abschnitt 3.2.4, S. 21), mit dem eine Vielzahl an Aufgaben erledigbar ist. Am Modell der Usage Spaces läßt sich dies gut deutlich machen. Die vorhandenen Usage Spaces werden durch einen siebten, die Arbeit, ergänzt (siehe Abbildung 4.1). Dieser stellt nicht nur Daten zur Verfügung, sondern kann auch weitergehende Arbeitsabläufe unterstützen. Daher geht er über den als Referenz genutzten Bereich Information hinaus.

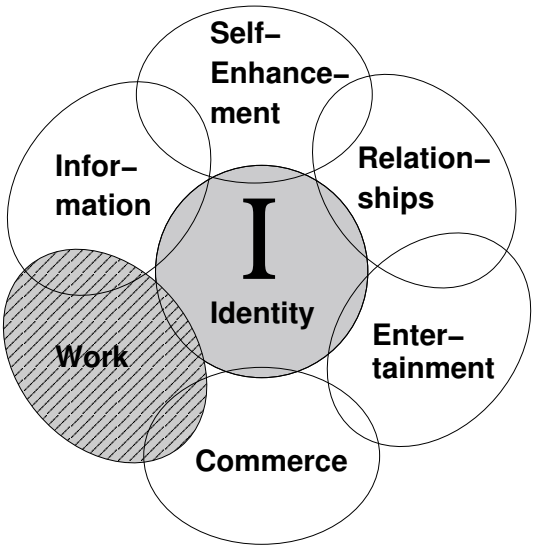


Abbildung 4.1: Usage Spaces mit siebtem Bereich Arbeit

So wie das von Marcus und Chen visionierte wirklich persönliche Gerät im privaten Bereich ist dieses Leitbild die Grundlage für persönliche Geräte im geschäftlichen Umfeld.

Wichtig für die Umsetzung dieses Leitbildes in konkrete Geräte ist die Eigenschaft, daß diese sich immer mitführen lassen und ohne lange Vorlaufzeit funktionieren, um jederzeit einsetzbar zu sein. Heutige Laptops mit recht hohem Gewicht, die zur Standardausstattung von Geschäftsleuten gehören, eignen sich dafür nicht. Besser sind leistungsfähige PDAs und zukünftige intelligente Mobiltelefone, die sich einfach um neue Funktionen erweitern lassen, wie es heutzutage mit J2ME (siehe Kapitel 6) schon eingeschränkt möglich ist.

Neben der Software sollte sich auch die Hardware durch Zusatzkomponenten u. ä. personalisieren und an die persönlichen Erfordernisse und Vorlieben anpassen lassen.

4.3.2 Leitbild „Funktionsgerät für eigenverantwortliche Expertentätigkeit im mobilen Einsatz“

Genauso wie beim stationären Funktionsarbeitsplatz hat hier der Experte eine klar definierte, spezielle Aufgabe mehrmals zu erfüllen. Er benötigt dafür ein System, das auf diese Arbeit zugeschnitten ist.

Einige Aufgaben können nicht von einem festen Arbeitsplatz aus, wie z. B. einem Büro, erledigt werden, sondern verteilen sich auf mehrere Orte. Beispielsweise benötigt ein Handlungsreisender sein Computersystem bei den einzelnen Kunden. Andere Aufgaben können nur mit tragbaren Geräten erledigt werden, oder es ist zumindest sinnvoll, diese einzusetzen. Arbeit außerhalb von Gebäuden ist hierfür ein Beispiel. Oft fallen beide Fälle zusammen, z. B. bei der Arbeit an wechselnden, schwer zugänglichen Orten.

Der fachliche Experte benötigt folglich bei seinem mobilen Einsatz die Möglichkeiten eines Funktionsarbeitsplatzes. Hierfür kann er ein mobiles Gerät einsetzen. Dieses ist auf diese Aufgabe spezialisiert.

Diese Beschreibung entspricht der information-appliance Philosophie (vgl. S. 21), bei der hauptsächlich *eine* Anwendung im Mittelpunkt steht. Bei der Realisierung einer konkreten Anwendung sollte das mobile Gerät so gestaltet werden, daß diese Aufgabe optimal unterstützt wird. Während dies beim stationären Funktionsarbeitsplatz hauptsächlich die Software betrifft, während die Hardware in der Regel ein Standardcomputer ist, kann ein mobiles Funktionsgerät auch hardwaremäßig auf die zu erledigende Aufgabe zugeschnitten werden, z. B. durch eine besondere Formgebung, spezielle Ein- und Ausgabemechanismen, integrierte Zusatzkomponenten u. ä.

Ein gutes Beispiel hierfür sind Tablett-PCs, die den Arzt bei seiner Visite unterstützen, indem sie Patientendaten, gestellte Diagnosen und Verordnungen anzeigen und neue aufnehmen. Der Arzt kann die mobilen Geräte mit einem Stift bedienen und so wie ihm bekannte Klemmbretter mit Papierformularen benutzen.

Die unterschiedliche Namensgebung der Leitbilder („mobiles Arbeitsgerät“ gegenüber „im mobilen Einsatz“) soll betonen, daß im ersten ein persönliches mobiles Arbeitsgerät jederzeit zugreifbar ist und bei Bedarf benutzt wird, während beim zweiten die zu erledigende, konkrete Arbeit vorhergeplant wird.

Abbildung 4.2 ordnet die neuen Leitbilder ein. Zu der bestehenden Differenzierung von vielfältiger Arbeit, bei der der Experte wechselnde Aufgaben zu erledigen hat, gegenüber spezieller Arbeit, bei der eine Aufgabe oft durchzuführen ist, kommt der Arbeitsort (stationär oder mobil) als zusätzliche Dimension hinzu.

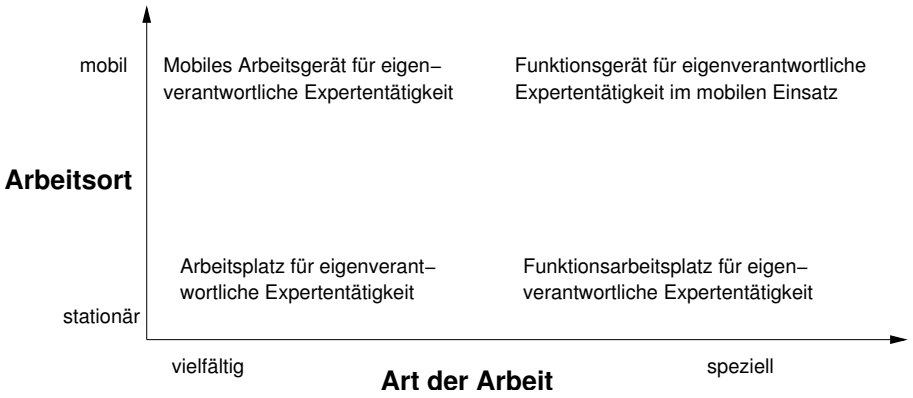


Abbildung 4.2: WAM-Leitbilder für mobile Anwendungen

4.4 Wichtige Metaphern

Metaphern sind Stilfiguren der Rhetorik, bei denen Worte in einem übertragenen, bildhaften Sinne gebraucht werden, um Beziehungen und Zusammenhänge aufzudecken. Zur Gestaltung von Softwaresystemen kann man **Entwurfsmetaphern** benutzen. Sie helfen, Leitbilder zu konkretisieren. Nach [Züllighoven 98, S. 79] dient eine Entwurfsmetapher „der Gestaltung von Softwaresystemen, indem sie Handhabung und Funktionalität für alle Beteiligten verständlicher macht.“

Eine wichtige Rolle spielen Metaphern auch als eine Technik des Extreme Programmings (XP), siehe [Beck 99]. Dort dienen sie als Architekturvorgaben, die Entwickler während der Arbeit anstelle von umfangreichen Architekturentwürfen benutzen.

4.4.1 Grundlegende WAM-Entwurfsmetaphern

Das Leitbild eines Arbeitsplatzes für eigenverantwortliche Expertentätigkeit des Werkzeug & Material-Ansatzes wird durch die beiden wichtigen Entwurfsmetaphern Werkzeug und Material getragen. Beim Entwurf eines Softwaresystems werden in der Realität vorhandene und virtuell neu geschaffene Gegenstände des täglichen Umgangs in die beiden Kategorien Werkzeug und Material eingeordnet.

Werkzeuge sind dabei Gegenstände, mit denen andere Gegenstände, die Materialien, bearbeitet werden können. So bearbeitet man oft Nägel mit dem Werkzeug Hammer und verändert dabei ihren Zustand oder man benutzt das Werkzeug Kalender, um Termine nachzuschlagen und sondiert so ihren Zustand.

Die Verwendung eines Werkzeugs muß vom Anwender erlernt werden. Dann kann er das Werkzeug benutzen, um verschiedene gleichartige Materialien zu bearbeiten. Er kann selbst entscheiden, wann er das Werkzeug benutzt und wie er es auf dem Arbeitsgegenstand einsetzt.

Materialien sind Gegenstände, die durch Bearbeitung zum eigentlichen Arbeitsergebnis werden. Sie müssen sich durch Werkzeuge bearbeiten lassen, oft auch durch mehrere verschiedene unter unterschiedlichen Aspekten. Meist lassen sich in der realen Welt vorhandene materialartige Gegenstände recht einfach in Softwarematerialien übertragen.

Die Einteilung von Gegenständen in die Kategorien Werkzeug und Material ist von der konkreten Arbeitssituation abhängig. Es gibt oft Fälle, in denen Gegenstände je nach Anwendung einmal als Material und dann als Werkzeug benutzt werden. In [Züllighoven 98, S. 86] findet sich das Beispiel des Bleistifts, der als Werkzeug zum Schreiben auf Papier verwendet werden kann und von Zeit zu Zeit mit dem Werkzeug Anspitzer als Material bearbeitet wird.

Die Übertragung von realen Gegenständen in Softwarematerialien erhält ihre wichtigsten Merkmale. Was die Gestalt und Handhabung von Werkzeugen betrifft, gibt es Änderungen. Diese müssen an das geänderte Medium angepaßt werden, ohne die fachliche Funktionalität zu beeinträchtigen. Materialien sind generell einfacher zu übertragen, doch gibt es auch hier technikbedingte Änderungen, wie z. B. die Einführung eindeutiger Identifikationsnummern o. ä.

Automaten arbeiten wie Werkzeuge ebenfalls auf Materialien. Allerdings arbeiten sie selbständig, d. h. ohne Interaktion mit dem Benutzer, vorgegebene, meist parametrisierbare Arbeitsschritte ab. Der Benutzer nimmt einige Einstellungen vor und startet dann den Automaten. Bei längeren Algorithmen kann dieser im Hintergrund laufen, während der Benutzer andere Arbeiten vornimmt.

Automaten werden für Arbeitsprozesse eingesetzt, die lästige Routinearbeit erfordern. Während der Anwender sich um Fälle kümmert, die er nicht delegieren kann, sind Automaten für die Standardfälle zuständig.

Eine weitere wichtige Metapher des WAM-Ansatzes ist die **Arbeitsumgebung**. In der realen Welt organisieren Menschen ihre Arbeit an einem festen Ort, ihrer Arbeitsumgebung. Dort können sie Gegenstände ablegen, anordnen und ihre Arbeit mit ihnen erledigen. In WAM-Systemen nehmen Arbeitsumgebungen Werkzeuge, Materialien und andere Software-Gegenstände auf. Ausführliches dazu findet sich bei [Lippert 99].

4.4.2 Metaphern auf mobilen Geräten

Es stellt sich die Frage, ob diese WAM-Metaphern auch für die Ausgestaltung der neuen Leitbilder für mobile Anwendungen benutzt werden können. Dies hätte den großen Vorteil, daß Anwendungen für stationäre Arbeitsplatzsysteme und mobile Geräte nach den gleichen grundlegenden Konzepten entwickelt werden könnten und vorhandenes Wissen der Entwickler angewendet werden würde.

Die Übernahme der Entwurfsmetaphern erscheint bei kleinen mobilen Geräten, wie z. B. PDAs, nicht selbstverständlich. Der Arbeitskontext und die Gerätecharakteristika sind sehr verschieden. In einer anderen Umgebung haben sich die Metaphern beispielsweise nicht bewährt:

Vergleich mit dem Webtop

In [OttoSchuler 00] wurde untersucht, inwieweit die Werkzeug-Metapher für Webtop-Anwendungen tragbar ist. Unter Webtop versteht man die Anzeigefläche innerhalb eines Web-Browsers. Die von diesem zu visualisierende Oberfläche wird von einem entfernten Server übertragen. Die Kommunikation zwischen Client und Server funktioniert nach dem Request-Response Prinzip, nach dem alle Aktionen vom Client ausgehen und die Webtop-Applikation nicht von sich aus mit dem Client kommunizieren kann.

Es wurden dort einige Bruchstellen gefunden, die hier kurz als Ausgangspunkt für eigene Überlegungen im mobilen Kontext zusammengefaßt werden:

- Während Werkzeuge abhängig von der Situation verwendet werden können und die einzelnen Bearbeitungsschritte nicht fest vorgegeben sind, geben Webtop-Anwendungen das Schema der Abläufe vor und sind sehr unflexibel.
- Im Gegensatz zu Werkzeugen, die von einem Benutzer verwendet werden, sind Webtop-Anwendungen immer mehrbenutzerfähig.
- Werkzeuge besitzen ein Gedächtnis, in dem ihre momentanen Einstellungen gesichert sind. Aufgrund der Mehrbenutzerfähigkeit und der eingeschränkten Kommunikationsfähigkeit von Webtop-Anwendungen muß hier bei jeder Anfrage der volle Zustand übergeben werden.
- Werkzeuge können Materialien feingranular bearbeiten. In Webtop-Anwendungen werden mehrere dieser Änderungen zu größeren Operationen zusammengefaßt.
- Werkzeuge geben unmittelbare Rückkopplung über ausgeführte Aktionen. Webtop-Anwendungen können dies aufgrund ihrer eingeschränkten Möglichkeiten oft nur begrenzt und verzögert.

Diese Unterschiede haben dazu geführt, daß die Werkzeug-Metapher als nicht sinnvoll anwendbar für Webtop-Anwendungen eingestuft wurde. Stattdessen entstand die Metapher der Fachlichen Services (siehe Abschnitt 4.5.2).

Die aufgeführten Punkte sind für mobile Anwendungen nicht relevant. Dadurch, daß die mobilen Geräte Algorithmen lokal ausführen können und nicht bei jeder Aktion ein Server kontaktiert werden muß, ist eine kontinuierliche und flexible Bearbeitung von Materialien möglich. Ein Werkzeuggedächtnis kann auf den mobilen Geräten implementiert werden. Einstellungen und Daten können lokal gespeichert werden.

Es stehen eine Vielzahl von Elementen wie Menüs, Buttons und Listen zur Verfügung. Sie bieten eine unmittelbare Rückkopplung für den Benutzer auch bei feingranularen Änderungen. Die Interaktionsmöglichkeiten sind gegenüber Webanwendungen somit deutlich größer.

Ein Werkzeug wird nur von einem Benutzer gesteuert, Mehrbenutzerfähigkeit ist nicht vorgesehen. Somit trifft auch dieser Punkt bei mobilen Anwendungen nicht zu.

Wie in Abschnitt 4.5 gezeigt wird, stellen Fachliche Services auf Servern in Kombination mit Werkzeugen auf mobilen Geräten eine gut einsetzbare Architektur dar. Auch die Verwendung von Webtops auf mobilen Geräten ist möglich. Allerdings nutzen sie, wie oben ausgeführt, deren Möglichkeiten nicht aus. Außerdem stehen für mobile Geräte in der Regel nur sehr eingeschränkte Web-Browser zur Verfügung. Einige sind auf WAP-Seiten beschränkt. Mit ihnen fällt es schwer, eine sinnvolle Request-Response Interaktion zu realisieren (siehe auch Abschnitt 2.3.2 auf S. 14).

Berücksichtigung der Besonderheiten mobiler Geräte

Die für den Webtop kritischen Punkte haben also für die Metaphernbildung auf mobilen Geräten keinen Einfluß. Jetzt bleibt zu untersuchen, inwieweit die Charakteristika der mobilen Geräte etwas anderes daran ändern: Eingeschränkte oder nicht vorhandene Netzanbindung, geringe Rechengeschwindigkeit, wenig Speicher, unkonventionelle Eingabemöglichkeiten, geringe Größe und Farbauflösung des Anzeigebereiches.

Die **Einschränkungen der Netzanbindung** müssen entsprechend berücksichtigt werden. Während Desktoprechnern im Regelfall ein permanenter und leistungsfähiger Netzzugriff möglich ist, sind mobile Geräte hier sehr viel limitierter (siehe Abschnitt 3.2.2).

Da größere Anwendungen nicht nur auf einem Gerät laufen, sondern Informationen mit ihrer Umgebung austauschen, bedeutet dies, daß eine Synchronisierung mit einem stationären Gerät vor und nach der mobilen Arbeit oder ein online-Zugriff auf Server nötig ist. Auch die Werkzeuge müssen so ausgelegt sein, daß sie diesen Sachverhalt berücksichtigen. Insbesondere ist zu beachten, daß aktuellere Änderungen von anderen Anwendern an denselben Materialien nicht sichtbar sind.

An den Metaphern Werkzeug und Material sind aber bis auf die zusätzlichen Synchronisations- bzw. Verbindungsschritte keine Änderungen vorzunehmen. Um diese zu berücksichtigen, wird weiter unten das Entwurfsmuster **Synchronisierer** eingeführt, siehe Abschnitt 5.3.

Die **geringe Rechengeschwindigkeit** mobiler Geräte führt dazu, daß lokale komplexe Berechnungen nur eingeschränkt möglich sind. Aus den im Kapitel 3 bespro-

chenen Überlegungen heraus wird der Anwender meistens nicht bereit sein, lange auf Ergebnisse zu warten. Er erwartet direktes Feedback.

Daher sind insbesondere Automaten, die direkt auf mobilen Geräten laufen, nicht sinnvoll, da diese in der Regel gerade das Merkmal besitzen, aufwendige Berechnungen durchzuführen und dadurch leistungsstarke Prozessoren erfordern. Aufgrund der geringen Rechenleistung ist auch die Ausführung eines Automaten im Hintergrund nicht möglich. Als Alternative bietet es sich an, die eigentliche Berechnung auf leistungsfähigere Server zu übertragen, auf die über das Netz zugegriffen werden kann. Gerade bei Automaten ist hier kein Bruch im Benutzungsmodell zu sehen, da ihr Anstoß lokal erfolgt, die Verarbeitung transparent für den Benutzer auf dem Server erledigt wird, während er andere Aufgaben bearbeiten kann, und das Ergebnis wieder auf das Gerät übertragen wird. Nötig hierzu ist eine entsprechend ausgelegte Netzanbindung (siehe Diskussion oben).

Werkzeuge benötigen im allgemeinen nicht soviel Rechenleistung. Wichtig bei ihnen ist, daß der Benutzer ein unmittelbares Feedback erhält. Hierfür muß zumindest die Bedienoberfläche ausreichend schnell sein. Längere Berechnungen sind zu vermeiden. Falls dies nicht möglich ist, ist der Benutzer über den Stand der Bearbeitung zu informieren. Die Wartebereitschaft der Anwender ist auf mobilen Geräten generell niedriger als auf Desktoprechnern, was erschwerend hinzukommt.

Im Gegensatz zu Desktopanwendungen können meist nicht mehrere Werkzeuge parallel ausgeführt werden. Einige Betriebssysteme (wie z. B. PalmOS) sehen dies gar nicht vor.

Die **geringe Speicherkapazität** hat zur Folge, daß sowohl die installierten Programme als auch die dazugehörigen Daten in ihrem Umfang beschränkt sind. Das heißt für Werkzeuge und Automaten, daß sie nicht sehr groß werden dürfen und daß sie möglichst keine umfassenden externen Bibliotheken benutzen sollten, da diese viel Speicherplatz belegen können. Nur die nötigsten Materialien können auf dem mobilen Gerät zur Verfügung stehen. Umfangreiche Datenbestände müssen so aufbereitet werden, daß nur die für die momentane Arbeit wichtigen übertragen werden, entweder bei der Synchronisierung vor und nach dem Einsatz oder online.

Auf einigen mobilen Geräten sind **unkonventionelle Eingabemöglichkeiten** vorhanden. Diese müssen eine passende Bedienung der Werkzeuge ermöglichen. Insbesondere ergibt sich das Problem, daß keine direkte Manipulation möglich ist, wenn keine Zeigegeräte wie Mäuse und Stifte zur Verfügung stehen. Daher sehen Werkzeuge für Geräte, die nur über Tasten verfügen, wie beispielsweise Mobiltelefone, anders aus. Allerdings erscheinen immer mehr dieser Geräte mit Navigationstasten für alle vier Richtungen oder kleinen Joysticks, die mehr Möglichkeiten bieten.

Die im Vergleich zu Desktopsystemen **geringe Größe des Anzeigebereichs** beschränkt die Fähigkeiten, komplexe Werkzeuge anzuzeigen. Sehr oft wird ein Werkzeug nicht ganz auf dem Bildschirm angezeigt werden können, sondern mehrere Seiten umfassen. Die **geringe Farbauflösung** erschwert die Darstellung zusätzlich. Insbesondere

wenn nur schwarz/weiß oder Graustufen zur Verfügung stehen, wird die Hervorhebung einzelner Elemente problematisch.

Entscheidend ist die Frage, ob trotz dieser Beschränkungen Werkzeuge als solche wahrgenommen werden können. Sie müssen speziell für kleine Bildschirme entwickelt sein und dem Benutzer das Gefühl geben, das Material direkt manipulieren zu können.

Desweiteren ändert sich der Umgang mit Werkzeugen, da Hardware und Software auf mobilen Geräten verstärkt zu einer Einheit verschmelzen, wie schon auf S. 34 beschrieben. In einigen Fällen werden auch die Materialien nicht mehr nur in Software abgebildet sein, sondern auch mit mobilen Geräten direkt in der Realität bearbeitet werden. Beispielsweise kann ein Wasserstandszähler über die serielle Schnittstelle eines mobilen Gerätes ausgelesen werden und der Stand im Werkzeug für die Berechnung des Verbrauchs genutzt werden.

Wenn die zusätzlichen Möglichkeiten durch Hardwaredesign und Peripheriegeräte gut genutzt werden, kann für den Benutzer also ein noch intensiverer Eindruck entstehen als mit Werkzeugen am Desktop-PC.

Schwierig wird die Abbildung der Arbeitsfläche. Erfahrene Benutzer von Desktop--PCs sind daran gewöhnt, auf ihrem virtuellen Arbeitsplatz Icons anzuordnen und mit direkter Manipulation die dahinterstehenden Elemente zu bearbeiten. Auf PDA-Bildschirmen – und erst recht auf noch kleineren Bildschirmen, wie die von Mobiltelefonen – reicht der zur Verfügung stehende Platz kaum aus, um dies ebenso abzubilden. Es wurden jedoch Möglichkeiten entwickelt, auch unter diesen Voraussetzungen die Metapher Arbeitsumgebung umzusetzen. So lassen sich Anwendungen und Daten in Kategorien einsortieren, von denen immer nur eine gleichzeitig angezeigt wird, und die Arbeitsumgebung verfügt über ein Menü, über das nicht darstellbare Funktionen ausgewählt werden können. Bei Mobiltelefonen finden sich hierarchische Menüstrukturen anstelle von graphischen Arbeitsumgebungen. Dort ist die Metapher nur sehr eingeschränkt zu realisieren.

Zusammenfassung

Die bedeutendsten WAM-Metaphern lassen sich wie oben aufgezeigt auf mobile Kontexte übertragen. In Tabelle 4.1 sind die wichtigsten Punkte, die zu beachten sind, aufgeführt.

4.5 Multi-Channeling

Im WAM-Kontext steht Multi-Channeling (dt. Mehrkanalfähigkeit) für die Anbindung verschiedenartiger Benutzungsschnittstellen an große Anwendungssysteme. Hauptidee dabei ist, daß die Funktionalität unabhängig von der konkreten Handhabung, Präsentation und Technik bereitgestellt wird. Hierzu dienen Fachliche Services.

In diesem Abschnitt wird untersucht, inwieweit Multi-Channeling geeignet ist, Anwendungen auf mobilen Geräten in Anwendungssysteme einzubetten. Das grundlegende Architekturmodell wird vorgestellt und mobile Geräte dort eingeordnet. Anschlie-

Metapher	Eignung für mobile Anwendungen
Werkzeug	- Metapher Werkzeug ist gut übertragbar, da keine Einschränkungen wie z. B. beim Webtop vorhanden sind - Werkzeuge sollten nicht zu komplex sein und sich einfach bedienen lassen - Hardware und Software des Gerätes verschmelzen zu einem als Einheit betrachteten Werkzeug - Parallele Arbeit mit verschiedenen Werkzeugen ist in der Regel nicht möglich
Material	- Im Regelfall gut umsetzbar, da sie keine visuelle Komponente besitzen - Synchronisation zum Abgleich von Datenänderungen nötig
Automat	- Nur kleine lokale Automaten sinnvoll - Längere Berechnungen sollten besser auf Servern laufen
Arbeitsumgebung	- Aufgrund der beschränkten Platzverhältnisse auf den Bildschirmen nicht in der vom Desktop gewohnten Form graphisch zu realisieren - Etablierte Drag & Drop-Funktionalität ist auf einigen Geräten schwierig umsetzbar - Trotzdem sinnvoll zur Sammlung von Werkzeugen und Materialien an einer Stelle

Tabelle 4.1: Eignung bestehender WAM-Metaphern für mobile Anwendungen

ßend werden verschiedene Typen Fachlicher Services erläutert und beschrieben, wie sie für die Erstellung von mobilen Anwendungen verwendet werden können.

4.5.1 Architekturmodell

Die Architektur von Multi-Channeling Systemen läßt sich wie in Abbildung 4.3 gezeigt in drei Schichten einteilen.

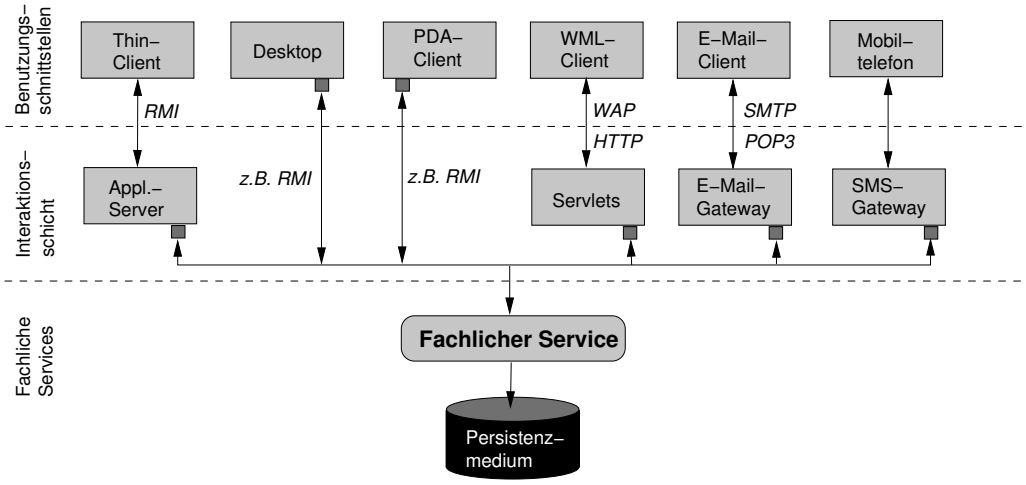


Abbildung 4.3: Multi-Channeling: Struktur mit Fachlichen Services

Die Schicht der **Fachlichen Services** stellt Dienstleistungen zur Verfügung, die von den darauf aufbauenden Schichten genutzt werden können. Um eine Anbindung verschiedener Systeme zu ermöglichen, enthalten sie im idealen Fall keine Annahmen über die anzuschließenden Systeme (Frontend und Backend). In der Regel wird hier auch auf das Persistenzmedium zugegriffen und die Materialien werden verwaltet.

Die **Interaktionsschicht** kapselt die Realisierung verschiedener Interaktionsformen inklusive ihrer technischen Umsetzung. Für „klassische“ WAM-Werkzeuge mit ihrer IAK/FK-Trennung ist dies die Funktionskomponente. Für Geräte und Techniken, die nach dem Client/Server-Prinzip angeschlossen werden, sind dies die Server-Komponenten wie Servlets oder Mail-Systeme.

Die oberste Schicht realisiert verschiedene **Benutzungsschnittstellen**. Diese richten sich nach den spezifischen Anforderungen der Geräte und der dort eingesetzten Techniken. Auf Desktop-PCs können dies Interaktionskomponenten sein, welche die Funktionskomponenten der darunterliegenden Interaktionsschicht benutzen, oder Web-Browser mit Java-Applets, die ihre Daten von Servlets empfangen.

Eine extreme Ausprägung dieses Konzepts stellen Thin Clients dar, die nur eine sehr geringe Eigenintelligenz besitzen und eng mit einem Server gekoppelt sind, der ihnen Informationen zur Verfügung stellt (vgl. [Bohlmann 00]). Im extremsten Fall wird schon die Bedienoberfläche auf dem Server der Interaktionsschicht aufbereitet

und der Thin Client kann nur mit elementaren Grafikoperationen (auf Widget-Ebene oder noch elementarer) umgehen. Dies wird auch mit dem Begriff UltraThin Client bezeichnet.

Das vorgestellte Modell hat den Vorteil, daß viele verschiedene Benutzungsumgebungen unterstützt werden können, ohne daß die zugrundeliegende Funktionalität jedesmal erneut implementiert werden muß. Mit Fachlichen Services können neue Kanäle hinzugefügt werden, indem ihre Interaktion und Benutzungsschnittstelle auf der Grundlage der vorhandenen Services implementiert werden. Wenn sie ähnliche Interaktionsformen aufweisen, reicht auch nur die Erstellung einer neuen Bedienoberfläche.

4.5.2 Fachliche Services

Otto und Schuler definieren Fachliche Services (manchmal auch Dienstleister genannt) in [OttoSchuler 00, S. 67] wie folgt: „Ein Fachlicher Service fasst kohärente Funktionalität und Wissen eines Anwendungsbereichs unabhängig von einer Benutzungsschnittstelle zusammen, kapselt sie und stellt sie als Dienstleistungsbündel zur Verfügung. Die Dienstleistungen eines Fachlichen Services werden über Aufträge von Konsumenten in Anspruch genommen.“

Sie unterscheiden zwei Ausprägungen Fachlicher Services:

Bereitstellung von Material und Umgang: Materialien werden verwaltet und ggf. persistent gespeichert. Zusätzlich werden fachliche Dienste bereitgestellt, die auf ihnen arbeiten. Die Materialien werden entweder durch den Fachlichen Service gekapselt, der alle nötigen Operationen anbietet, oder können als Kopien nach außen gegeben werden.

Vergegenständlichung von Vorgängen: Wissen über die Abarbeitung von Vorgängen mit möglichen Reihenfolgen, Zuständigkeiten und Implikationen wird gekapselt. In der Regel besitzen solche Fachlichen Services einen fachlichen Zustand, der sie von reinen Funktionsbibliotheken unterscheidet.

Fachliche Services bieten eine Reihe von Vorteilen:

- Sie eignen sich sowohl als Konzept zur Implementation von Anwendungen auf Plattformen, für welche die Werkzeugmetapher nicht tragfähig ist, als auch als Grundlage von Werkzeugen.
- Sie kapseln fachliche Funktionalität und fachliches Wissen an einem Ort und stellen es mehreren Benutzern zur Verfügung.
- Sie bieten ihre Funktionalität handhabungs- und präsentationsunabhängig an und können so von verschiedenartigen Endgeräten angesprochen werden.

- Durch die Zusammenfassung und Kapselung fachlicher Funktionalität in Fachlichen Services wird sie wiederverwendbar und austauschbar.

Im Gegensatz zu Werkzeugen, die von *einem* Benutzer bedient werden, sind Fachliche Services auf Mehrbenutzerfähigkeit ausgelegt. Dies hat Auswirkungen auf ihre technische Realisierung. Operationen müssen atomar sein, damit sie sich nicht gegenseitig stören. Services können mit Transaktionen, die mehrere Operationen zusammenfassen, oder Sessions, die eine andauernde logische Verbindung zwischen Service und Kunde darstellen, realisiert sein oder zustandslos implementiert werden. Zustandslose Services merken sich keine Informationen über Kunden zwischen mehreren Aufrufen. Details hierzu finden sich in [Nilius 02, S. 68ff].

4.5.3 Integration mobiler Geräte

Bei der Einbindung mobiler Geräte in Multi-Channeling Systeme lassen sich zwei Fälle grundsätzlich unterscheiden:

1. Mobile Anwendungen, die über eine online-Verbindung zu Servern des Anwendungssystems verfügen und
2. Mobile Anwendungen, die den Datenaustausch mit dem Anwendungssystem über eine regelmäßige Synchronisierung durchführen und im mobilen Einsatz keine Verbindung herstellen können.

Der **erste Fall** paßt gut in das oben skizzierte Architekturmodell, bei dem eine Serverkomponente der Interaktionsschicht mit einer konkreten Benutzungsschnittstelle als Client gekoppelt ist. Die mobile Anwendung greift über ihre Interaktionsschicht auf Fachliche Services zu, um Berechnungen durchzuführen. Nur die Darstellung der Bedienoberfläche und einfache Berechnungen erfolgen auf dem Gerät.

Für mobile Anwendungen eignet sich das oben beschriebene Thin Client Modell nur sehr begrenzt, da heutzutage die Netzanbindung noch zu langsam, fehleranfällig und teuer ist (vgl. Mobile Kommunikation in Abschnitt 3.2.2). Deshalb sollte das mobile Gerät soviele Berechnungen wie im Rahmen seiner Leistungsfähigkeit möglich sind selber übernehmen.

Auf die einsetzbare Technik soll hier nicht weiter eingegangen werden. Es gibt aber eine ganze Reihe von erprobten Technologien, die verwendbar sind, z. B. eine Kommunikation über Java Servlets oder die Remote Method Invocation (RMI) von Java.

Im **zweiten Fall** werden Daten nur bei der Synchronisierung ausgetauscht. Im mobilen Einsatz kann kein entfernter Fachlicher Service aufgerufen werden. Dies bedeutet, daß Fachliche Services auf dem Gerät selber implementiert werden müssen. Dazu ist Voraussetzung, daß das Gerät leistungsfähig genug ist und daß dem Fachlichen Service alle benötigten Ressourcen zur Verfügung gestellt werden können. Da beispielsweise Fachliche Services oft auf ein Persistenzmedium zurückgreifen, muß dieses auch in

geeigneter Form auf dem mobilen Gerät realisiert werden können. Für einen Lösungsvorschlag siehe die Diskussion eines passenden Konstruktionsmusters auf Seite 55.

In diesem Szenario entfällt der positive Aspekt Fachlicher Services, daß sie an einer zentralen Stelle laufen und verschiedene Endgeräte unterstützen können. Dennoch bieten sie in der Konstruktion Vorteile, indem die gleiche fachliche Funktionalität im Anwendungssystem und in der mobilen Anwendung genutzt werden kann. Die Entwicklung der Funktionalität muß nur einmal erfolgen. Durch die Zusammenfassung im Fachlichen Service ist sie leicht austauschbar.

Bei der Synchronisierung müssen die Daten der Fachlichen Services im System und diejenigen auf dem mobilen Gerät abgeglichen werden. Je nach ihrer Komplexität und dem Grad der parallelen Bearbeitung kann dies eine aufwendige Operation sein. Sie ist allerdings notwendig, um die Konsistenz der Daten sicherzustellen (vgl. Abschnitt 5.3 über Synchronisation).

4.6 Zusammenfassung

In diesem Kapitel wurde gezeigt, daß sich viele Grundlagen des WAM-Ansatzes auf Anwendungen für mobile Geräte übertragen lassen. Der Autor hat neue, aus den vorhandenen abgeleitete Leitbilder für den mobilen Einsatz vorgeschlagen, das „Mobile Arbeitsgerät für eigenverantwortliche Expertentätigkeit“ und das „Funktionsgerät für eigenverantwortliche Expertentätigkeit im mobilen Einsatz“. Anschließend wurde nachgewiesen, daß die grundlegenden Entwurfsmetaphern ebenfalls übernommen werden können, wobei bei einigen gewisse Einschränkungen zu machen sind.

Mobile Anwendungen lassen sich in eine Multi-Channeling Architektur einbinden. Sie können so Fachliche Services nutzen, die Funktionalität zur Verfügung stellen. Hierfür wurden zwei Realisierungsalternativen diskutiert: den online-Zugriff auf Fachliche Services, die auf entfernten Servern laufen, und die lokale Ausführung Fachlicher Services auf den mobilen Geräten.

Kapitel 5

Konstruktionsmuster

Nach der Betrachtung der Grundlagen von WAM für mobile Anwendungen im letzten Kapitel ist nun zu prüfen, wie die Realisierung von Anwendungen auf mobilen Geräten konstruktiv aussehen kann. Hierbei soll davon ausgegangen werden, daß Entwickler möglichst viel aus ihrer Erfahrung mit Desktop-WAM-Anwendungen übernehmen möchten, sofern dies sinnvoll ist. In vielen Fällen gibt es auch schon ein lauffähiges WAM-System oder eines in Entwicklung, das um eine mobile Variante ergänzt werden soll.

Hier wird zunächst dieser häufiger auftretende Fall betrachtet, bei dem eine mobile Anwendung zusätzlich zu einem schon vorhandenen oder gerade im Entstehen befindlichen System entwickelt wird. Danach werden die Unterschiede untersucht, falls nur eine singuläre mobile Anwendung benötigt wird.

5.1 Mobile Anwendung im Rahmen eines größeren Anwendungssystems

Wenn eine Anwendung auf mobilen Geräten als Komponente eines großen Anwendungssystems konzipiert wird, stellt sich die Frage, ob Teile stationär und mobil gemeinsam genutzt werden können. Im folgenden wird erörtert, inwieweit das sinnvoll und möglich ist.

Es bietet Vorteile, wenn die Architektur und die Implementierung nur möglichst wenig geändert werden müssen:

- Die Programmierer können die Entwicklung der mobilen Anwendung basierend auf ihren Erfahrungen mit dem stationären Anwendungssystem angehen. Sie müssen sich nicht zusätzlich zur Änderung der Zielplattform mit einer anderen Architektur beschäftigen. Auch eine parallele Entwicklung von stationärem und mobilem System wird dadurch besser unterstützt.
- Für den Anwender wird die Benutzung der mobilen Anwendung erleichtert, wenn er dort aus der stationären Anwendung gewohnte Begriffe und Konzepte vorfindet. Sofern die Handhabung und Präsentation der stationären Anwendung auch für mobile Geräte geeignet ist, bietet eine weitgehende optische Übereinstimmung zusätzliche Vorteile.

- Viel Arbeit, die schon zur Erstellung des Anwendungssystems geleistet wurde, muß für die mobile Anwendung nicht noch einmal erbracht werden. Der durch die Übernahme von Code weggefallene Aufwand kann in eine bessere Anpassung an das Gerät und das damit verbundene Benutzungsmodell investiert werden.

- Wenn schon existierende, gut getestete Codeteile, die ihre Tauglichkeit bereits im Anwendungssystem bewiesen haben, übernommen werden können, können Fehler vermieden werden. Es muß aber sichergestellt werden, daß durch die Code-Migration keine neuen Fehler entstehen.

- Falls das stationäre Anwendungssystem und die mobile Anwendung zeitlich parallel entwickelt werden, ist es von Vorteil, wenn eine gemeinsame Codebasis verwendet wird. Eine Abkopplung der Entwicklung des mobilen Systems kann hingegen zu Inkonsistenzen und dadurch zu Fehlern an den Schnittstellen führen.

Nachteilig kann es sich auswirken, wenn eine Architektur übernommen wird, die für mobile Anwendungen nicht geeignet ist. Auch sollte Code, der viele Teile enthält, die für die mobile Anwendung nicht genutzt werden können, überarbeitet werden. Es muß ebenfalls immer darauf geachtet werden, daß nicht implizit ein Benutzungsmodell für die mobile Anwendung übernommen wird, das nicht angemessen ist.

Eine Wiederverwendung von bestehenden Teilen für die mobile Anwendung ist der Erfahrung des Autors nach in WAM-Systemen aus fachlichen und technischen Gründen häufig sinnvoll und möglich:

Fachliche Gründe

Im Entwicklungsprozeß eines Systems werden Werkzeuge und Materialien identifiziert. In vielen Projekten ist es erforderlich und sinnvoll, die gleichen Materialien mit den gleichen Werkzeugen sowohl im stationären System als auch auf dem mobilen Gerät zu bearbeiten.

Beispielsweise wird oft die Arbeit am Desktop-Rechner vorbereitet. Mit dem mobilen Gerät werden diese Informationen vor Ort genutzt; neue Daten werden erhoben. Wiederum am Desktop-Rechner können diese kontrolliert, nachbereitet und ergänzt werden. Hierfür können in der Regel die gleichen Werkzeuge verwendet werden. Wenn dies nicht sinnvoll ist, kann ggf. eine abgespeckte und angepaßte Version der Desktop-Werkzeuge auf den mobilen Geräten genutzt werden.

Wenn eine Service-Architektur eingesetzt wird (siehe Abschnitt 4.5), wird das fachliche Wissen eines Bereiches in einem Fachlichen Service gebündelt. Dieser ist so angelegt, daß er sowohl von Desktop-Werkzeugen als auch von Werkzeugen der mobilen Anwendung genutzt werden kann.

Technische Gründe

WAM-Systeme weisen einige Abstraktionsschichten auf. Die Funktionalität wird durch gekapselte Komponenten erbracht. Im Inneren sind diese kohärent, während sie mit anderen Komponenten möglichst lose gekoppelt sind. Dies erleichtert die Wiederverwendung einzelner Elemente und die Anpassung an eine veränderte Technik. Im folgenden Abschnitt wird dies genauer betrachtet.

Anwendungen nach WAM werden in der Regel unter Benutzung eines Rahmenwerkes erstellt, z. B. JWAM für WAM-Anwendungen in der Programmiersprache Java. Dieses gibt die grundlegende Architektur vor. Aufgrund dieses Wissens können Teile der Anwendung leichter an geänderte Bedingungen wie den mobilen Kontext angepaßt werden, als wenn die Architektur des Anwendungssystems nicht festgelegt wäre.

5.1.1 Kombination von Werkzeugen, Materialien und Fachlichen Services

Die folgenden Erläuterungen beschränken sich auf den Hauptfall, bei dem die mobile Anwendung mit den Metaphern Werkzeug, Material und Fachlicher Service entwickelt wird. Anwendungen nach dem Webtop-Modell oder mit E-Mail- und Kurznachrichten-Clients werden wegen ihrer prinzipiellen Einschränkungen nicht betrachtet (siehe Abschnitt 2.3).

Werkzeuge nach WAM bestehen aus typischen Komponenten, die bestimmte Aufgaben erfüllen und dabei Dienstleistungen anderer Komponenten nutzen:

Funktionskomponente (FK): Funktionskomponenten sind für die fachliche Funktionalität zuständig. Sie arbeiten direkt auf Materialien oder nehmen Fachliche Services in Anspruch. Ihre Operationen können sie mehreren Interaktionskomponenten gleichzeitig anbieten. Dabei registrieren sich Interaktionskomponenten bei Funktionskomponenten. Bei Änderungen an den Materialien werden sie von ihnen informiert. Somit wird eine Entkopplung von FK und IAK erreicht.

Interaktionskomponente (IAK): Interaktionskomponenten enthalten die Handhabung von Werkzeugen. Sie greifen auf jeweils eine Funktionskomponente zurück und können Materialien nicht direkt verändern. Ihre Realisierung erfolgt unabhängig von einer speziellen Repräsentation.

Funktionskomponente und Interaktionskomponente können auch zusammengefaßt sein. Man spricht dann von monolithischen Werkzeugen (engl. monotools).

Graphische Bedienoberfläche (GUI): Die Oberflächen legen die konkrete graphische Repräsentation und Bedienung der Interaktion fest. Beispielsweise wird hier die Position, Größe und Ausrichtung von Widgets bestimmt.

Technische Details zur Kopplung von Interaktionskomponenten und Oberflächen finden sich im Kasten auf Seite 57.

Werkzeuge können auch andere Werkzeuge als Unterkomponenten enthalten. Diese stellen sowohl Funktionalität als auch Repräsentation zur Verfügung.

Werkzeuge arbeiten auf **Materialien**. Diese sind eigene Gegenstände und nicht Bestandteil von Werkzeugen. Werkzeuge sprechen verschiedene Typen von Materialien unter gemeinsamen Aspekten an.

Wie in Abschnitt 4.5.2 beschrieben, kapseln **Fachliche Services** Funktionalität und stellen sie den Funktionskomponenten von Werkzeugen zur Verfügung. Sie arbeiten dazu in der Regel direkt auf Materialien.

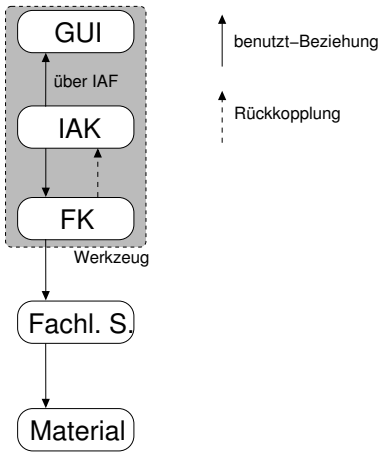


Abbildung 5.1: Hierarchie von Material, Fachlichem Service und Werkzeug

Mit diesen Elementen ergibt sich sich eine Hierarchie von aufeinander basierenden Komponenten, wie sie so, wie hier idealtypisch beschrieben, in der Praxis meist realisiert wird (siehe Abbildung 5.1):

- Materialien bilden die Grundlage.
- Fachliche Services benutzen und verwalten sie.
- Funktionskomponenten greifen auf die Dienste Fachlicher Services zurück und realisieren die Funktionalität von Werkzeugen.
- Interaktionskomponenten bedienen sich dieser Funktionalität und realisieren die Handhabung von Werkzeugen.
- GUIs übernehmen die Präsentation der Werkzeuge für den Anwender.

Die einander benutzenden Komponenten können für die Konstruktion mobiler Anwendungen verwendet werden. Je nach Einsatzkontext können einige aus dem bestehenden

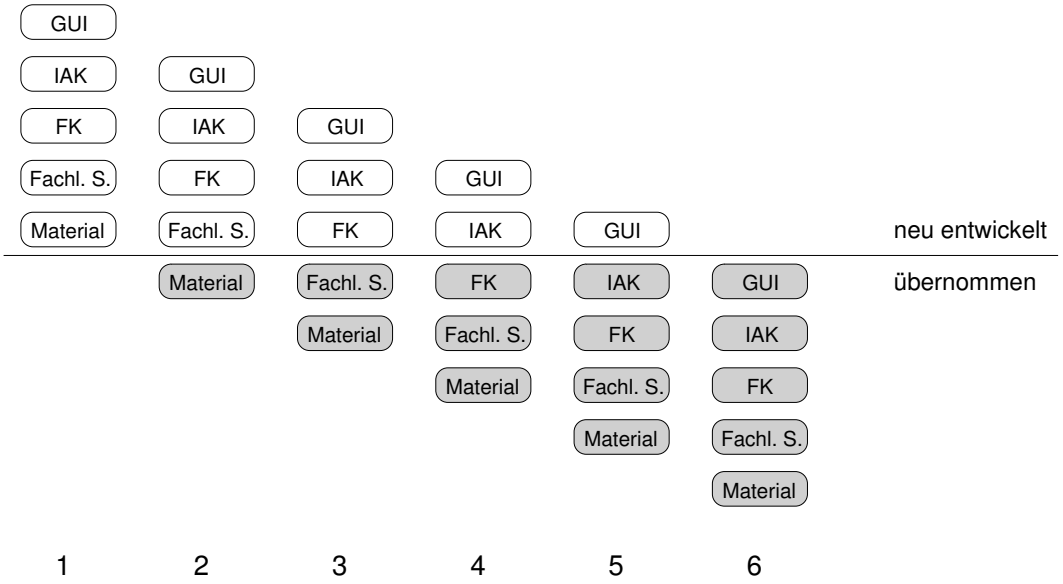


Abbildung 5.2: Übernahmemöglichkeiten von Elementen vorhandener Werkzeuge und Materialien für mobile Anwendungen

System übernommen werden, während andere angepaßt oder neu geschrieben werden müssen. Aufgrund der hierarchischen benutzt-Beziehung ist es im Normalfall nicht möglich, höher in der Hierarchie stehende Komponenten zu übernehmen, wenn sich eine Komponente darunter geändert hat. Dadurch werden sechs Fälle unterschieden (siehe Abbildung 5.2):

1. Vollkommen eigenständige Implementierung:

   Die mobile Anwendung wird von Grund auf neu entwickelt und auf die mobilen Geräte zugeschnitten. Siehe Abschnitt 5.1.2. Einige Realisierungsalternativen, die nicht auf dem WAM-Ansatz beruhen, werden in Abschnitt 5.1.8 kurz vorgestellt.

2. Beibehaltung der Materialien; Neuentwicklung von Services und Werkzeugen:

   Die Materialien, auf denen die Fachlichen Services und Werkzeuge arbeiten, werden als Grundlage übernommen. Die Fachlichen Services und die Werkzeuge werden komplett neu erstellt. Siehe Abschnitt 5.1.3.

3. Entwicklung neuer Werkzeuge:

   Die von Fachlichen Services angebotene Funktionalität ist passend. Die Werkzeuge werden für die mobile Anwendung neu entwickelt, wie in Abschnitt 5.1.4 diskutiert wird.

4. Austausch der Handhabung bei gleicher Funktion:

   Die Funktionalität der Werkzeuge ist für die mobile Anwendung geeignet. Die Handhabung und Präsentation wird neu entwickelt. Siehe Abschnitt 5.1.5.

5. Angepaßte Oberfläche für vorhandene Werkzeuge:

   Die Werkzeuge werden für die mobile Anwendung übernommen, wobei ihre Oberfläche angepaßt wird, siehe Abschnitt 5.1.6.

6. Vollständige Übernahme des Codes:

   Das gleiche System, das auch auf dem Desktop läuft, kommt – in der Regel nach kleineren Anpassungen – auf mobilen Geräten zum Einsatz. Siehe Abschnitt 5.1.7.

Vom ersten Fall, bei dem die Anwendung vollkommen neu erstellt wird, bis zum letzten, bei dem nur geringe Anpassungen erforderlich sind, nimmt der Entwicklungsaufwand ab. Außerdem nimmt die Spezialisierung der mobilen Anwendung ab, während die Leistungsfähigkeit der Geräte in der Regel zunehmen muß: Im ersten Fall kann die Anwendung auf die verwendeten Geräte von Grund auf zugeschnitten werden. Im letzten Fall werden dieselben Quelltexte verwendet, so daß keine Anpassung erfolgen kann und die mobilen Geräte fast über die Rechenleistung sowie Ein- und Ausgabemöglichkeiten von Desktop-PCs verfügen müssen.

In den folgenden Abschnitten werden die einzelnen Alternativen genauer untersucht.

5.1.2 Vollkommen eigenständige Implementierung

Eine eigenständige Implementierung wird nötig, wenn auf dem mobilen Gerät andere Materialien und Werkzeuge als im Anwendungssystem eingesetzt werden und auch nicht einzelne Teile übernommen werden können. Sie ist auch erforderlich, wenn die mobile Anwendung in einer anderen Programmiersprache als das Hauptprogramm erstellt wird.

Falls schon die Entwicklung des stationären Systems nach den WAM-Prinzipien erfolgte, können die entstandenen Szenarios, das Glossar und andere Dokumente für die Erstellung der mobilen Anwendung genutzt werden. Der Entwurf der zugrundeliegenden Materialien und Werkzeuge sollte sich zumindest in der Benennung und ihren grundsätzlichen Umgangsformen an denen der Desktopanwendung orientieren, um das Verständnis der Anwender zu erleichtern und kein zu großes Umlernen und Umdenken erforderlich zu machen.

Vorteilhaft ist bei diesem Vorgehen, daß die Anwendung mit ihren Materialien, Werkzeugen und anderen Elementen auf das mobile Gerät zugeschnitten werden kann. Nur diejenigen Umgangsformen, die für die mobile Anwendung nötig sind, müssen implementiert werden. Die Werkzeuge können von vornherein an die Beschränkungen des mobilen Gerätes angepaßt werden, und der Netzzugriff kann so gestaltet werden, daß er den Möglichkeiten entspricht.

Wichtig wird der **Datenabgleich** mit den Servern. Wenn sich die zu synchronisierenden Daten in ihren Formaten nicht genau entsprechen, ist besondere Sorgfalt von Nöten. Dem Synchronisierer (siehe Abschnitt 5.3) kommt die Rolle zu, Daten der mobilen Anwendung in die des stationären Anwendungssystems zu konvertieren und umgekehrt. Zwei Vorgehensweisen sind dabei prinzipiell möglich:

1. Der Synchronisierer wandelt die Materialien des Quellsystems in die Materialien des Zielsystems um. Die erzeugten Materialien werden dann ins Zielsystem übertragen, z. B. durch Hinzufügen zu Fachlichen Services, die diese Materialtypen verwalten.

2. Der Synchronisierer arbeitet auf einer techniknäheren Ebene. Aus den Materialien des Quellsystems werden die benötigten Daten extrahiert und direkt, ohne Umwandlung in Materialien des Zielsystems, in dessen Persistenzsysteme eingespeist.

Der erste Ansatz bietet den Vorteil, daß der Synchronisierer nur Materialien bearbeitet und die Technik hinter den Persistenzsystemen und Services nicht zu kennen braucht. Dies trägt zur Kapselung und damit Änderbarkeit und Wartbarkeit des Systems bei. Andererseits kann die Umwandlung der Materialien ein komplexerer Vorgang sein, wenn sich die Materialtypen sehr unterscheiden oder sogar in verschiedenen Programmiersprachen implementiert sind. In praktischen Projekten hat der Autor mit dieser Vorgehensweise gute Erfahrungen gemacht.

Der zweite Ansatz ist direkter, indem er mehrfache Umwandlungen vermeidet und die Daten ohne Umwege schreibt. Dafür benötigt er zusätzlich zu dem Wissen über die verwendeten Materialtypen detailliertes Wissen über die Persistenzsysteme. Diese Vorgehensweise ist daher nur bei großen Datenmengen in geschwindigkeitskritischen Bereichen sinnvoll.

5.1.3 Beibehaltung der Materialien; Neuentwicklung von Services und Werkzeugen

Bei diesem Ansatz werden in der mobilen Anwendung und im stationären Anwendungssystem die gleichen Materialien als Basis verwendet. Darauf aufbauend werden verschiedene Fachliche Services und Werkzeuge realisiert.

Die Übernahme vorhandener Materialien bietet einige Vorteile:

- Herkömmliche und mobile Werkzeuge können in einem Kontext eingesetzt werden, da sie die gleichen Materialien verwenden.

- Auch viele Fachliche Services, die auf Materialien arbeiten, können weiterhin benutzt werden. Sie müssen nicht speziell an die mobile Anwendung angepaßt werden.

- Der Austausch der Materialien über den Synchronisierer ist einfach. Sie können kopiert werden ohne Konvertierung in andere Formate.

5.1.4 Entwicklung neuer Werkzeuge

Für Geräte, die nicht leistungsfähig genug sind, die Werkzeuge der Desktopversion auszuführen, oder für die andere Werkzeuge erforderlich sind, ist dieser Ansatz gedacht.

Die Fachlichen Services werden übernommen. Wenn viel Funktionalität nicht in den Werkzeugen sondern in den Services realisiert ist, kann so einiges an Arbeit gespart werden. Außerdem ist auf diese Art sichergestellt, daß gleiche fachliche Funktionalität der mobilen und der stationären Anwendung auch gleich implementiert ist.

Diese Architektur setzt das auf S. 44 vorgestellte Multi-Channeling-Konzept um. Fachliche Services und darin gekapselte Daten stehen zentral zur Verfügung und können von verschiedenen Endgeräten genutzt werden, die über jeweils angepaßte Interaktionsschichten und Benutzungsschnittstellen verfügen.

Fachliche Services greifen in den meisten Fällen auf ein **Persistenzsystem** zu. Dieses wird für die mobile Anwendung sehr oft anders zu realisieren sein als für große Anwendungssysteme. Während stationäre Systeme ihre Massendaten in eigenständigen Datenbanken oder im Dateisystem ablegen, gibt es auf Mobilrechnern verschiedenartige Konzepte.

Bedingt durch die geringe Speicherkapazität der mobilen Geräte empfiehlt es sich, auf ihnen nur die für die tägliche Arbeit direkt benötigten Daten zu halten. Über den Synchronisierer können diese ausgewählt, auf das Gerät kopiert und später wieder in das System überspielt werden.

Für diese Daten muß ein geeignetes Austausch- und Speicherformat gefunden werden. Eine Möglichkeit ist hier der Einsatz von XML[1]. XML ist ein standardisiertes Format, das von vielerlei Geräten und Programmiersprachen verstanden wird. Auch gibt es einige für mobile Geräte geeignete XML-Parser.[2] Die ausgetauschten Dateien sind prinzipiell in menschenlesbarer Form, so daß eine Fehlersuche erleichtert wird.

Allerdings ist XML nicht immer einsetzbar, da das Einlesen und Parsen bzw. Generieren und Schreiben von XML-Dateien rechenzeitbelastend ist. Auch benötigen die XML-Bibliotheken einigen Speicherplatz und – dies ist insbesondere bei drahtloser Kommunikation der größte Nachteil – XML-Dateien sind verhältnismäßig groß.

Wie aufwendig diese Änderungen sind, hängt von der Architektur des Systems ab. Beim WAM-Ansatz gibt es das Konzept der Registratur, die eine Umstellung auf ein anderes Persistenzsystem so weit wie möglich vereinfachen soll.

[1] Die XML-Spezifikation und verwandte Standards finden sich unter `http://www.w3.org/XML/`.

[2] Siehe z. B. kXML (`http://kxml.org/`) oder TinyXml (`http://sourceforge.net/projects/tinyxml`)

Registratur und Registrator:

Eine **Registratur** ist ein Persistenz- und Kommunikationsmedium (siehe [Havenstein 00]). Die Registratur nimmt Materialien auf, die von einer Benutzergruppe gemeinsam bearbeitet werden. Materialien können hineingestellt, gesucht, ausgeliehen und zurückgestellt werden.
Die technische Realisierung der Registratur kann auf verschiedene Arten erfolgen. Beispielsweise können die Daten im Dateisystem, in Datenbanken oder in XML-Dateien gehalten werden.

Ein **Registrator** verwaltet die Registratur. Die Anwendung wickelt ihren gesamten Datentransfer über den Registrator ab, so daß die verwendete Registratur austauschbar ist. Der Registrator übernimmt fachliche Zusatzfunktionen; z. B. gibt er Fehlkarten heraus, die Anwender informieren, wenn Materialien schon entliehen sind.

Für die Entwicklung mobiler Anwendungen läßt sich dieses Konzept gut nutzen, um die zugrundeliegende Datenbank durch eine für mobile Geräte besser geeignete Implementation zu ersetzen. Im Rahmen des ersten Beispielprojekts wird auf Seite 84 beschrieben, wie der Austausch einer relationalen Datenbank durch XML-Dateien realisiert werden kann.

Falls Fachliche Services jedoch ohne Abstraktionsschicht direkt auf dem Persistenzsystem arbeiten, ist der Wechsel zu einem anderen Medium unter Umständen schwierig und fehleranfällig.

5.1.5 Austausch der Handhabung bei gleicher Funktion

Manchmal gibt es den Fall, daß die fachliche Funktionalität, die von der Funktionskomponente bereitgestellt wird, für ein Werkzeug auf dem mobilen Gerät geeignet ist, die Handhabung des Werkzeugs jedoch nicht. Wenn auch ein alleiniger Austausch der Benutzungsoberfläche, wie in Abschnitt 5.1.6 beschrieben, nicht erfolgverheißend erscheint, muß auch die Interaktionskomponente angepaßt werden. Hier können dann spezifische Interaktionen implementiert werden, unter Berücksichtigung der softwareergonomischen Erfordernisse für mobile Geräte.

Dadurch, daß Funktions- und Interaktionskomponente nur lose gekoppelt sind, kann eine Interaktionskomponente ersetzt werden, ohne daß die Implementation der Funktionskomponente davon betroffen ist.

5.1.6 Angepaßte Oberfläche für vorhandene Werkzeuge

In den meisten Fällen muß die **Bedienoberfläche** von Werkzeugen angepaßt werden. Die zur Verfügung stehende Ausgabefläche auf dem mobilen Gerät ist geringer,

die darstellbare Farbanzahl eingeschränkt. Auch die Eingabemöglichkeiten unterscheiden sich und sollten berücksichtigt werden. Die Benutzbarkeit einer Anwendung hängt wesentlich von einer guten Verwendung der Interaktionsmöglichkeiten ab.

Zum Beispiel kann das GUI auf mehrere Bildschirmseiten aufgeteilt, die Schriftgröße angepaßt und Bilder passend skaliert werden. Falls ein Touch Screen vorhanden ist, sollte damit die Eingabe möglich sein. Das Eingeben längerer Texte sollte vermieden werden. In Abschnitt 3.3 sind weitere Empfehlungen aufgeführt.

Es ist von großem Vorteil, wenn von vornherein im System geeignete Abstraktionen von der konkreten Präsentation berücksichtigt wurden. Wenn Code für Funktionalität und für die Darstellung vermischt wurde, kann der Prozeß aufwendiger sein.

Der WAM-Ansatz kennt als Lösung für dieses Problem zum einen die Unterscheidung in Funktions- und Interaktionskomponenten (siehe S. 50) und zum anderen die Trennung von Handhabung und Präsentation mittels Interaktions- und Präsentationsformen.

Interaktions- und Präsentationsformen:

Die Kopplung zwischen Präsentations- und Interaktionskomponente geschieht mittels Interaktionsformen. Diese legen nur abstrakt den Interaktionstyp fest, nicht aber dessen grafische Darstellung und Bedienung. Für letzteres sind die Präsentationsformen zuständig, die an das jeweils verwendete graphische Toolkit angepaßt sind.
Beispielsweise gibt es eine Interaktionsform ifSingleSelection, bei der genau ein Element von mehreren ausgewählt wird. Mögliche Präsentationsformen, die dies realisieren, sind Comboboxen, Radiobuttons, Listen mit einfacher Auswahl u. a.
In Tabelle 5.1 sind einige Interaktions- und Präsentationsformen aus JWAM aufgeführt.

Bei der technischen Umsetzung dieses Konzeptes in JWAM sind die Interaktionsformen als Interfaces realisiert. Die Präsentationsformen implementieren eins oder mehrere dieser Interfaces und beinhalten den Code für die Präsentation, was meist durch Erben von den vorhandenen speziellen Widget-Klassen realisiert ist. Interaktionskomponenten sprechen die Präsentationsformen nur über die Schnittstelle der entsprechenden Interaktionsformen an.

Diese Aufteilung der Zuständigkeiten und die Kapselung der Bedienoberfläche machen es möglich, die Präsentation ohne Änderungen am restlichen System zu ändern. Eine Anpassung an die Darstellungsmöglichkeiten mobiler Geräte ist hiermit einfach. Nur die Präsentationsformen müssen für jede Klasse mobiler Geräte neu implementiert werden. Mit ihnen kann dann die neue Oberfläche zusammengesetzt werden. Voraussetzung hierfür ist allerdings eine saubere Programmierung, da in den anderen Klassen keine Oberflächenkomponenten benutzt werden dürfen.

Falls sich jedoch die Bedienung der Oberfläche so sehr verändern sollte, daß ein anderes Benutzungsmodell nötig ist, kann auch die Interaktionskomponente betroffen

Interaktionsform	Präsentationsformen
ifActivator	pfJButton, pfJList, pfJMenuItem, pfJPopupMenu
ifDrag	pfJList, pfJTableOfContentsList, pfJTreeTable, pfJIcon
ifDrop	pfJList, pfJPanel, pfJTreeTable, pfJIcon, pfJButton, pfJTilingPanel, pfJToggleButton
ifFillInBoolean	pfJCheckBoxMenuItem, pfJToggleButton, pfJBooleanRadioGroup, pfJCheckBox
ifFillInDomainValue	pfJFilenameField, pfJDirectoryNameField, pfJTextArea, pfJPasswordField, pfJCheckBox, pfJTextField
ifFillInNumber	pfJProgressBar, pfJScrollingIntegerField, pfJIntegerField
ifSingleSelection	pfJList, pfJRadioGroup, pfJComboBox, pfJTabbedPaneList
ifTextDisplay	pfJDialog, pfJFrame, pfJIcon, pfJButton, pfJLabel

Tabelle 5.1: Wichtige Interaktionsformen und einige sie implementierende Präsentationsformen für Swing aus JWAM 1.7.0

sein und muß geändert werden (siehe vorherigen Abschnitt).

5.1.7 Vollständige Übernahme des Codes

Es ist für die Entwicklung am einfachsten, wenn derselbe Code auf Desktopsystemen und mobilen Geräten ausgeführt werden kann. Es sind keine zusätzlichen Elemente zu entwerfen und zu implementieren. Das System kann auf dem Desktop entwickelt, programmiert und getestet werden. Nachdem es dort läuft, kann es auf mobile Geräte übertragen werden, wo evtl. noch Anpassungen vorzunehmen sind und gerätespezifisch getestet werden muß.

Dieser ideale Fall ist in der Praxis selten zu erreichen. Gründe hierfür sind die unterschiedliche Technik und die andere Benutzbarkeit mobiler Geräte. In den vorausgegangenen Abschnitten wurden diese schon untersucht. Hier noch einmal kurzgefaßt:

- Die Prozessorleistung von mobilen Geräten reicht oft nicht aus und ihr Speicherplatz ist zu begrenzt.
- Desktopsysteme gehen meist von einer permanenten Datenverbindung aus, die auf mobilen Geräte oft nicht zu realisieren ist.
- Die Ausgabefläche ist zu klein für die Bedienoberfläche von Desktopsystemen. Daher und aus anderen Gründen (u. a. anderes Benutzungsmodell, umständli-

chere Eingabe insbesondere von Texten) ist eine 1:1-Übertragung des GUIs nicht sinnvoll.

Nichtsdestotrotz kann sich aber der Fall ergeben, daß dieser Weg dennoch erfolgversprechend ist, beispielsweise wenn die Werkzeuge klein und wenig komplex sind und während ihrer Benutzung kein Netzzugriff benötigt wird. Die mangelnde Rechengeschwindigkeit kann unter Umständen durch besonders angepaßte Betriebssysteme[3] und andere Performanzoptimierungen ausgeglichen werden. Der verfügbare Speicher kann bei einigen Geräten durch zusätzliche Speicherkarten oder Festplatten im Miniformat erhöht werden.

Für einen ersten Prototyp oder eine „proof of concept"-Implementierung muß auch das GUI nicht unbedingt angepaßt werden, sofern es auf dem kleinen Bildschirm überhaupt angezeigt werden kann.

Grundvoraussetzung für die Methode ist, daß derselbe Code auf den Geräten läuft oder zumindest für sie individuell übersetzt werden kann. Oftmals stehen nur plattformspezifische APIs zur Verfügung, die dieses verhindern. Der Einsatz von plattformübergreifenden Sprachen wie Java kann hier Abhilfe schaffen, obwohl auch dort die Portabilität gerade im Bereich der mobilen Geräte durch unterschiedliche Standards eingeschränkt ist (siehe Kapitel 6).

Zusammenfassend läßt sich sagen, daß eine vollständige Übernahme des Codes nur in wenigen Fällen auf besonders leistungsfähigen mobilen Geräten machbar und sinnvoll ist. Dort kann dieses Vorgehen für einen ersten Test der Realisierbarkeit, auf den dann Anpassungen folgen, aber nützlich sein.

5.1.8 Weitere Realisierungsmöglichkeiten

Neben den hier beschriebenen Verfahren, die sich stark am WAM-Ansatz orientieren, gibt es andere, welche die zu portierende Anwendung als Black Box betrachten und eine generische Übertragung auf mobile Geräte versuchen. Diese besitzen den Nachteil, daß spezifische Eigenschaften von WAM-Anwendungen, wie z. B. die Unterscheidung von Werkzeugen und Materialien, nicht ausgenutzt werden können. Vorteilhaft ist dagegen, daß die Anwendungen selber nicht verändert werden müssen bzw. nur ihre Präsentation angepaßt werden muß. Ein erster Prototyp kann dadurch meist schnell auf dem mobilen Gerät zum Laufen gebracht werden.

Die Grundidee ist, daß die Anwendungslogik von der Präsentation getrennt wird, ähnlich wie im „Trennung von Handhabung und Präsentation"-Muster von WAM. Dadurch kann die Oberfläche für verschiedene Geräte aufbereitet werden. Die eigentliche Anwendung wird in einigen Ansätzen mitübertragen und auf dem mobilen Gerät ausgeführt. In anderen werden nur die Oberflächendaten vom System zum mobilen Gerät und die Reaktionen des Benutzers vom mobilen Gerät zum System übertragen (siehe die Diskussion von Thin Clients auf S. 44).

[3] Ein Beispiel hierfür, ein javabasiertes Betriebssystem, das sehr effizient Java Bytecode auch auf kleinen Geräten ausführen kann, wird von SavaJe Technologies entwickelt (`http://www.savaje.com`).

Ein Ansatz stammt von Stefan Müller-Wilken. In [Müller-Wilken 02] beschreibt er, wie mit Hilfe der Techniken Reflection[4] und Accessibility[5] von Java die Oberflächenstruktur von Anwendungen extrahiert werden kann. Diese kann dann in einer in XML definierten Sprache (Java Swing Markup Language, JSML) dargestellt werden, die alle wichtigen Elemente umfaßt. Diese XML-Beschreibung kann auf verschiedene mobile Geräte übertragen werden. Für jeden Typ von mobilen Geräten sind Geräteprofile erforderlich, die aus diesen Daten wieder eine Oberfläche zusammenbauen und auf dem Gerät darstellen. Dies können beispielsweise Treiber für WAP-Browser, HTML-basierte Webbrowser oder native Anwendungen sein.

Die Benutzereingaben werden wieder zum Anwendungssystem zurückgeschickt und dort verarbeitet. Auf diese Art und Weise lassen sich Anwendungen von verschiedenen entfernten Endgeräten aus steuern. Nachteilig ist jedoch, daß dazu eine permanente Netzverbindung bestehen muß, die die Interaktionsgeschwindigkeit bestimmt und Kosten verursacht.

Ähnliche Verfahren setzen ebenfalls an der Schnittstelle zwischen Anwendungslogik und Präsentation an. Allerdings ist bei vielen von ihnen die Definition der Schnittstelle in einer abstrakten Zwischensprache nötig, da sie nicht mit Reflection arbeiten. Ein bekannteres Beispiel hierfür ist XUL, die „XML User Interface Language“, die aus dem Mozilla-Projekt entsprungen ist, siehe [XUL 01].

5.2 Singuläre mobile Anwendung

In diesem Abschnitt wird der Fall betrachtet, daß der Entwurf einer mobilen Anwendung im Mittelpunkt eines Projektes steht. Für stationäre Server und Desktop-PCs werden nur Komponenten entwickelt, sofern sie zur Unterstützung der mobilen Anwendung benötigt werden. Dies sind insbesondere Komponenten zum Zugriff auf permanente Speicher und zur Abwicklung einer Synchronisation.

Das in den letzten Abschnitten zur Konstruktion von Anwendungen Gesagte ist großenteils übertragbar. Die Entwickler sind hier jedoch freier in der Konstruktion der mobilen Anwendung, da keine Architekturvorgabe durch ein schon vorhandenes oder parallel in Entwicklung befindliches Anwendungssystem vorliegt. Daher erscheint es sinnvoll, die Mittel, die der WAM-Ansatz hierfür zur Verfügung stellt, etwas genauer zu betrachten.

Die folgenden Abschnitte sollen einen kurzen Überblick über die Konstruktion interaktiver Systeme nach dem WAM-Ansatz geben, so wie er für die bisherigen Leitbilder und Metaphern ausgearbeitet wurde. Anschließend wird geprüft, inwieweit sich das vorhandene auch auf den mobilen Kontext übertragen läßt. Einige Empfehlungen werden angeführt, die dessen Besonderheiten zu berücksichtigen.

[4] Mit dem Reflection API von Java können zur Laufzeit Informationen über Felder, Methoden und Konstruktoren geladener Klassen abgefragt werden.

[5] Das Accessibility API ermöglicht den Zugriff auf Informationen von GUI-Komponenten (z. B. angezeigter Wert, mögliche Aktionen, graphische Repräsentation, ...).

5.2.1 Konstruktion interaktiver Systeme nach WAM

Die Konstruktion erfolgt aufbauend auf Leitbildern und Metaphern (siehe Abschnitte 4.2 und 4.4) durch Konzeptions- und Entwurfsmuster.

Nach [Züllighoven 98, S. 105] sind **Konzeptionsmuster** „für Entwickler gedacht und unterstützen die *Modellierung* des Anwendungsbereichs. Sie basieren auf den für alle Beteiligten verständlichen Entwurfsmetaphern und sind auf einen begrenzten Anwendungsbereich ausgerichtet." Damit helfen Konzeptionsmuster beim Entwurf von Visionen des zu realisierenden Systems. Beispielsweise gibt es Konzeptionsmuster zum Werkzeugentwurf, zum Materialentwurf und zum Zusammenhang zwischen den beiden Klassen von Gegenständen.

Aufbauend auf den Konzeptionsmustern dienen **Entwurfsmuster** dem konstruktiven objektorientierten Entwurf. Mit ihnen lassen sich Teile größerer Softwarearchitekturen in einer gemeinsamen Sprache beschreiben. Entwurfsmuster enthalten auch einen konstruktiven Teil, der für die Implementation in einer konkreten Programmiersprache verwendet werden kann. Genaueres zum Begriff des Entwurfsmusters findet sich in [Gamma et al. 95] und [Züllighoven 98, S. 106ff]. Beispiele für Entwurfsmuster im WAM-Ansatz sind die schon beschriebene Trennung von Interaktion und Funktion (siehe S. 50) und die ebenfalls schon erläuterte Trennung von Handhabung und Präsentation (siehe S. 57).

In Abbildung 5.3 sind wichtige Konzeptions- und Entwurfsmuster des WAM-Ansatzes mit ihren Beziehungen dargestellt.

5.2.2 Berücksichtigung der Besonderheiten des mobilen Kontextes

In diesem Abschnitt wird betrachtet, inwieweit sich Konzeptions- und Entwurfsmuster für den Entwurf mobiler Anwendungen verwenden lassen, und welche anderen Dinge zu beachten sind.

Wie in den vorausgegangenen Abschnitten gezeigt wurde, lassen sich die WAM-Leitbilder und -Entwurfsmetaphern an die Gegebenheiten mobiler Anwendungen auf begrenzt leistungsfähigen mobilen Geräten anpassen und weiterverwenden.

Konzeptionsmuster des WAM-Ansatzes orientieren sich eng an den Entwurfsmetaphern. Sie sind recht abstrakt gehalten und beinhalten keine Details der inneren Konstruktion von Anwendungssystemen. Daher sind sie gut übertragbar und ein geeignetes Hilfsmittel für den Entwurf mobiler Anwendungen.

Die **Entwurfsmuster** lassen sich nicht ohne weiteres übernehmen, da sie einen konstruktiven Teil beinhalten. Dieser ist zwar unabhängig von konkreten Programmiersprachen formuliert, enthält aber Voraussetzungen über die Leistungsfähigkeit der Ablaufumgebung.

Wie der Autor in den Abschnitten 5.1.6 und 5.1.7 gezeigt hat, lassen sich bei entsprechend leistungsfähigen Geräten weite Teile von Anwendungssystemen nach dem

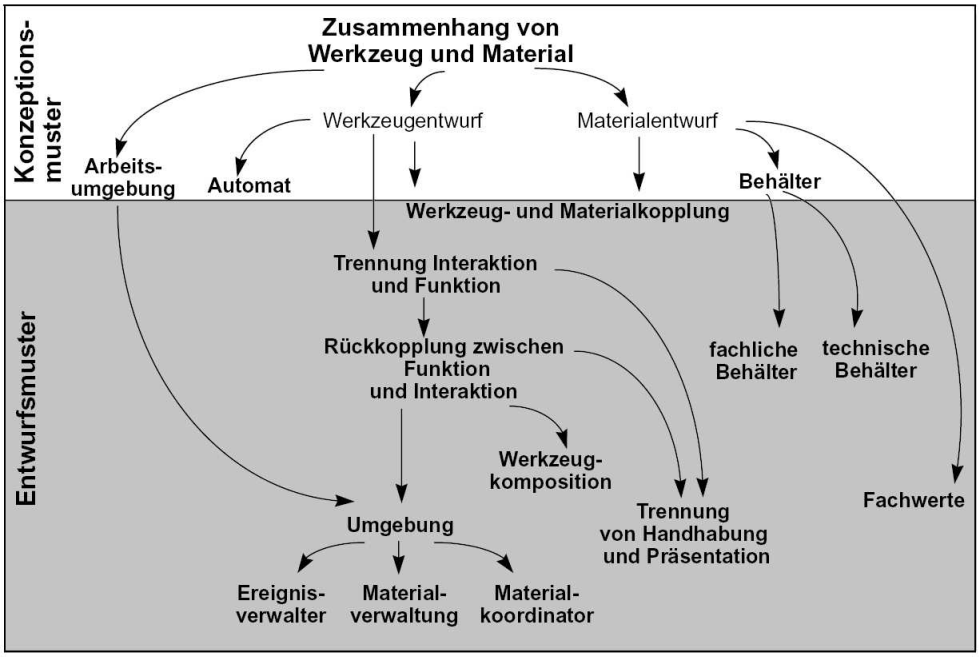


Abbildung 5.3: Hierarchie wichtiger Konzeptions- und Entwurfsmuster des WAM-Ansatzes, aus [Züllighoven 98, S. 202]

WAM-Ansatz übernehmen. Für schwächere Geräte ist dies aber nicht in diesem Maße möglich. Auf diesen Geräten können unter Umständen auch nicht alle Entwurfsmuster eingesetzt werden. Vielen Entwurfsmustern ist gemein, daß sie zur klareren Strukturierung und Gliederung des Codes beitragen, indem sie mit loser Kopplung, Schnittstellen und anderen Abstrahierungselementen arbeiten. Damit soll das Verständnis des Systems, die Fehlerfindung und Wartbarkeit erleichtert werden. Auf der anderen Seite erhöht dieses Prinzip die Anzahl der Objekte des Systems und verlangsamt die Ausführung durch Indirektionen. Eine Möglichkeit, dies zu umgehen, ist eine objektorientierte Entwicklung und ein Umsetzen durch die Kompilation in ausführbaren Code, der keine Objekte mit dynamischer Bindung u. ä. mehr verwendet und dadurch effizienter ausführbar ist.

Auf mobilen Geräten muß nun genau für den Einzelfall geprüft werden, inwieweit die Vorteile der Verwendung von Entwurfsmustern die Nachteile in bezug auf erhöhten Speicherverbrauch und langsamerer Ausführung überwiegen. Denn im Gegensatz zu stationären, leistungsfähigen Systemen fallen diese Faktoren viel mehr ins Gewicht. Gerade die Objekterzeugung ist ein aufwendiger Vorgang, der so weit wie möglich auf leistungsschwachen Geräten vermieden werden sollte.[6]

[6]Man findet viele weitere Tips zur Optimierung von mobilen Anwendungen, beispielsweise die Verwendung von String Buffer statt String-Konkatenation für J2ME, vgl. z. B. *Efficient MIDP Programming*, Forum Nokia, 2002, `http://www.microjava.com/articles/techtalk/efficient/`. Es sollte

Konkret kann dies beispielsweise den Verzicht auf die Trennung von Funktion und Interaktion bedeuten. Desktopsysteme werden zunehmend so realisiert. Dadurch, daß nur ein Objekt Funktion und Interaktion zur Verfügung stellt, entfallen einige Objekte, die nicht mehr erzeugt und verwaltet werden müssen (z. B. Benachrichtigungsnachrichten zur Realisierung der losen Kopplung zwischen FK und IAK).

Wenn kein vorhandenes Anwendungssystem die zu verwendende Programmiersprache vorgibt und die Wahl auch sonst nicht eingeschränkt ist, stellt die **Auswahl der Entwicklungssprache** eine wichtige Entscheidung dar. Hier sind mehrere Fragen zu beantworten:

- Welche Geräte werden durch die Sprache unterstützt? Eine Entwicklung in einer systemnahen Sprache bietet Performanzvorteile, ist in der Regel aber nicht für andere Geräte verwendbar. Sprachen, die mit einem Zwischencode und Virtuellen Maschinen arbeiten (siehe z. B. J2ME, das im nächsten Kapitel genauer vorgestellt wird), haben einen breiteren Einsatzbereich, sind aber prinzipbedingt langsamer in der Ausführung.
- Wie performant ist die verwendete Sprache auf den vorgesehenen Geräten? Dies betrifft auch die verwendete Ablaufumgebung (Betriebssystem, virtuelle Maschine, Speicherausbau u. a.).
- Welche Gerätemerkmale können benutzt werden? Einige Sprachen unterstützen keine Bibliotheken zum vollen Zugriff auf gerätespezifische Funktionen, z. B. Adreßdaten oder externe Speichermedien.
- Kann die Sprache auch für die Komponenten des stationären Systems eingesetzt werden? Wenn die mobile Anwendung im Vordergrund steht, ist diese Frage nicht so entscheidend wie im Fall, daß eine mobile Anwendung als Teil eines großen Anwendungssystems entwickelt wird. Für eine Synchronisierung über native Formate u. ä. ist sie aber dennoch nicht unwichtig.

Desweiteren spielen auch Lizenzkosten, einsetzbare Entwicklungsumgebungen, Verfügbarkeit von technischem Support u. a. eine Rolle.

Die Entwicklung einer mobilen Anwendung ohne entsprechendes stationäres System erleichtert ein Zuschneiden auf das mobile Gerät. Die in Kapitel 3 ausgearbeiteten **software-ergonomischen Empfehlungen** für Anwendungen auf mobilen Geräten können so von Anfang an berücksichtigt werden. Sie sind insbesondere wichtig bei ...

- ... der Festlegung der zu unterstützenden Funktionalität. Diese sollte nicht so komplex wie für stationäre Systeme sein.
- ... der Entwicklung des Benutzungsmodells der Software, so daß sie auch im mobilen Einsatz gut bedienbar ist.

jedoch immer geprüft werden, inwieweit die Lesbarkeit und Wartbarkeit darunter leiden.

- ... dem Design von Dialogen und anderen Ausgabeelementen.
- ... der möglichst optimalen Nutzung der vorhandenen Eingabemöglichkeiten.

5.3 Entwurfsmuster Synchronisierer

An einigen Stellen in der bisherigen Arbeit wurde schon deutlich, daß eine Synchronisierung von Daten zwischen mobiler Anwendung und stationärem System nötig ist. Dafür wird hier das Entwurfsmuster Synchronisierer eingeführt. Die Beschreibung folgt dabei den etablierten Gliederungspunkten Problem, Kontext, Lösung und Diskussion.

5.3.1 Problem

Immer dann, wenn Dateien nicht an einer Stelle vorgehalten werden sondern dezentral in unterschiedlichen Systemen, ist eine Synchronisierung nötig, um lokale Änderungen an allen Stellen sichtbar werden zu lassen und somit einen konsistenten Zustand sicherzustellen. Diese Synchronisierung kann entweder automatisiert erfolgen (was bei verteilten Dateisystemen Standard ist) oder mit Werkzeugen für einen expliziten Datenabgleich durch die Anwender. Letzteres findet man bei mobilen Geräten, die zur Synchronisation mit einem stationären Rechner verbunden werden.

Der folgende kurze Überblick über die umfangreiche Theorie zu dem Thema orientiert sich an einem Artikel von Balasubramaniam und Pierce, [Balas.Pierce 98]. Sie unterscheiden zwei Arten von Synchronisierern: **Dateisynchronisierer** (engl. file synchronizer) entdecken Datenänderungen auf der Basis von Dateien. **Datensynchronisierer** (engl. data synchronizer) besitzen mehr Wissen über den strukturellen Aufbau der Dateien und können ihre einzelnen Elemente abgleichen.

Der Prozeß der Synchronisierung läßt sich in zwei Stufen einteilen:

1. **Erkennung von Änderungen (engl. update detection):** Finden der Elemente, die sich seit der letzten Synchronisierung geändert haben. Bei Dateisynchronisierern sind die Elemente hierbei Dateien, bei Datensynchronisierern Felder innerhalb strukturierter Dateien, z. B. einzelne Adressen eines Adreßbuches.

   Die Implementierung dieser Suche kann auf verschiedenen Strategien beruhen. Die Extreme sind dabei der triviale Erkenner, der alle Elemente ohne Überprüfung als geändert bezeichnet, was bei wenigen Elementen und schneller Datenverbindung durchaus akzeptabel sein kann, und andererseits der exakte Erkenner, der jede Änderung genau feststellt, aber u. U. auf Kosten höherem Speicherplatzverbrauchs und längerer Berechnungszeit. In der Anwendung praktikablere Erkenner können beispielsweise mit dem Datum der letzten Änderung arbeiten. Im Zweifelsfall ist es meist günstiger, eine pessimistische Entscheidung zu treffen, also Daten, die nicht sicher als unverändert nachgewiesen werden können, als geändert zu betrachten.

2. Datenabgleich (engl. reconciliation): Verknüpfung von Änderungen zu einem neuen, synchronisierten Zustand.

Falls Elemente nicht geändert wurden, ist keine Aktion nötig. Ansonsten muß überprüft werden, ob ein Konflikt vorliegt, also Elemente an mehreren Stellen so geändert wurden, daß sie nicht automatisch in einen gemeinsamen konsistenten Zustand überführt werden können. In diesem Fall können verschiedene Strategien zur Auflösung dieses Konfliktes verwendet werden, u. a. können die Elemente unverändert gelassen und eine Fehlermeldung angezeigt werden oder dem Benutzer kann die Möglichkeit gegeben werden, eine der Änderungen zu verwerfen oder die Elemente manuell zu bearbeiten.

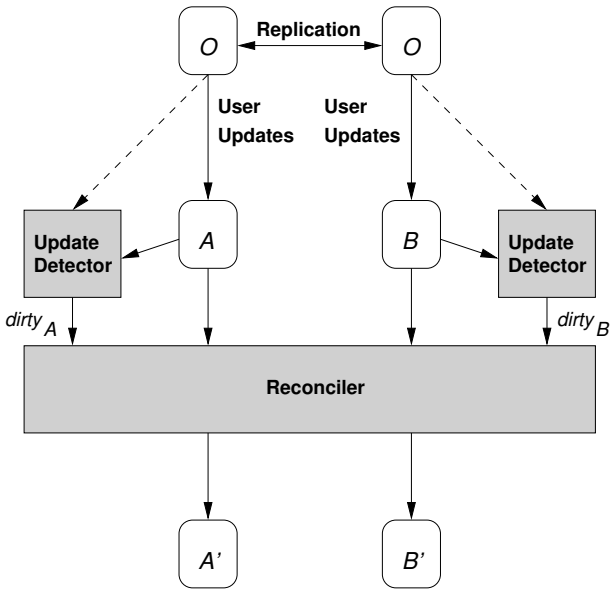


Abbildung 5.4: Formale Darstellung der Synchronisation, nach [Balas.Pierce 98]

Formal kann die Synchronisierung wie in Abbildung 5.4 dargestellt werden. Nach der Replikation beginnen beide Kopien mit demselben Zustand O. Dieser ändert sich durch Aktionen der Benutzer lokal zu den Zuständen A und B. Bei der Synchronisierung startet zunächst die Erkennung von Änderungen. Anhand des letzten gemeinsamen Zustandes O und des aktuellen Zustands A bzw. B werden die Prädikate $dirty_A$ und $dirty_B$ gebildet, die angeben, für welche Elemente lokale Änderungen vorliegen. Im folgenden Abgleich werden diese Informationen verwendet, um aus den lokalen Zuständen A und B sowie den Änderungsprädikaten $dirty_A$ und $dirty_B$ die neuen Zustände A' und B' zu berechnen. Falls keine Konflikte vorliegen, sind diese identisch.

5.3.2 Kontext

Als Beispiele für Synchronisierer bei mobilen Geräten werden im folgenden HotSync von Palm und ActiveSync von Microsoft betrachtet.

Palm HotSync

PDAs mit PalmOS können seriell über eine Dockingstation, über Infrarot oder auf mehrere andere Arten mit einem stationären PC verbunden werden. Dies dient im wesentlichen drei Zwecken: Der Synchronisation von Daten des Handhelds und des Desktoprechners, der Datensicherung der mobil erfaßten Daten und der Installation neuer Anwendungen und ihrer Datenbanken auf dem Handheld. Palm hat hierfür den Begriff HotSync geprägt. Ein PDA wird dabei über einen eindeutigen Benutzernamen identifiziert. Es kann für die Standard-PIM-Applikationen festgelegt werden, ob Daten abgeglichen werden oder ob entweder PC- oder Handheld-Daten die jeweils anderen überschreiben. Falls ein Konflikt beim Abgleich gefunden wird, bleibt das geänderte Element in beiden Versionen erhalten. Es wird eine Protokollnachricht erzeugt und der Benutzer sollte den Konflikt manuell auflösen, z. B. durch Löschen eines Elementes.

Die Synchronisation läuft folgendermaßen ab (vgl. Abbildung 5.5): Auf dem PC wartet der Hotsync Manager im Hintergrund auf eine Verbindung. Diese erfolgt auf Knopfdruck an der Dockingstation oder über Software gesteuert. Auf dem mobilen Gerät startet daraufhin der HotSync Client und weckt den HotSync Manager. Diese beiden tauschen sich über die SyncManager API aus, die technische Einzelheiten der Verbindung kapselt.

Der HotSync Manager ruft nacheinander alle angemeldeten **Conduits** auf. Das sind Programme, welche die Daten einzelner Applikationen synchronisieren; z. B. ist der Datebook Conduit dafür zuständig, die Termine, die im Kalender gespeichert sind, abzugleichen. Programmierer von Conduits können auf das Conduit Development Kit zurückgreifen, um eigene Conduits in C, C++ und Java zu erstellen (zur Vertiefung siehe [RhodesMcKeehan 01, Part III: Designing Conduits]). Falls nur die Sicherung von Daten erforderlich ist, kann der vorhandene Backup Conduit benutzt werden, der automatisch die Anwendungsdatenbanken des mobilen Geräts auf den PC kopiert, jedoch keinen Datenabgleich vornimmt. Conduits sollten so schnell wie möglich ablaufen und keine Benutzereingaben erfordern.

Wenn ein Conduit Programmdaten auf dem stationären System verändert, wird die betreffende Anwendung durch Notifier informiert, damit keine Konflikte oder Anomalien bei einem unbeabsichtigten gleichzeitigen Zugriff auf die Daten auftreten können. Notifier werden wie Conduits im Regelfall vom Programmierer der Applikation erstellt.

Microsoft ActiveSync

Wie bei Palms HotSync erfolgt bei Microsoft ActiveSync (bis zur Version 3.0 Windows CE Services genannt) der Abgleich mit einem stationären Computer. Bei einer Synchronisierung können folgende Vorgänge ausgeführt werden:

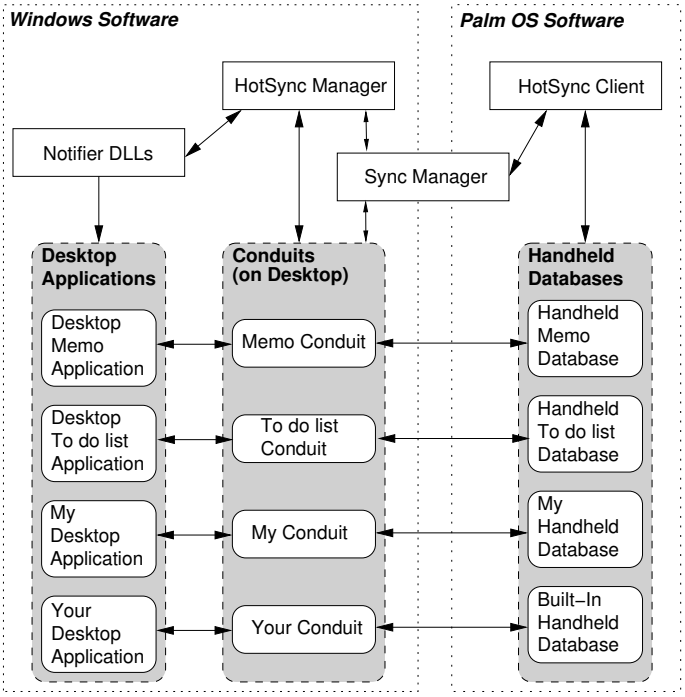


Abbildung 5.5: Struktur der Synchronisation unter PalmOS HotSync (nach: PalmSource, Inc: Introduction to Conduit Development, 2002, `http://www.palmos.com/dev/support/docs/conduits/`)

- Daten werden synchronisiert. Die Erkennung von Änderungen erfolgt aufgrund von Zeitstempeln und Benutzereinstellungen.
- Dateien können ausgetauscht werden. Da Handheld und Desktopsystem oft verschiedene Datenformate verwenden, findet eine automatische Konvertierung statt.
- Datenbanktabellen werden im- und exportiert.

Im Gegensatz zu HotSync erfolgt der Datenabgleich automatisch, solange die Geräte verbunden sind (diese Voreinstellung kann in den Optionen geändert werden). Änderungen auf einem System werden in das andere sofort übertragen. Ebenfalls in den Einstellungen kann das Verhalten bei Konflikten geändert werden. Möglich sind Präferenzen für Desktop-PC oder Handheld oder ein manuelles Auflösen.

Es ist möglich, eine sogenannte Partnerschaft festzulegen. Dabei wird dem mobilen Gerät ein eindeutiger Name zugeordnet, über den es beim nächsten Anschluß an denselben PC automatisch identifiziert wird. Individuelle Einstellungen zur Synchronisation, die einmal getroffen wurden, bleiben so erhalten. Ein mobiles Gerät kann mit bis zu zwei Geräten Partnerschaften eingehen (z. B. mit dem privaten PC und dem im Büro); ein PC kann beliebig viele mobile Geräte verwalten.

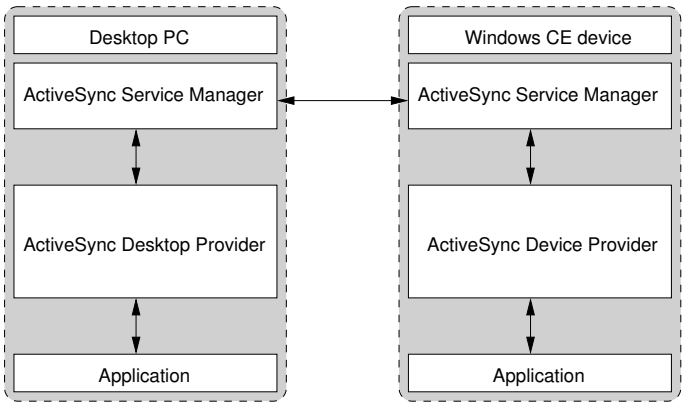


Abbildung 5.6: Struktur der Synchronisation unter MS ActiveSync (nach: MSDN: Microsoft Windows CE.net, ActiveSync, `http://msdn.microsoft.com/library/default.asp?url=/library/en-us/wceactsy/htm/ceoriactivesync.asp`)

ActiveSync ist wie folgt aufgebaut (vgl. Abbildung 5.6):

Der mit ActiveSync mitinstallierte Service Manager ist für die allgemeinen Synchronisationsaufgaben verantwortlich. Er kann Verbindungen aufbauen, Konflikte auflösen und Daten austauschen. Er besteht aus einem Teil, der auf dem Desktop-PC läuft, und einem, der auf dem mobilen Gerät installiert ist. Service Provider sind für die Ausführung der Synchronisierung einer Applikation zuständig. Sie werden vom Service Manager aufgerufen und bestehen ebenfalls aus zwei Teilen: dem Desktop Provider,

der auf dem PC läuft, und dem Device Provider auf dem mobilen Gerät. Microsoft liefert mit ActiveSync Service Provider für diverse Standardapplikationen aus.

5.3.3 Lösung

Es erscheint sinnvoll, den für mobile Geräte sehr wichtigen Prozeß der Synchronisierung mit einer Entwurfsmetapher zu unterstützen, so daß er einen hohen Stellenwert erhält und leicht verständlich ist. Doch gibt es in der nicht-technischen Alltagswelt keinen passenden Vergleich.

Microsoft hat in Windows 95 versucht, mit dem Aktenkoffer (engl. briefcase) eine Metapher zur Synchronisation von Dokumenten zu etablieren. Joel Spolsky schreibt dazu in [Spolsky 01]: „A metaphor, badly chosen, is worse than no metaphor at all. Remember the briefcase from Windows 95? [...] It was supposed to be a ‚briefcase', where you put files to take home. But when you took the files home, you still had to put them on a floppy disk. So, do you put them in the briefcase or on a floppy disk?"

Ein weiterer Grund, der gegen eine Entwurfsmetapher Synchronisierer spricht, ist, daß sie nicht orthogonal zu den anderen Entwurfsmetaphern wäre. Während Werkzeuge, Materialien, Automaten, Arbeitsumgebung, Fachliche Services u. a. unabhängig voneinander sind, lassen sich Synchronisierer gut als Automaten, Werkzeuge oder Fachliche Services realisieren.

Das Konzept des Synchronisierers ist jedoch nicht nur als Entwurfsmuster für Entwickler sinnvoll. Anwender müssen Grundlagen der Synchronisation verstehen, um in der mobilen Anwendung damit umzugehen. Es wird im Regelfall nicht möglich sein, eine Synchronisierung unsichtbar für Anwender zu integrieren. Diese müssen Besonderheiten wie einen evtl. mobil nicht vollständig zugreifbaren Datenbestand, entfernt aktualisierte Daten und daraus folgende Konflikte bei der Synchronisierung kennen, um die mobile Anwendung bedienen zu können. Daher ist für sie ein Verständnis des Musters Synchronisierer ebenfalls nützlich.

Unter Berücksichtigung dieser Punkte erfolgt folgende Definition:

Definition Entwurfsmuster Synchronisierer:

Im WAM-Kontext sind Synchronisierer (engl. synchronizer) spezielle Automaten, Werkzeuge oder Fachliche Services, die Abweichungen von bestimmten im System gehaltenen und lokal auf einem mobilen Gerät vorhandenen Daten feststellen und die Unterschiede abgleichen können.

Synchronisierer verfügen über zwei Hauptoperationen:

- Kopieren der mobil benötigten Daten eines Anwendungssystems auf ein mobiles Gerät
- Kopieren und Abgleichen der Daten eines mobilen Gerätes mit einem Anwendungssystem mit automatischem oder manuellem Auflösen von Konflikten.

Für ihre technische Realisierung können z. B. die oben vorgestellten Techniken Palm HotSync und MS ActiveSync in Automaten, Werkzeuge oder Fachliche Services eingebunden werden. Eine Alternative ist die Verwendung von SyncML, einer auf XML basierenden Sprache zur Synchronisation verschiedener Geräte und Anwendungen über beliebige Netzwerke.[7]

Anmerkung: Im JWAM-Rahmenwerk in der Version 1.8.0 ist schon eine Klasse Synchronizer im Package de.jwam.technology.util vorhanden. Sie ist dafür zuständig, Prozesse *innerhalb* des Rahmenwerkes zu synchronisieren und Multithreading zu vermeiden. Sie hat mit dem hier vorgestellten Konzept nichts zu tun.

5.3.4 Diskussion

Beim Einsatz des Entwurfsmusters Synchronisierer sind hauptsächlich zwei Fragen zu klären:

1. Welche Daten müssen auf das mobile Gerät kopiert werden?

2. Wie kann mit einem konkurrierenden Zugriff auf Materialien umgegangen werden?

Auswahl der zu kopierenden Daten

Im Normalfall liegen im stationären Anwendungssystem weit mehr Daten vor, als für die mobile Arbeit benötigt werden und auf dem mobilen Gerät Platz finden. Daher stellt sich die Frage, welche Daten zu kopieren sind. Auch wenn alle kopiert werden könnten, macht es Sinn, eine Auswahl zu treffen, da in der Regel die Selektion benötigter Dateien und ihre Übertragung auf das mobile Gerät schneller ist als ein bedingungsloses Kopieren aller Dateien. Außerdem können so Probleme mit konkurrierendem Zugriff minimiert werden, vgl. die Diskussion im nächsten Abschnitt.

Die Entscheidung über den Umfang der Daten hängt in der Praxis davon ab, ob eine online-Verbindung aufgebaut werden kann, über die ursprünglich nicht vorgesehene Daten nachträglich geladen werden können. In diesem Fall müssen Daten, die zwar potentiell mobil benötigt werden könnten, dies aber unwahrscheinlich ist, nicht unbedingt vorher auf das Gerät übertragen werden.

Wenn mehr Daten benötigt werden, als Platz zur Verfügung steht, können sie vielleicht komprimiert werden, indem nicht benötigte Informationen weggelassen werden. Beispielsweise werden statt vollständiger Kundendaten nur Name, Adresse und bezogene Produkte und Dienstleistungen übertragen. Der Synchronisierer ist dann auch dafür zuständig, die benötigten Informationen herauszuziehen und nach der mobilen Arbeit im Anwendungssystem wieder zu konsistenten Materialien zusammenzustellen.

[7] Die SyncML-Spezifikation und verwandte Dokumente sind im World Wide Web unter `http://www.openmobilealliance.org/syncml/` zu finden.

Welche Daten für die mobile Anwendung benötigt werden, ist stark von der Anwendungsdomäne und der Arbeitsorganisation abhängig, so daß dafür hier keine allgemeinen Vorgaben gemacht werden können. Oftmals findet man eine Gliederung in Stammdaten, die immer benötigt werden, und Auftragsdaten, von denen nur aktuelle interessant sind.

Umgang mit konkurrierendem Zugriff auf Materialien

Ebenfalls wichtig bei der Benutzung von Synchronisierern ist die Frage, wie mit Konflikten umgegangen wird. Diese treten auf, wenn mehrere Benutzer Änderungen an denselben Elementen vorgenommen haben, so daß sie sich nicht in einen gemeinsamen konsistenten Zustand überführen lassen. In WAM-Systemen kann dies bei paralleler Bearbeitung von Materialien auftreten.

Zu beachten ist ggf. auch der Spezialfall, daß sich Materialien zu einem Zeitpunkt nur an genau einem Ort befinden dürfen. Dies ist bei Gegenständen der Realität der Normalfall. Bei der Abbildung in Anwendungssysteme kann es sinnvoll sein, das ebenfalls so zu modellieren.

Für den Einsatz mobiler Geräte und ihre Anbindung über Synchronisierer müssen diese Möglichkeiten berücksichtigt werden. Wenn im Anwendungssystem schon eine Komponente vorhanden ist, die für die Verwaltung von Materialien zuständig ist – wie z. B. eine Registratur (siehe Abschnitt 5.1.4) oder ein Fachlicher Service (siehe Abschnitt 4.5.2) –, dann kann diese benutzt werden. Dies bietet den Vorteil, daß 1. keine neue Komponente aufwendig entwickelt werden muß und 2. ein anwendungssystemweit einheitliches Modell in der Regel verständlicher und besser wartbar ist.

In einigen Fällen wird aber eine eigene Komponente nötig sein, da entweder keine Materialverwaltung im System vorhanden oder diese für die Anbindung der mobilen Geräte nicht geeignet ist. Dann kann es sinnvoll sein, auf Konzepte zurückzugreifen, die im WAM-Kontext schon untersucht wurden.

Für die softwaretechnische Umsetzung einer Registratur wurde das Original/Kopie-Modell entwickelt (vgl. [Havenstein 00, S. 7ff]). Dort werden drei Fälle unterschieden:

1. Exklusiver Zugriff auf das Material; es gibt keine Kopien.

   Das Material darf nur von einem Benutzer zur Zeit bearbeitet werden. Die anderen dürfen es nicht einsehen.

   Für den mobilen Kontext bedeutet dies, daß das Material bei der Synchronisierung beim Kopieren auf das mobile Gerät im System gelöscht oder gesperrt wird, bis es wieder vom Gerät zurückübertragen wird. Da das Material immer vollständig ersetzt wird, können keine Konflikte auftreten.

   Dieser Fall tritt selten auf, da die Bearbeitungsdauer von Materialien auf mobilen Geräten im Regelfall sehr viel größer ist als normalerweise im Anwendungssystem mit Werkzeugen. In vielen dem Autor bekannten Projekten findet die Synchronisierung beispielsweise täglich statt. Das Modell eignet sich jedoch gut, wenn

Materialien abwechselnd im Anwendungssystem und auf dem mobilen Gerät bearbeitet werden, z. B. durch dieselbe Person im Büro und unterwegs.

2. Exklusiver Zugriff auf ein Original; Kopien sind möglich.

   Nur ein Benutzer darf das Material bearbeiten. Andere Benutzer dürfen jedoch Kopien für einen lesenden Blick auf das Material anfertigen.

   Auch hier können keine Konflikte auftreten. Das Modell ist für Materialien sinnvoll, die in der mobilen Anwendung nur lesend benutzt und nicht verändert werden. Das Original verbleibt im System und kann dort verändert werden. Jedesmal bei der Synchronisierung wird das Material im aktuellen Stand auf das mobile Gerät übertragen und steht dort zur Verfügung.

   Ein anderer Fall ergibt sich, wenn das Original auf einem mobilen Gerät zur Verfügung steht und im System eine Kopie verbleibt. Dies bietet sich an, wenn das Material nur bei der Durchführung der mobilen Arbeitsaufgabe verändert wird und nicht bei der stationären Arbeit. Bei letzterer steht eine Kopie des Materials als Dokumentation des letzten Standes zur Verfügung.

3. Zugriff immer nur auf Kopien möglich.

   Das Original wird zentral verwaltet. Benutzer erhalten Kopien, mit denen sie arbeiten können. Über Zeitstempel o. ä. kann festgestellt werden, ob ein Zurückstellen möglich ist.

   Wenn eine andere Kopie des Materials vorher zurückgestellt wurde, tritt ein Konflikt auf, der dann aufzulösen ist. Dieses Modell erscheint sinnvoll, wenn auch im Anwendungssystem mehrere Benutzer mit dem Material umgehen und dabei nur selten schreibend darauf zugreifen müssen.

Die Wahl einer konkreten Strategie sollte immer mit Blick auf das gesamte Anwendungssystem erfolgen. Vgl. hierzu die Diskussion von Multi-Channeling in Abschnitt 4.5.

5.4 Zusammenfassung

Wie begründet wurde, können neben den grundlegenden Metaphern des WAM-Ansatzes auch darauf aufbauende Konzeptionsmuster zur Modellierung des Anwendungsbereiches für mobile Anwendungen verwendet werden. Von der Leistungsfähigkeit der Hardware und der Verfügbarkeit von Softwareplattformen hängt ab, inwieweit bestehende Entwurfsmuster zur technischen Umsetzung eingesetzt und sogar Teile des Codes von stationärem und mobilem System gemeinsam entwickelt werden können. Dazu wurden verschiedene Möglichkeiten vorgestellt, die sich an der typischen Architektur von WAM-Systemen orientieren.

In dem Fall, in dem eine mobile Anwendung ohne bestehendes stationäres Anwendungssystem erstellt werden soll, sind die Entwickler freier in ihren Gestaltungsmöglichkeiten. Auch dazu wurden einige Empfehlungen gegeben.

Das neu eingeführte Entwurfsmuster Synchronisierer bietet Hilfestellungen für die Konzeption und Realisierung des Datenaustausches zwischen stationärem System und mobiler Anwendung. Der theoretische Hintergrund wurde erläutert, verschiedene technische Realisierungen bei PDAs wurden vorgestellt und sich ergebende Fragen zur Auswahl der zu kopierenden Daten und zum konkurrierenden Zugriff auf Materialien wurden diskutiert.

Kapitel 6

J2ME: Java für mobile Geräte

In diesem Kapitel wird die im folgenden zur Implementation eingesetzte Programmierplattform J2ME beschrieben. Zuerst wird der große Zusammenhang dargestellt und J2ME in die Java-Technologie eingeordnet. Das zu ihrer (Weiter-)Entwicklung eingesetzte Verfahren, der Java Community Process und ihr Aufbau mit Konfigurationen, Profilen und Optional Packages wird beschrieben. Dem Profil MIDP und den beiden optionalen Packages für PDAs widmen sich aufgrund ihrer Bedeutung spezielle Abschnitte ausführlich. Abschließend folgt eine kurze Bewertung der Technik unter dem Blickwinkel der Eignung für Anwendungen nach dem WAM-Ansatz.

6.1 Die Java-Familie

Die Java 2 Micro Edition (J2ME) ist ein Teil der Java-Produktfamilie der Firma Sun Microsystems (Sun). Die erste Java-Version wurde 1995 veröffentlicht. Seitdem ist der Einsatzbereich der Sprache sehr gewachsen. Zur Wahrung der Übersichtlichkeit hat Sun eine Aufteilung in drei Ausgaben (engl. editions) vorgenommen, siehe Abbildung 6.1:

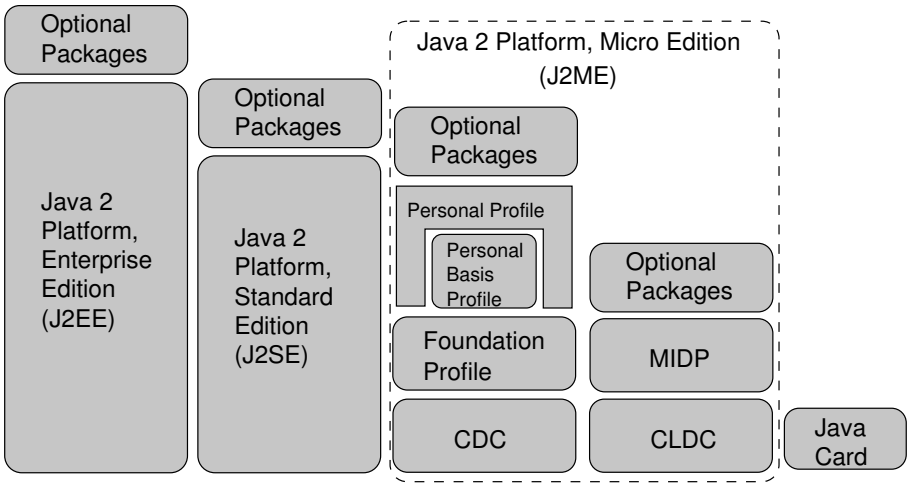


Abbildung 6.1: Die drei Java-Plattformen

Java 2 Standard Edition (J2SE): Die J2SE entstand direkt aus den ersten Versionen der Programmiersprache Java und ihren Bibliotheken. Sie ist hauptsächlich für Desktop-Computer und Workstations gedacht. Insbesondere sind Klassen für die Client-Seite von Geschäftsanwendungen enthalten.

Java 2 Enterprise Edition (J2EE): Mit der J2EE können verteilte und serverbasierte Anwendungen entwickelt werden. Eingesetzte Techniken dafür sind u. a. Enterprise JavaBeans (EJB), Servlets und Java Server Pages (JSP).

Java 2 Micro Edition (J2ME): Die J2ME wurde für Geräte mit begrenztem Speicher, beschränkten Ein- und Ausgabemöglichkeiten und geringer Verarbeitungsgeschwindigkeit entworfen.

Jede dieser Plattformen besteht aus der Spezifikation einer virtuellen Maschine für die Sprache, einem Satz von Bibliotheken und zugehörigen Softwarewerkzeugen.

Die Standards für die Java 2 Plattform entstehen im Java Community Process (JCP). Daher ist ein grundlegendes Verständnis dessen Zwecks, Vorgehens und der in ihm erstellten Dokumente wichtig, um die Entwicklung der Standards verstehen und ihre Verfügbarkeit und Reife abschätzen zu können. Dies ist insbesondere für die Java 2 Micro Edition bedeutend, da sie eine junge Technologie ist und viele Spezifikationen gerade erst im Entstehen begriffen sind oder vor kurzem veröffentlicht wurden. Im Anhang A findet sich eine Zusammenfassung des JCPs.

6.2 Grundlegender Aufbau der J2ME

Durch die Vielfalt an Zielgeräten ist die J2ME weit unterteilt. Sie besteht aus vielen Standards, die eine große Bandbreite an Geräten mit unterschiedlicher Leistungsfähigkeit abdecken. Diese Standards sind in drei Kategorien eingeordnet: Konfigurationen (engl. configurations), Profile (profiles) und optionale Pakete (optional packages), siehe Abbildung 6.2.

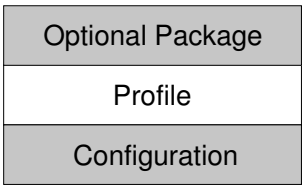


Abbildung 6.2: Kategorien der J2ME

Konfigurationen bilden die Grundlage für viele Geräte mit gemeinsamen Eigenschaften. In ihnen werden zulässige Java Sprachelemente, Kernbibliotheken und die Spezifikation von passenden virtuellen Maschinen definiert. Alle Elemente sind recht

allgemein gehalten und nicht spezialisiert, z. B. finden sich keine APIs für Bedienschnittstellen oder konkrete Netzwerkprotokolle, die nur auf wenigen Geräten einsetzbar wären. Für die J2ME gibt es die beiden Konfigurationen CDC und CLDC, die im nächsten Abschnitt beschrieben werden.

Ein **Profil** erweitert eine Konfiguration für speziellere Geräteklassen für einen bestimmten Einsatzzweck. Es ist daher vertikal ausgeprägt. Ein Profil setzt immer auf genau einer Konfiguration auf und kann andere Profile als Basis benutzen. Es existieren schon eine Reihe von Profilen. In den Abschnitten 6.4 und 6.6 werden die für das Thema dieser Arbeit wichtigsten vorgestellt.

Optional Packages stellen spezialisierte Funktionen für eingeschränktere Aufgabenbereiche zur Verfügung. Es handelt sich um Bibliotheken, die zusätzlich zu denen des Profils und der Konfiguration benutzt werden können. Ein Optional Package kann potentiell mit mehreren Profilen benutzt werden.

6.3 Konfigurationen

Die Basis von J2ME bilden zwei Konfigurationen:

Die **Connected Device Configuration (CDC)** ist für Geräte mit ausreichender Prozessorleistung und guter, möglicherweise permanenter Netzanbindung gedacht. Implementierungen auf Geräten müssen über mindestens 512 KByte für die Java Umgebung und mindestens 256 KByte als Runtime Memory verfügen. In diese Klasse fallen z. B. Set-Top Boxen für Fernseher, Bildtelefone und leistungsstärkere PDAs.

CDC beinhaltet eine minimale Auswahl von APIs der J2SE, die eine sinnvolle Verwendung zulassen. Dabei wurden als deprecated gekennzeichnete Methoden entfernt. Als virtuelle Maschine gibt es von Sun Microsystems die CVM, die der Java Virtual Machine für J2SE ähnelt. Sie ist auf weitgehende Portabilität ausgerichtet. Klassen können nicht nur dynamisch geladen sondern auch schon vorgeladen benutzt werden, sich so z. B. im ROM eines Gerätes befinden.

CDC liegt seit März 2001 als Final Release vor. Es bildet die Grundlage für mehrere Profile. Einige davon werden im nächsten Abschnitt vorgestellt.

Im Gegensatz zu CDC ist **CLDC**, die **Connected Limited Device Configuration** für leistungsschwache, typischerweise batteriebetriebene Geräte mit eingeschränktem Netzwerkzugriff gedacht. Geräte müssen über mindestens 128 KByte für die Java Umgebung und mindestens 32 KByte als Runtime Memory verfügen. Hierfür typische Geräteklassen sind Mobiltelefone, PDAs und Smartphones.

Wichtig sind die Sprachunterschiede im Gegensatz zur originalen Spezifikation für Java. So werden in CLDC 1.0 keine Gleitkommazahlen unterstützt, was hauptsächlich damit begründet wird, daß die meisten Zielgeräte keine Gleitkommaeinheiten besitzen und eine Emulation in Software viel Leistung benötigt. Es steht keine Finalization zur Verfügung, d. h. man kann keine Methode deklarieren, die bei Vernichtung eines Objektes aufgerufen wird. Außerdem wurde die Fehlerbehandlung um einige Exceptions gestrafft. Eine Virtuelle Maschine für CLDC muß zusätzlich keine nativen Methoden

über das Java Native Interface, benutzerdefinierte Class Loader oder Reflection unterstützen.

Seit Mai 2000 gibt es CLDC 1.0 als Final Release. Mittlerweile wurde auch eine Nachfolge-Spezifikation verabschiedet. CLDC 1.1 existiert seit März 2003 als Final Release. Es ist vollständig rückwärtskompatibel zur Version 1.0. Als Hauptänderung ist darin wieder Unterstützung für Gleitkommazahlen vorhanden. Dies führt zur Ergänzung von zwei Klassen für Gleitkommazahlen unterschiedlicher Genauigkeit und mehrerer Methoden im API. Ansonsten wurde das API an einigen Stellen überarbeitet.

Für CLDC stehen viele Virtuelle Maschinen zur Verfügung. Von Sun Microsystems stammen die Referenzimplementierung, genannt KVM, sowie eine virtuelle Maschine mit Just-in-time Compiler, die im Projekt Monty entwickelte Hotspot Virtual Machine. Von anderen Herstellern gibt es eigene Entwicklungen für eine Vielzahl an Geräten. Sie unterscheiden sich u. a. im benötigten Speicherplatz und der Ausführungsgeschwindigkeit.

Zum heutigen Zeitpunkt gibt es ein Profil für CLDC, MIDP, das aufgrund seiner Wichtigkeit für diese Arbeit weiter unten in einem eigenen Abschnitt vorgestellt wird.

CDC enthält ein komplettes CLDC. Daher sind Programme für CLDC auch auf CDC (und damit potentiell leistungsfähigeren Geräten) lauffähig.

6.4 Profile für CDC

Es gibt eine Reihe von Profilen, die auf Basis der CDC implementiert wurden. Hier werden kurz die bedeutendsten vorgestellt:

Foundation Profile: Das Foundation API stellt – der Name sagt es schon – eine Grundlage dar, die die zugrundeliegende Konfiguration erweitert und auf der andere Profile aufsetzen können. Es ist daher breit angelegt. Es beinhaltet eine Teilmenge des J2SE APIs. GUI-Elemente sind nicht enthalten. Anwendungen, die keine Bedienoberfläche benötigen, können auch direkt auf dem Foundation Profile aufsetzen. Das Final Release datiert vom März 2001.

Personal Basis Profile: Dieses Profil ist zwischen Foundation- und Personal Profile angesiedelt. Es enthält schon grundlegende GUI-Klassen auf Basis des AWTs und ist somit für Anwendungen gedacht, die mit leichtgewichtigen AWT-Komponenten auskommen. Im Juni 2002 erschien das Final Release.

Personal Profile: Der Nachfolger der Personal Java Umgebung. Gedacht für Anwendungen, die eine leistungsfähige und zuverlässige Internetanbindung und volle Unterstützung für Java Applets und ausgefeilte AWT Bedienoberflächen nach dem JDK 1.1 Standard benötigen. Seit September 2002 liegt das Final Release vor.

Mit dem Personal Profile können klassische ausführbare Programme mit main-Methode, Applets und sogenannte Xlets definiert werden, ausführbare Programme mit einem festen Lebenszyklus, der von der aufrufenden Umgebung gesteuert werden kann. Im Gegensatz zu Applets benötigen Xlets keine einbettende Webseite, sind limitiert in der Interaktion mit ihrem Kontext und können auch pausiert werden.

Game Profile: Entwickelt werden 2D- und 3D-Funktionen, Möglichkeiten zum direkteren Hardwarezugriff, ein Sound API u. ä. für Kernfunktionalitäten von Spieleplattformen. Der JCP Prozeß besteht seit Juni 2001. Bisher ist noch kein Community Draft verfügbar.

6.5 Optional Packages

Wie oben beschrieben, können Profilen mit Optional Packages weitere Funktionalitäten hinzugefügt werden. Wichtige schon definierte Profile sind:

Wireless Messaging API: Das hauptsächlich für Mobiltelefone gedachte Wireless Messaging API (WMA) ermöglicht den Empfang und den Versand von Nachrichten. Dieses können Textnachrichten oder binäre Nachrichten sein. Die zugrundeliegende Technik baut u. a. auf dem GSM-SMS System für Kurznachrichten auf.

Das Optional Package ist mit CDC und CLDC einsetzbar.

Mobile Media API: Dieses Optional Package dient dem einfachen Zugriff auf Audio- und Videoströme und deren Kontrolle. Die Spezifikation legt keine konkreten Sprach- und Videoformate fest sondern nur deren Einbindung. Über Manager können Player und DataSources erzeugt werden. Player steuern über Controls die Wiedergabe von Multimediaströmen. Die dazu benötigten Daten stammen aus den DataSources, die sich u. a. auch um die Protokollabwicklung und Synchronisation kümmern.

Es gibt von Sun eine Referenzimplementation des Mobile Media APIs (MMAPI) für CLDC/MIDP. Sie ermöglicht u. a. einfache Tongenerierung und die Verwendung von MIDI- und Wave-Audiodateien sowie MPEG-1 Video.

Bluetooth API: Mit dem Bluetooth API kann diese Technologie für Kurzstreckenfunk auf den mobilen Geräten zum einfachen kabellosen Datenaustausch mit der Umgebung genutzt werden.

Die Referenzimplementation von Motorola setzt mindestens die Konfiguration CLDC voraus.

RMI Optional Package: Diese Erweiterung macht den Zugriff auf entfernt angebotene Dienste mittels der Java Remote Method Invocation von J2SE auch für mobile Geräte möglich. Dadurch kann die Kommunikation zwischen zwei Java-Anwendungen auf einer höheren Ebene realisiert werden.

Die Referenzimplementation von Sun baut auf CDC / Foundation Profile auf.

Kurz vor dem Erscheinen der Final Release stehen zwei Optional Packages für PDAs, die aufgrund ihrer Bedeutung für diese Arbeit in Abschnitt 6.7 genauer vorgestellt werden.

6.6 MIDP – Java für Mobile Information Devices

Das Mobile Information Device Profile (MIDP) ist gedacht für kleine, batteriebetriebene, eingeschränkte Geräte mit drahtloser Internetanbindung. Beispiele hierfür sind Mobiltelefone, PDAs, Smartphones und hochentwickelte Funkrufsysteme. Es ist in der Version 1.0 seit September 2000 verfügbar und wurde seitdem auf einer Vielzahl an Geräten implementiert, mit steigender Tendenz.[1] Mittlerweile ist die nächste Version der Spezifikation erschienen: MIDP 2.0 (Next Generation). Diese liegt seit November 2002 als Final Release vor.

Kennzeichnend für MIDP sind die geringen Hardwareanforderungen, die eine effiziente Ausführung auch auf leistungsschwachen Geräten ermöglichen. Dementsprechend eingeschränkt sind die Möglichkeiten. Trotzdem ist alles vorhanden, um vollständige Anwendungen für kleine mobile Geräte zu erstellen.

Zentrale Elemente von MIDP sind MIDlets als ausführbare Programme. Sie verfügen über einen Lebenszyklus, der von einem Application Manager auf den Geräten gesteuert wird. Zur Entwicklung von Bedienoberflächen stehen zwei Konzepte zur Verfügung. Das High Level UI enthält vordefinierte Elemente wie Listen und Formulare, deren Aussehen und Funktionalität weitgehend von der MIDP-Implementation bestimmt wird. Damit soll ein einheitliches Erscheinungsbild aller Anwendungen eines mobilen Gerätes sichergestellt werden. Außerdem wird so, ganz im Sinne von Kapitel 3, eine gute Benutzbarkeit unterstützt. Das Low Level UI stellt eine Zeichenfläche zur Verfügung, die relativ frei genutzt werden kann. Es ist für speziellere Oberflächen gedacht.

Ebenfalls vorhanden sind Bibliotheken zur Persistenz (das Record Management System) und für Netzwerkverbindungen (das Generic Connection Framework).

Im Anhang B findet sich eine ausführliche Beschreibung dieses wichtigen Profils mit Details zu weiteren Konzepten.

6.7 Optional Packages for PDAs – Java für Personal Digital Assistants

Zwei Optional Packages erweitern die CLDC 1.0 um Funktionalität, die der größeren Leistungsfähigkeit der Personal Digital Assistants Rechnung trägt. Sie können nicht nur von PDAs und Smartphones benutzt werden, sondern auch von anderen Geräten,

[1] Nach Angabe von Sun Microsystems vom März 2002 wurden 50 Millionen Mobiltelefone mit J2ME bis Ende 2002 erwartet und 400 Millionen in den folgenden beiden Jahren.

welche die Anforderungen erfüllen, wie z. B. Highend-Mobiltelefonen. Die Pakete sind das PIM Optional Package und das FileConnection Optional Package.

Die Optional Packages sind aus der Entwicklung eines eigenen Profils für PDA, des Personal Digital Assistant Profile (PDAP) hervorgegangen, die eingestellt wurde. Obwohl sie im selben JSR entwickelt wurden, sind sie unabhängig voneinander einsetzbar. Seit März 2003 liegen die Proposed Final Drafts vor.

6.7.1 Personal Information Management

Eine wichtige Erweiterung gegenüber MIDP ist die Möglichkeit, Daten des Personal Information Management (PIM) mit den nativen Datenbanken des PDAs austauschen zu können. Dabei handelt es sich um Adreßbuch-, Kalender- und Aufgabenliste-Datenbanken. Optional können auch externe Daten, wie solche auf SIM-Karten von Mobiltelefonen, gelesen werden. Wichtig dabei ist die Datensicherheit, so daß Java-Applikationen nicht unberechtigt auf diese Informationen zugreifen können. Es ist dem Benutzer daher möglich, Rechte zu vergeben (nur lesender Zugriff, nur schreibender Zugriff, lesen und schreiben).

Als Industriestandard zum Im- und Export der Daten werden das vCard 2.1 und das vCalendar 1.0 Format[2] unterstützt. Diese sind in die meisten PIM-Applikationen integriert und definieren Felder und Kategorien, die über Anwendungsgrenzen hinweg unterstützt werden.

6.7.2 Schnittstellen

Das FileConnection Optional Package erweitert das Generic Connection Framework von CLDC um Zugriffsmöglichkeiten auf das Dateisystem der Geräte und vorhandene Speicherkarten. Unterstützt werden können alle gängigen Typen von Speicherkarten, u. a. Smart Media Cards, Compact Flash Cards, Secure Digital Cards, MultiMedia Cards und Memory Sticks.

Aus Gründen der Datensicherheit darf nicht auf Record Stores, die mit dem Record Management System von MIDP erstellt wurden, über die API für Dateisysteme zugegriffen werden. Außerdem kann der Zugriff auf private und betriebssysteminterne Dateien beschränkt werden.

Das zentrale neue Interface ist FileConnection, das eine StreamConnection (siehe Abschnitt B.7 im Anhang) spezialisiert. Über das Schema file://<host>/<path> können Verbindungen angesprochen werden, wobei <host> den Domänennamen bezeichnet und <path> den Pfad, der aus der Wurzel (gibt die Quelle an, z. B. / für die oberste Ebene des Dateisystems oder SDCard für eine Speicherkarte) und optional Verzeichnissen und einem Dateinamen besteht. Über Listener, die an zentraler Stelle registriert werden, können Applikationen über das Einsetzen und Entfernen von Speicherkarten informiert werden und darauf reagieren.

[2] Diese Formate werden seit Ende 1996 durch das Internet Mail Consortium gepflegt, siehe `http://www.imc.org/pdi/`

6.8 Bewertung

In diesem Abschnitt erfolgt eine kurze Bewertung der J2ME Technologie und insbesondere CLDC/MIDP unter Berücksichtigung ihrer Eignung für mobile Anwendungen nach dem WAM-Ansatz.

Die Java 2 Micro Edition steht mit ihrem Profil MIDP fast konkurrenzlos da, wenn es um die Programmierung von Mobiltelefonen geht. Bis auf BREW (siehe Seite 13) hat sich keine andere Technik, die eine Programmierung der Geräte durch Nichthersteller ermöglicht, durchgesetzt. Mobiltelefone mit der J2ME dagegen sind in ausreichender Zahl verfügbar. MIDP hat sich hier als Standard etabliert. Auch für PDAs und andere mobile Geräte gibt es nur wenige übergreifende Systeme, von denen sich kein anderes durchgesetzt hat. Für J2ME sind virtuelle Maschinen auf den meisten Plattformen von mehreren Herstellern verfügbar.

Ein weiterer Vorteil ist die Einhaltung der Java Sprachdefinition. Von wenigen Ausnahmen (z. B. Unterstützung von Gleitkommazahlen) abgesehen, ist diese unverändert. Die Anbindung an das javabasierte JWAM-Rahmenwerk kann dadurch ohne Bruch erfolgen und vorhandene Kenntnisse und Softwarewerkzeuge können weiterhin genutzt werden. Nach der Erfahrung des Autors ist das MIDP einfach zu verwenden. Gerade der Verzicht auf komplexere Bedienschnittstellenklassen begünstigt eine schnelle Erlern- und leichte Benutzbarkeit. Über die Optional Packages für PDAs kann eine weitergehende Unterstützung für leistungsfähigere Geräte ohne Bruch eingebunden werden.

Nachteilig ist, daß nicht immer vollkommene Kompatibilität herrscht. Einige Elemente sind auf verschiedenen Plattformen unterschiedlich implementiert; Möglichkeiten und Beschränkungen (z. B. Speicherplatz für persistente Daten, Anzahl gleichzeitig geöffneter Netzwerkverbindungen) weichen voneinander ab. Desweiteren haben die meisten Hersteller ihre Versionen um proprietäre Zusatz-APIs erweitert, um speziellere Funktionen nachzurüsten, die J2ME nicht bietet.

Das Personal Profile ist fast so mächtig wie die J2SE. Auf Geräten, die es unterstützen, können komplexere Interaktionen realisiert werden. Insbesondere die Übernahme vorhandener Codeteile fällt dadurch deutlich einfacher als mit MIDP.

Kapitel 7

Beispielprojekte

In diesem Kapitel werden zwei Projekte beschrieben, an denen der Autor mitgewirkt hat. Sie zeigen die praktische Umsetzung einiger der in den vorangegangenen Kapiteln beschriebenen Konzepte. Für jedes mobile Leitbild wurde ein Projekt gewählt, um die Vielfalt des Themas aufzuzeigen.

7.1 Mobile Spezialanwendung für Tablett-PC

In diesem Projekt ging es um die Anbindung mobiler Geräte für den Außeneinsatz an ein parallel in Entwicklung befindliches größeres Anwendungssystem.

7.1.1 Anforderungen

Bei der Anwendung handelt es sich um ein bereichsübergreifendes und integratives Anwendungssystem für den größten deutschen Wasserver- und Abwasserentsorger. Primär sollen mit der Anwendung die Prozesse zur Betriebsführung und Instandhaltung optimiert werden. Methodisch wird die Anwendung sozioökonomisch, d. h. bedarfsgerecht und anwendermotivierend, erstellt.

Das System wird mit JWAM 1.7.0 nach dem WAM-Ansatz unter Einsatz von XP-Techniken entwickelt. Die Werkzeuge werden mit Swing unter J2SE realisiert. DB2 arbeitet als relationales Datenbanksystem im Hintergrund. Desweiteren werden externe Systeme wie SAP und ein geographisches Informationssystem (GIS) eingebunden.

Die mobile Anwendung soll bei der Kanalreinigung eingesetzt werden. Auf außeneinsatztauglichen Tablett-PCs mit Windows CE sollen sich die Arbeiter vor Ort über die auszuführenden Arbeiten informieren können. Jedes Reinigungsfahrzeug wird mit einem mobilen Rechner ausgestattet. Kartenausschnitte zeigen die Umgebung und den Verlauf der Kanäle an. Hauptaufgabe der Anwendung ist die Erfassung des Kanalzustandes. Dazu sind einige Eintragungen vorzunehmen, z. B. die Angabe des Wasserstandes und des Verschmutzungsgrades, unzulässige Einleitungen von Fremdstoffen u. ä. Außerdem ist für den Nachweis der Arbeitszeit eine Stundenrückmeldung auszufüllen, welche die individuell je Arbeiter gearbeitete Zeit pro Kanalrohrklasse dokumentiert.

Für Arbeitsvorbereiter, welche die Touren der Reinigungswagen planen, wurde schon ein stationäres System entwickelt, das eine ähnliche Funktionalität besitzt: Es lassen sich Umgebungskarten anzeigen, Arbeitsaufgaben planen sowie Kanalzustandsrückmeldungen und Stundenrückmeldungen aufrufen und bearbeiten.

7.1.2 Gerätetechnik und Systemarchitektur

Die mobile Anwendung folgt dem zweiten mobilen Leitbild, das oben eingeführt wurde: Sie ist ein Funktionsarbeitsgerät für eigenverantwortliche Expertentätigkeit im mobilen Einsatz. Gerät und Anwendung sollen auf die oben beschriebene Aufgabe der Dokumentation der Kanalreinigung optimal zugeschnitten werden. Beim mobilen Gerät handelt es sich um einen Tablett-PC unter Windows CE 3.0, der über einen großen farbigen Touchscreen mit 800×600 Pixeln Auflösung verfügt und somit auch komplexere Bedienoberflächen darstellen kann. Er ist für den Einsatz unter schwierigen Bedingungen geeignet und stoß- und spritzwassergeschützt. Als Entwicklungssprache wurde wie im stationären Anwendungssystem Java gewählt, das auf dem mobilen Gerät in Form einer Virtuellen Maschine für Personal Java 3.1 verfügbar ist. Zum Datenaustausch mit dem Anwendungssystem wird eine an einen Desktoprechner über die serielle Schnittstelle angeschlossene Docking Station verwendet. ActiveSync dient zur Übertragung der Daten. Der Anschluß von fachspezifischen Zusatzgeräten, wie z. B. eines Wasserzähler-Lesegerätes, ist möglich.

Arbeitstechnisch ist die Integration der mobilen Anwendung so gelöst, daß morgens vor Arbeitsbeginn oder am Vorabend die für den Tag zu bearbeitenden Aufträge und Kanäle inklusive der dazugehörigen Daten (vorhandene Kanalzustandsrückmeldungen und Kartenausschnitte) aus dem Anwendungssystem auf das Gerät übertragen werden. Nach Ende der Arbeit im Außendienst werden die ausgefüllten Kanalzustandsrückmeldungen und Stundenrückmeldungen in das System eingepflegt. Dazu dient das Synchronisierer-Werkzeug, das dem Entwurfsmuster Synchronisierer (vgl. Abschnitt 5.3) folgt, siehe Abbildung 7.1.

Abbildung 7.1: Screenshot Projekt 1: Synchronisierer

Nach Verbindung eines mobilen Gerätes mit einem PC kann ein Fahrzeug ausgewählt werden und die Daten der laut Einsatzplan für dieses Fahrzeug abzuarbeitenden Aufträge werden kopiert. Nach der Arbeit besteht die Option, die Daten des mobilen Gerätes mit denen des Anwendungssystems zu synchronisieren. Dabei werden Änderungen an den Daten in die Datenbank des Anwendungssystems eingepflegt.

7.1.3 Implementierungsdetails der Anwendung

Da ein großer Teil der für die mobile Anwendung benötigten Funktionalität schon im System vorhanden war, bot es sich an, diese zu nutzen. Das Gerät und die Java-Ablaufumgebung sind leistungsfähig genug, um auch speicher- und rechenintensivere Teile zu verarbeiten.

Anhand dreier Werkzeuge soll die praktische Umsetzung verschiedener in Kapitel 5 diskutierter Strategien gezeigt werden.

Mobiler Rückmelder

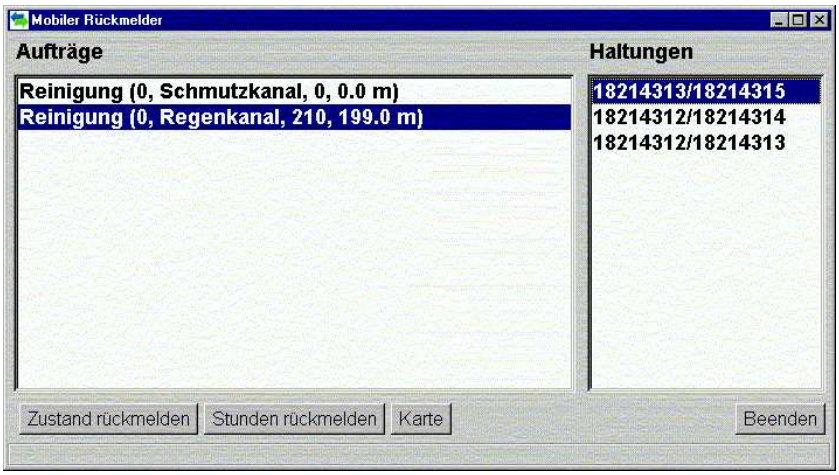


Abbildung 7.2: Screenshot Projekt 1: Mobiler Rückmelder

Der mobile Rückmelder ist das Hauptwerkzeug der mobilen Anwendung, das die anderen aufruft. Es zeigt zwei Listen: Aufträge und dazu passende Haltungen (einzelne Kanalabschnitte), siehe Abbildung 7.2.

Die Konstruktion ist Fall 3 zuzuordnen, vgl. Abschnitt 5.1.4: Das Werkzeug wurde für die mobile Anwendung neu entworfen, da im System keines mit passender Funktionalität vorhanden war. Die zugrundeliegenden Materialien sind dieselben. Auch die Fachlichen Services wurden übernommen, allerdings mit veränderter Realisierung des Persistenzsystems. Dieses wird durch die Fachlichen Services des Anwendungssystems gekapselt. Es existierten schon Implementationen für das Dateisystem (für die Entwicklung und zum Testen) und für eine Datenbank (für das Produktivsystem). Für das mobile Gerät mußte eine Lösung gefunden werden, welche den Zugriff auf die Materialien ohne Verbindung zum Anwendungssystem erlaubt. Dazu wurden die benötigten Services zusätzlich in XML implementiert. Bei der Übertragung der Daten holt der Synchronisierer die benötigten Materialien aus den im Anwendungssystem laufenden Services und fügt sie in die XML-Services ein. Diese erzeugen XML-Dateien mit den Daten der Materialien. Sie werden auf das mobile Gerät übertragen und dort

in den XML-Services verwendet. Bei der Rücksynchronisierung ins Anwendungssystem werden die geänderten XML-Dateien durch den Synchronisierer übertragen und mit den im Anwendungssystem vorhandenen Materialien abgeglichen.

Beim Design wurde auf eine gute Benutzbarkeit geachtet. Die Größe der Buttons und der Listen wurde so gewählt, daß sie sich gut lesen und mit dem Stift auswählen lassen.

Kartenanzeiger

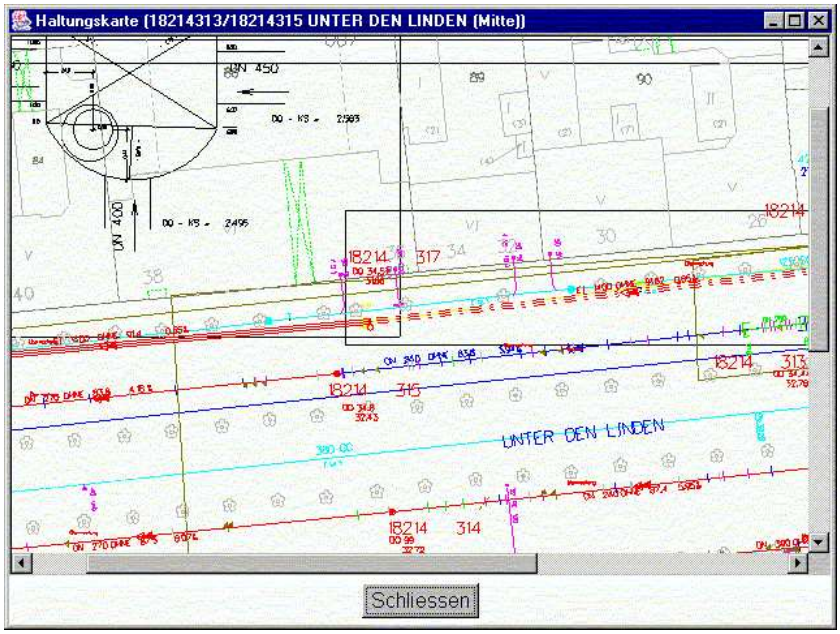


Abbildung 7.3: Screenshot Projekt 1: Kartenanzeiger

Dieses Werkzeug zeigt Übersichtskarten zu Aufträgen und Detailkarten einzelner Haltungen an. Obwohl im Anwendungssystem schon ein ähnliches Werkzeug vorhanden ist, wurde es von Grund auf neu entwickelt (Fall 1, vgl. Abschnitt 5.1.2).

Das vorhandene Werkzeug greift auf Daten des geographischen Informationssystems zu. Dadurch ermöglicht es die Darstellung beliebiger Kartenausschnitte nach Eingabe einer Adresse und kann sie in verschiedenen Maßstäben anzeigen. Auf dem mobilen Gerät ist das nicht möglich, da im Außeneinsatz keine Verbindung zu einem GIS-Server besteht. Daher erzeugt das Synchronisierer-Werkzeug beim Kopieren der Daten auf das mobile Gerät die benötigten Bilddateien im GIF-Format und überträgt sie. Der Kartenanzeiger auf dem mobilen Gerät kann diese Bilder anzeigen (siehe Abbildung 7.3).

Kanalzustandsrückmelder

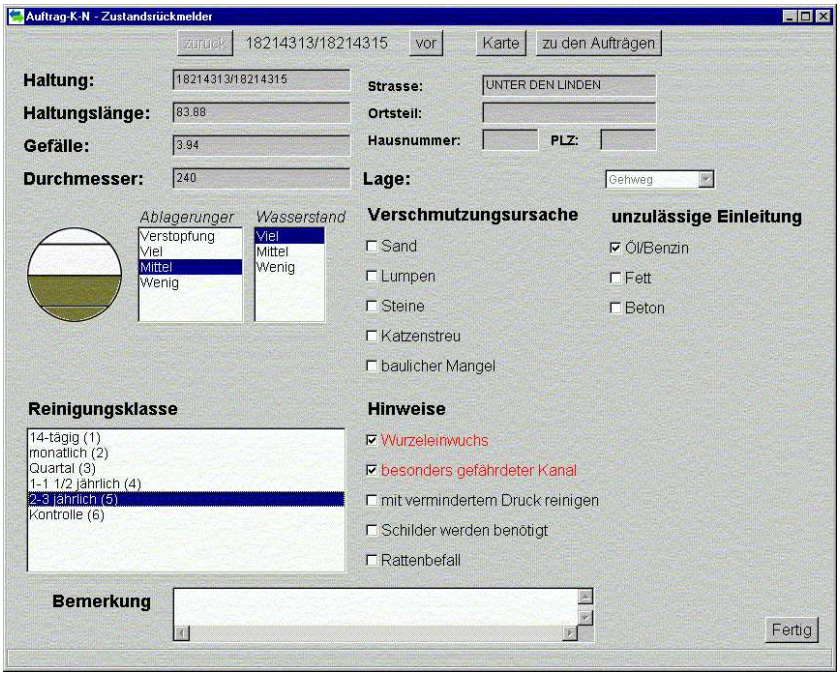


Abbildung 7.4: Screenshot Projekt 1: Kanalzustandsrückmelder

Der Kanalzustandsrückmelder ist ein wichtiges Werkzeug der mobilen Anwendung, mit dem der Kanalzustand erfaßt werden kann. Dafür können diverse Angaben vorgenommen werden, siehe Abbildung 7.4.

Im Anwendungssystem läuft ein Kanalzustandsrückmelder als Subtool innerhalb eines umfangreichen Werkzeugs für den Arbeitsvorbereiter, der u. a. für die Erstellung neuer Aufträge darauf zurückgreifen kann. Es wurde festgestellt, daß das vorhandene Werkzeug genau die Funktionalität bietet, die für die mobile Anwendung sinnvoll ist. Allerdings greift es über Services auf die Materialien in Datenbanken zu und benutzt Swing für die Bedienoberfläche, während das mobile Gerät bei der Benutzung im Außeneinsatz keinen Zugriff auf die Datenbanken bietet und nur AWT als GUI-API zur Verfügung steht.

Daher wurde nach dem in Abschnitt 5.1.6 entwickelten Verfahren das Werkzeug mit IAK, FK, Fachlichen Services und Materialien übernommen und nur die GUI-Komponente und die Services wurden neu geschrieben (Fall 5 im Schema).

Die Anbindung des GUIs ist mit Interaktions- und Präsentationsformen gelöst, so daß sich der Wechsel der Oberflächenkomponenten einfach gestaltete. Es waren nur die benötigten Präsentationsformen in AWT neu zu realisieren. Dabei wurde auf eine gute Benutzbarkeit auf mobilen Geräten geachtet, z. B. durch Verwendung einer leichter mit einem Stift anwählbaren Liste von Elementen anstelle einzelner Radiobuttons und den

Einsatz von Farben, welche die Auswahl von Elementen auf einen Blick zeigen. Außerdem ist die Schriftart und die Größe und Anordnung der einzelnen Elemente dem kleinen Bildschirm angepaßt worden. Die bestehende Interaktionskomponente konnte dank der Abstraktion durch Interaktionsformen bis auf zwei kleine Anpassungen weiterverwendet werden.

7.1.4 Bewertung

In diesem Projekt konnte eine Vielzahl von theoretisch erarbeiteten Konzepten praktisch umgesetzt werden. Hauptsächlich betrifft dies die Konstruktionsmuster. Für verschiedene Werkzeuge kamen unterschiedliche Strategien der gemeinsamen Verwendung von Komponenten in stationärem System und mobiler Anwendung zum Einsatz. Hier zeigte sich, daß die konzeptionellen Überlegungen sinnvoll für den Gebrauch in größeren praktischen Projekten anwendbar sind.

Eine wichtige Rolle spielten auch software-ergonomische Überlegungen, um eine gute Benutzbarkeit zu erreichen. In Autor-Kritiker-Zyklen mit späteren Anwendern des Systems wurden getroffene Entwurfsentscheidungen rückgekoppelt. Die Präsentation konnte so gestaltet werden, daß sie den Möglichkeiten des mobilen Gerätes und der vorgesehenen Einsatzweise entspricht und sich trotzdem eng an den entsprechenden Anwendungen des stationären Systems orientiert.

7.2 Erweiterung eines Mobiltelefons zum persönlichen Arbeitsgerät

Dieser Abschnitt beschreibt ein Projekt, das den Gedanken des ersten mobilen Leitbildes ausführt. Ein Mobiltelefon wird mit Zusatzfunktionen für berufliche Aufgaben ausgestattet, um ein persönliches Gerät zu schaffen, das jederzeit Zugriff zu wichtigen privaten und beruflichen Informationen verschafft.

7.2.1 Unterstützte Funktionen und Hardwarevoraussetzungen

Auf heutigen typischen Mobiltelefonen für den Geschäftskundenmarkt laufen schon eine Reihe von Anwendungen zum Zugriff auf persönliche Informationen. Diese umfassen in der Regel ein Adreßbuch mit Kontaktinformationen, einen Kalender mit Terminverwaltung und eine Aufgabenliste.

Die vorinstallierten Applikationen werden in diesem Projekt prototypisch um beruflich relevante erweitert. Diese sind eine Datenbank mit technischen Informationen zu angebotenen Produkten, eine Kunden- und Terminverwaltung für den Außendienst und die Möglichkeit, Besprechungsräume im voraus zu reservieren.

Damit das Gerät erweiterbar ist, muß es über die Möglichkeit verfügen, Programme zu installieren und auszuführen. In diesem Fall wird die Plattform J2ME in der Ausprägung CLDC/MIDP verwendet, die auf vielen modernen Mobiltelefonen vorhanden ist, vgl. Kapitel 6. Die Entwicklung erfolgte auf einem Desktop-Rechner unter Verwendung eines Emulators. In Abbildung 7.5 ist darin die Arbeitsumgebung zu sehen.

Mittels der Navigationstaste können Menüpunkte angewählt werden. Darüber liegen links und rechts Softbuttons für den schnellen Zugriff auf Menüpunkte.

7.2.2 Beispielablauf

Das Beispiel des Eintragens eines neuen Termins soll einen typischen Ablauf demonstrieren. Ein Kundenberater wählt aus dem Arbeitsplatz das Werkzeug Kunden-Verwaltung aus. Er erhält eine Liste mit allen verfügbaren Kunden (Abb. 7.6 links). Hier kann er neue Kontakte eintragen oder auf Daten der vorhandenen zugreifen. Er wählt eine Kundin aus. Dadurch werden ihm die dazugehörigen Daten angezeigt. Die Liste ist länger als die Karte, deshalb kann er mit der Navigationstaste vertikal scrollen (Abb. 7.6 mitte und rechts).

Der Kundenberater erkennt, daß kein neuer Termin vereinbart wurde. Daher ruft er die Kundin unter der hinterlegten Telefonnummer an und vereinbart einen neuen Termin. Diesen trägt er anschließend in die Anwendung ein. Da in der Karte mit den Kundendetails mehr als zwei Kommandos zur Verfügung stehen, kann er mit dem rechten Softbutton ein Menü öffnen (Abb. 7.7 links). Er wählt den letzten Eintrag aus und sieht in einer neuen Karte relevante Informationen zur Terminverwaltung (Abb. 7.7 mitte). Auf der nächsten Karte (Abb. 7.7 rechts) kann er Datum und Uhrzeit auswählen.

7.2.3 Implementierungsdetails

Die Applikationen bauen auf einem kleinen Rahmenwerk auf, welches die grundlegenden WAM-Metaphern überträgt und auf MIDP abbildet. Werkzeuge sind als MIDlets realisiert. Dadurch können mehrere Werkzeuge in einer MIDlet-Suite zusammen installiert werden. Sie können auf gemeinsame Ressourcen zugreifen, u. a. die Materialien, die als herkömmliche Java-Klassen realisiert sind. Eine Liste mit den MIDlets einer MIDlet-Suite kann so als Arbeitsplatz interpretiert werden, auf dem Werkzeuge liegen.

Zur Realisierung der Werkzeuge wird ein DisplayManager verwendet, der für die Verwaltung von Displayables nach dem Stapel-Prinzip zuständig ist (genauer beschrieben in [Muchow 02]). Damit können Karten definiert werden, zwischen denen navigiert werden kann. Ein Werkzeug besteht so aus einer Zusammenfassung mehrerer Karten.

Schon mit dieser recht einfachen Struktur lassen sich kleine WAM-Anwendungen erstellen. Es lassen sich keine komplexen Interaktionsformen nutzen, doch reicht der Funktionsumfang der Werkzeuge für einfache persönliche, mobile Arbeitsgeräte aus.

7.2.4 Bewertung

Das Projekt zeigt, mit welch einfachen Mitteln nützliche Anwendungen auf sehr eingeschränkten mobilen Geräten realisierbar sind. Es zeigt ebenfalls, daß viele in dieser Arbeit entwickelte Konzepte umsetzbar sind, insbesondere das Leitbild eines mobilen

Arbeitsgerätes mit Arbeitsumgebung, Werkzeugen und Materialien. Die Wichtigkeit der Gestaltung nach software-ergonomischen Grundlagen wird auch deutlich.

Die Konstruktionsmuster sowie ausgefeilte Architekturüberlegungen mit mehreren Schichten sind eher für die Umsetzung auf leistungsstärkeren Geräten wie PDAs und Tablett-PCs gedacht. Auf heutigen Mobiltelefonen lassen sie sich noch nicht befriedigend realisieren.

Abbildung 7.5: Screenshot Projekt 2: Arbeitsumgebung im Emulator

Abbildung 7.6: Screenshot Projekt 2: Eintragen eines Termins 1/2

Abbildung 7.7: Screenshot Projekt 2: Eintragen eines Termins 2/2

Kapitel 8

Schlußwort

8.1 Zusammenfassung

Das Thema dieser Arbeit war die Frage, inwieweit sich der WAM-Ansatz zur Erstellung von Anwendungen für mobile Geräte eignet. Desweiteren wurde untersucht, welche Unterschiede zu einer Entwicklung für stationäre Systeme bestehen und inwiefern vorhandenes Wissen der Entwickler und bestehende Architekturen und Programmteile für mobile Anwendungen genutzt werden können.

Dazu wurden zuerst Konzepte zur Mobilität untersucht und Forschungsrichtungen aufgezeigt. Klassen mobiler Geräte mit ihren typischen Eigenschaften wurden unter Berücksichtigung ihrer Eignung als Anwendungsplattform beschrieben.

Anschließend wurden software-ergonomische Grundlagen zusammengestellt, welche insbesondere für die Programmentwicklung auf kleinen, leistungsschwachen mobilen Geräten eine Rolle spielen. Nach der Behandlung von allgemeinen software-ergonomischen Grundlagen wurden dazu die Besonderheiten mobiler Geräte betrachtet. In diesem Rahmen wurde die spezielle Rolle von Consumer Devices untersucht und es wurden unterschiedliche Ansätze von Betriebssystemen und Programmen für PDAs dargestellt. Das Modell der Usage Spaces hat der Autor zur Beschreibung der Möglichkeiten persönlicher mobiler Geräte benutzt und später um einen Bereich für mobile Arbeit erweitert. Der Interaktion mit mobilen Geräten mit ihren spezifischen Ein- und Ausgabemöglichkeiten wurden eigene Abschnitte gewidmet.

Dann wurde der WAM-Ansatz betrachtet und seine für diese Arbeit relevanten Elemente wurden vorgestellt. Darauf aufbauend wurde untersucht, inwieweit sich die grundlegenden WAM-Leitbilder für mobile Anwendungen eignen. Der Autor ist zu dem Ergebnis gekommen, daß neue Leitbilder wichtig sind, um die Besonderheiten des mobilen Kontextes zu betonen, und hat aus etablierten Leitbildern zwei neue mobile Leitbilder abgeleitet und erläutert. Danach wurden Metaphern des WAM-Ansatzes untersucht und es wurde gezeigt, daß sie sich – bei einigen mit gewissen Einschränkungen – auch sinnvoll für mobile Systeme verwenden lassen.

Mit dem WAM-Ansatz lassen sich mehrkanalfähige Anwendungen erstellen. Das grundlegende Architekturmodell dafür wurde vorgestellt. Fachliche Services, die Anwendungsfunktionalität bündeln und mehreren unterschiedlichen Kunden zur Verfügung stellen können, wurden erläutert. Darauf aufbauend wurde beschrieben, wie mit diesem Ansatz mobile Geräte integriert werden können.

Nachfolgend wurden Konstruktionsmuster betrachtet, die Hilfsmittel zur konstruktiven Umsetzung der vorgestellten Leitbilder und Metaphern bieten. Dabei wurden die Fälle unterschieden, daß eine mobile Anwendung im Rahmen eines größeren Anwendungssystems entsteht und daß nur eine singuläre mobile Anwendung entwickelt werden soll. Für den ersten Fall wurden die Basiselemente von WAM-Anwendungen, Werkzeuge, Fachliche Services und Materialien, mit ihrem festgelegten inneren Aufbau als Grundlage genommen, um verschiedene Stufen der Übernahme im Anwendungssystem vorhandenen Codes herauszuarbeiten. Diese reichen von der vollständigen Neuentwicklung über den selektiven Austausch einiger Elemente bis zu einer größtenteils unveränderten Übernahme zu Prototyping- und Testzwecken. Für den zweiten Fall hat der Autor wichtige zu beachtende Punkte zusammengestellt und begründet, warum vorhandene Entwurfsmuster zur Konstruktion von Anwendungen auf wenig leistungsfähigen Geräten nicht immer anwendbar sind. Mit dem Synchronisierer, der Daten des stationären Anwendungssystems und mobiler Geräte abgleicht und überträgt, wurde ein neues Entwurfsmuster untersucht und eingeführt.

Anschließend wurde die auf mobilen Geräten weitverbreitete Programmierplattform Java 2 Micro Edition mit ihren Konfigurationen, wichtigen Profilen und optionalen Paketen vorgestellt. Dabei wurden ausführlich das für die Entwicklung von Anwendungen für kleine mobile Geräte gedachte Profil MIDP und damit verwendbare optionale Pakete für PDAs erläutert.

Beispielprojekte haben die praktische Umsetzung mehrerer in dieser Arbeit vorgestellter Konzepte gezeigt und interessante Implementierungsdetails erläutert.

8.2 Bewertung

Durch Untersuchung der bestehenden Grundlagen des WAM-Ansatzes vor dem Hintergrund der Besonderheiten mobiler Geräte und den Anforderungen mobiler Anwendungen hat der Autor herausgearbeitet, daß sich die meisten Elemente für eine Entwicklung im mobilen Kontext anpassen und übernehmen lassen.

Die neu eingeführten Leitbilder, die aus etablierten abgeleitet sind, lenken die Aufmerksamkeit der Entwickler auf die zu berücksichtigenden Besonderheiten. Gleichzeitig können sie von ihrer Erfahrung mit dem Design stationärer Anwendungssysteme nach dem WAM-Ansatz profitieren. Dazu trägt auch die Übernahme von WAM-Metaphern bei. Durch diese Überlegungen wird die Entwicklung mobiler Anwendungen nahtlos in den WAM-Ansatz integriert.

Auch für die technische Umsetzung gibt der Autor durch Konstruktionsmuster Hilfestellungen. Die Unterscheidung in verschiedene Fälle macht die Übernahme von bestehendem Programmcode auf vielen Ebenen möglich, je nach den Eigenschaften des Anwendungssystems und der Leistungsfähigkeit der eingesetzten mobilen Geräte. Die Orientierung an den Abstraktionen und Schichten von WAM-Systemen mit den vorgestellten Mustern ermöglicht eine weitgehende, sinnvolle Wiederverwendung von Design und Programmtexten. Die Synchronisation, die bei den meisten mobilen Anwendungen eine wichtige Rolle einnimmt, wird durch ein eigenes Entwurfsmuster unterstützt.

In den vorgestellten Beispielprojekten und weiteren haben die in dieser Arbeit entwickelten Konzepte ihre praktische Umsetzbarkeit gezeigt. In folgenden Projekten wird sich ihre Tauglichkeit weiter beweisen müssen.

Damit wurde ein guter Überblick über die Möglichkeiten der Entwicklung nach dem WAM-Ansatz auf diesem heute schon wichtigen und zukünftig voraussichtlich noch bedeutenderem Gebiet gegeben und es wurden an vielen Stellen im Detail wichtige Punkte herausgearbeitet.

8.3 Ausblick

Aufgrund des Umfangs dieses Themas ist es nicht möglich, in einer Arbeit alle Aspekte im Detail auszuleuchten. Ich sehe die Arbeit als Grundlage, auf deren Basis vertiefende Arbeiten in Teilbereichen möglich sind.

Ich habe festgestellt, daß sich viele Entwurfsmuster nicht einfach auf mobile Anwendungen für leistungsschwache Geräte übertragen lassen, weil sie mit vielen Abstraktionen und Indirektionen arbeiten, die zu vielen Objekten und damit zu Performanzproblemen führen. Hier wäre es interessant zu ergründen, welche neuen Entwurfsmuster zusätzlich zum Synchronisierer für eine objektorientierte Entwicklung auf solch kleinen Geräten sinnvoll sind. Ansätze dazu existieren bereits. Jörg Roth hat schon einige „Patterns of Mobile Interaction“ herausgearbeitet, siehe [Roth 02b].

Wie ich in Kapitel 5 diskutiert habe, kann auf leistungsstarken Geräten vieles von der Konstruktion von Anwendungen für stationäre Systeme übernommen werden. Daher können auch abgespeckte Versionen von bestehenden Rahmenwerken, wie z. B. JWAM, eingesetzt werden. Dies wurde in einigen Projekten schon erfolgreich praktiziert. Für Geräte, auf denen dies nicht möglich ist, kann es trotzdem sinnvoll sein, mit Rahmenwerken zu arbeiten. Wenn in mehr Projekten nach dem WAM-Ansatz mobile Anwendungen entwickelt worden sind, wird es möglich sein, aus den gesammelten Erfahrungen ein entsprechendes Rahmenwerk zu extrahieren.

Auch im software-ergonomischen Bereich lassen sich weitere Forschungen anschließen. Mobile Geräte bieten interessante Möglichkeiten für innovative Mensch-Maschine-Schnittstellen, wie ich einige schon in Abschnitt 3.3 vorgestellt habe. Auch der Umgang mit mobilen Anwendungen sowohl im privaten als auch im beruflichen Bereich kann im Mittelpunkt weiterer Nachforschungen stehen.

Letztlich bleibt zu sagen, daß das gesamte Gebiet von mobilen Geräten sehr in Bewegung ist. Durch die dynamische Entwicklung von Hardware und Software stellt es meiner Ansicht nach ein sehr interessantes Forschungsgebiet dar. Während einige Beschränkungen, die in dieser Arbeit ausführlich diskutiert werden, wohl auch zukünftig bestehen werden, wie die geringe Größe der Ausgabebildschirme und unkonventionelle Eingabemethoden, werden andere wahrscheinlich schon bald keine so große Rolle mehr

spielen, wie beispielsweise die mangelnde Prozessorleistung, der begrenzte Speicherplatz und eingeschränkte Netzverbindungen. In Kombination mit neuartigen Geräteklassen, wie z. B. Wearables, und bald verstärkt verfügbaren Diensten, wie z. B. den Location Based Services (LBS), ergeben sich faszinierende Möglichkeiten.

Anhang A

Der Java Community Process

Im Java Community Process entstehen Standards für die Java 2 Plattform. In ihm werden **Java Specification Requests (JSR)** bearbeitet. In einem JSR wird empfohlen, eine neue Spezifikation zu entwickeln oder eine bedeutende Änderung an einer bestehenden durchzuführen. Eine Spezifikation definiert dabei Aspekte der Java Technologie, wie die Sprache, die virtuelle Maschine, Plattformversionen, Profile oder Application Programming Interfaces (APIs), vgl. [JCP 01].

Der JCP wird vom Program Management Office (PMO) administriert. Dieses ist eine Gruppe innerhalb Sun Microsystems, die auch dem Exekutivkomitee (EC) vorsteht, dessen Mitglieder die Entwicklung der Java Technologie leiten.

Die Definition der J2ME ist eine Sammlung sogenannter Building Blocks. Diese sind eine Untermenge einer oder mehrerer APIs, die in der J2SE oder J2EE Platform Edition Specification definiert sind. APIs sollen auf diese Art und Weise wiederverwendet werden, Parallelentwicklungen vermieden werden. Spezifikationen von Profilen für die J2ME entstehen durch Kombination vorhandener Building Blocks mit neuen APIs, welche Funktionalität implementieren, die noch nicht in anderen Bibliotheken vorhanden ist.

Building Blocks für J2ME lassen sich schnell definieren oder ändern, um der rasanten Entwicklung auf dem Markt für mobile Geräte Rechnung zu tragen.

Der Java Community Process umfaßt in der Version 2.1[1] folgende Schritte, hier vereinfacht dargestellt (vgl. Abbildung A.1):

1. **Anstoß einer neuen oder überarbeiteten Spezifikation:** Nachdem ein JSR beim Program Management Office eingegangen ist, erhält er eine Nummer und wird veröffentlicht. Das Exekutivkomitee entscheidet innerhalb von 14 Tagen, ob er genehmigt wird.

2. **Erstellung des Community Draft:** Nach der Genehmigung wird eine Expertengruppe unter Leitung des Specification Lead gebildet. Diese wählt selbständig ihren Arbeitsstil. Im allgemeinen werden Gruppensitzungen und Konferenzen abgehalten sowie auf Mailinglisten und in Webforen diskutiert. Es entsteht ein erster Entwurf.

[1] Seit kurzem ist Version 2.5 in Kraft, die sich aber nicht sehr unterscheidet. Da die meisten der in dieser Arbeit vorgestellten Spezifikationen nach Version 2.1 erstellt wurden, wird diese hier vorgestellt.

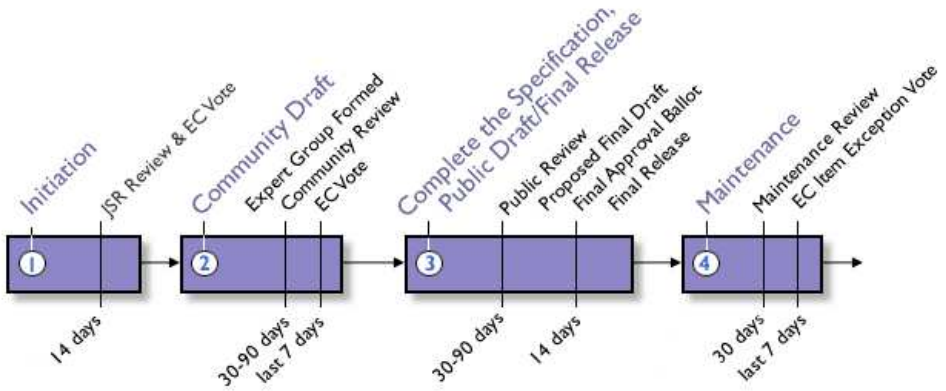


Abbildung A.1: Abfolge der Schritte im Java Community Process, aus: `http://www.jcp.org/en/introduction/timeline`

Dieser wird im **Community Review**, einer 30- bis 90-tägigen Zeitspanne, von Mitgliedern des JCP geprüft und kommentiert. Danach wird er sofort oder nach einer Überarbeitung innerhalb von 30 Tagen für das Public Review veröffentlicht.

3. Vervollständigung der Spezifikation: Das **Public Review** dauert ebenfalls 30 bis 90 Tage. Von den JCP-Webseiten kann der Spezifikationsentwurf von jedermann heruntergeladen werden. Kommentare können an das Expertenteam zurückfließen. Dies soll sie einarbeiten und mit einer Historie der Änderungen wieder veröffentlichen.

Nach Beendigung des Public Reviews wird der **Proposed Final Draft** erarbeitet und veröffentlicht. Auf seiner Grundlage wird die Referenzimplementation (RI) und das Technology Compatibility Kit (TCK) erstellt. Die RI ist eine prototypische „proof of concept"-Implementierung. Das TCK besteht aus Tests, Werkzeugen und Dokumentation, die es Implementatoren ermöglicht, die Vereinbarkeit ihrer Werke mit dem Standard zu prüfen.

Anschließend entsteht der **Final Draft** als endgültige Version. In einer Abstimmung, dem Final Approval Ballot, die 14 Tage dauert, wird in der Expertengruppe über die Verabschiedung des Final Drafts und der zugehörigen Referenzimplementation und des TCK abgestimmt. Im positiven Fall kommt es zum **Final Release**.

4. Wartung: Nach dem Final Release sind alle Dokumente und Programme aktuell zu halten. Es wird ein Maintenance Lead bestimmt, typischerweise der Specification Lead. Er ist dafür zuständig, Kommentare zur Spezifikationen auszuwerten, kleinere Revisionen durchzuführen und die RI und das TCK auf dem aktuellen Stand zu halten.

Anhang B

J2ME-Mobile Information Device Profile

Das Mobile Information Device Profile (MIDP) ist die Java-Variante für kleine, batteriebetriebene, eingeschränkte Geräte mit drahtloser Internetanbindung wie Mobiltelefone, PDAs und Smartphones.

B.1 Aufbau und Hardwarevoraussetzungen

MIDP 1.0 setzt auf der Konfiguration CLDC in der Version 1.0 auf. Die Hardwareanforderungen von MIDP erweitern diejenigen der zugrundeliegenden CLDC Konfiguration. Das Gerät muß mindestens über einen Bildschirm von 96×54 Pixel Größe mit 1 Bit Farbtiefe verfügen, der eine angenäherte Pixelform (Seitenverhältnis) von 1:1 haben sollte. Zur Eingabe kann entweder eine einhändige Tastatur (wie eine Standard ITU-T Mobiltelefontastatur), eine zweihändige Tastatur (z. B. mit dem QWERTY- oder QWERTZ-Layout) oder ein Touch Screen benutzt werden.

Die Speichervoraussetzungen sind mind. 128 KByte für MIDP Komponenten, mind. 8 KByte für persistente Anwendungsdaten (siehe Record Management System, Abschnitt B.6) sowie 32 KByte für die Java Runtime (Heap). Geräte müssen ferner über ein zweiwege Netzwerk verfügen, das drahtlos sein kann, keine permanente Verbindung haben muß und über begrenzte Bandbreite verfügen kann.

B.2 MIDlets und MIDlet Suite

Ein ausführbares Programm im MIDP nennt sich MIDlet, angelehnt an der Namensgebung Applet für Java-Programme in Webbrowsern. Die abstrakte Oberklasse für alle MIDlets ist javax.microedition.midlet.MIDlet. Für konkrete Programme sind einige vorgegebene Methoden zu implementieren, die zur Ablaufsteuerung genutzt werden (siehe die ausführliche Erläuterung in Abschnitt B.4).

Mehrere Programme können in einer MIDlet Suite zusammengefaßt werden. Eine MIDlet Suite ist die Komponente, die auf den mobilen Geräten installiert, benutzt und danach wieder entfernt werden kann. Daher muß auch eine Anwendung, die nur aus einem MIDlet besteht, zur Übertragung in eine MIDlet Suite eingebettet werden. Es ist sinnvoll, zusammengehörige Programme in einer Suite zu verbreiten, da so der Verwaltungsaufwand und (bei drahtloser Übertragung) Netzkosten reduziert werden.

Zu jeder MIDlet Suite gehören zwei Dateien: Das **Java Archive File (JAR)** enthält die Class-Dateien, Ressourcen wie z. B. Bilder und ein Manifest. Im Manifest sind

Metadaten über die MIDlet Suite gespeichert, beispielsweise für jedes MIDlet der Name, evtl. ein zugehöriges Icon, das von einigen Geräten angezeigt werden kann, und der Klassenname. Das **Java Application Descriptor File (JAD)** enthält Informationen über die JAR-Datei in Textform. Das sind die Informationen, die auch im Manifest des JARs stehen, und zusätzlich die URL der JAR-Datei, deren Größe, benötigter Platz für persistente Dateien und eine menschenlesbare Beschreibung. Zusätzlich sind eigene, anwendungsspezifische Attribute möglich, die beispielsweise zur Parameterübergabe genutzt werden können, ohne daß die JAR-Datei verändert werden muß.

Die Erstellung der Quellcodes, die Übersetzung und das Packen der Ressourcen kann mit den gleichen Werkzeugen und Programmen wie bei der J2SE erfolgen. Nach der Übersetzung und vor der Übertragung auf das mobile Gerät ist ein zusätzlicher Schritt nötig, das sogenannte **Preverifying**. In der J2SE wird beim Laden von Java-Klassen auf dem Zielgerät durch den Bytecode Verifier geprüft, ob sie ohne Sicherheitsprobleme ausgeführt werden können. Dieser ist mindestens 50 KByte groß. Für mobile Geräte mit ihren eingeschränkten Möglichkeiten ist das zuviel. Daher wurde hier das Verifying aufgeteilt. Auf dem Entwicklungsrechner findet das Preverifying statt, das einen Großteil des Prozesses erledigt. Dabei werden die Klassendateien durch zusätzliche Attribute angereichert, so daß sie um etwa 5% wachsen. Auf dem Zielgerät kommt dann ein einfacherer Verifier zum Einsatz, der in 10 KByte zu realisieren ist.

B.3 Übertragung und Installation

Durch die Zweiteilung der Dateien einer MIDlet Suite ist es möglich, anhand einer sehr kleinen Datei (JAD) zu prüfen, ob eine Übertragung der JAR-Datei auf das mobile Gerät gewünscht und möglich ist, u. a. ob der noch zur Verfügung stehende Speicher groß genug ist. Die Informationen in der JAD-Datei sind dann ausreichend, um die JAR-Datei zu finden, zu übertragen und installieren zu können.

Ein Addendum zur MIDP-Spezifikation legt fest, wie eine drahtlose Übertragung einer MIDlet Suite auf ein mobiles Gerät ablaufen kann. Dies wird als Over the Air Provisioning (OTA) bezeichnet. Beispielhaft könnte dieser Vorgang wie folgt aussehen:

Auf einer Webseite, z. B. in der Wireless Markup Language (WML) nach dem WAP-Standard, werden MIDlets angeboten. Der Benutzer wählt eine davon aus und erhält eine Beschreibung und eine JAD-Datei. Der **Application Manager** auf dem mobilen Gerät ist für die Verwaltung der MIDlets zuständig. Er prüft, ob alle Voraussetzungen für die Installation der ausgewählten MIDlet Suite erfüllt sind und macht dies dem Benutzer bekannt. Dieser kann sich dann für den (oft kostenpflichtigen) Download entscheiden. Der Application Manager lädt anhand der Informationen in der JAD-Datei die Archivdatei, installiert sie auf dem Gerät und registriert die enthaltene MIDlet Suite.

Neben der drahtlosen Übertragung über das Telefonnetz sind je nach Gerät auch andere Übertragungswege denkbar und realisiert, z. B. von einem Desktoprechner aus per seriellem Kabel oder durch Infrarotübertragung von einem anderen mobilen Gerät.

B.4 Lebenszyklus von Midlets

Ähnlich wie Applets in der J2SE haben MIDlets einen Lebenszyklus (siehe Abbildung B.1).

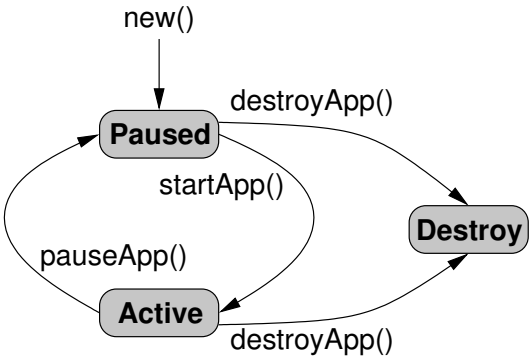


Abbildung B.1: Lebenszyklus eines MIDlets

Direkt nach ihrer Erzeugung befinden sie sich im Paused Zustand. Durch Aufruf von startApp() wechseln sie in den Zustand Active. Durch pauseApp() können sie wieder pausiert werden. In beiden Zuständen können sie durch Aufruf von destroyApp() angehalten werden. Die angegebenen Operationen werden vom Application Manager aufgerufen, bevor der Zustandswechsel erfolgt. Sie sind vom Anwendungsprogrammierer zu implementieren. In ihnen sollte auf den bevorstehenden Zustandswechsel reagiert werden, z. B. indem beim Starten die Bedienschnittstelle aufgebaut wird oder vor dem Beenden des Programms noch Parameter im persistenten Speicher gesichert werden.

Das MIDlet selbst kann seinen Zustand nicht eigenständig wechseln. Es kann jedoch dem Application Manager Bescheid geben. Die entsprechend aufzurufenden Methoden sind notifyPaused(), resumeRequest() und notifyDestroyed().

B.5 Bedienoberfläche

Eine Bedienoberfläche für mobile Geräte ist schwierig zu gestalten: Sehr eingeschränkt sind die technischen Möglichkeit, sehr heterogen die Geräte und ihr Benutzungsmodell (siehe Kapitel über Software-Ergonomie). Das UI des MIDPs, genannt Liquid Crystal Display User Interface (**LCDUI**), beschränkt sich daher auf ein einfaches Layout und Elemente, die auf allen Geräten angezeigt werden können. Es verwendet soweit wie möglich abstrakte Konzepte, die auf verschiedenen Geräten jeweils angepaßt implementiert werden können.

Display und Commands

Jedes MIDlet besitzt genau ein Display Objekt, das die Anzeige des mobilen Gerätes repräsentiert und kontrolliert. An ihm können Eigenschaften der Anzeige abgefragt wer-

den, z. B. die verfügbare Farbtiefe. Das Display kann nur eine Komponente zur Zeit anzeigen. Diese Komponenten sind Unterklassen von javax.microedition.lcdui.Displayable. Es können vordefinierte Elemente wie Formulare, Auswahllisten u. ä. sein (siehe unten: High Level UI) oder ein Canvas, das vom Anwendungsprogrammierer selbst gezeichnet werden muß (siehe unten: Low Level UI).

Das LCDUI verwendet ein ereignisbasiertes System mit Commands. Die Klasse javax.microedition.lcdui.Command modelliert ein abstraktes Kommando, das vom Benutzer ausgelöst werden kann. Abstrakt deshalb, da die konkrete Realisierung nicht festgelegt ist. Commands können z. B. an Buttons, Menüleisten oder Tastenkombinationen gebunden werden. Die konkrete Darstellung ist dabei geräteabhängig. Dies ermöglicht eine Vielfalt an unterschiedlichen Bedienkonzepten und Benutzungsmodellen. Commands sind sowohl mit dem High Level API als auch mit dem Low Level API nutzbar.

Commands werden Displayable Objekten hinzugefügt. Bei ihrer Erzeugung sind drei Parameter anzugeben:

- Ein **Label**, eine kurze Beschreibung in Textform, die für den Benutzer angezeigt werden soll. Hierbei ist auf die Länge zu achten, da bei den meisten Geräten nur begrenzt Platz zur Verfügung steht. Bei einigen Mobiltelefonen sind z. B. nur 15 Zeichen pro Zeile möglich und über Softbuttons können nur 5 Zeichen sinnvoll angezeigt werden.

- Ein **Typ**, der einer von mehreren vorgegebenen Konstanten entsprechen kann. Dies ändert nichts an der Funktionalität des Kommandos, gibt der Implementation des MIDP auf dem mobilen Gerät aber einen Hinweis, wie das Kommando am besten dargestellt werden kann. So besitzen einige Geräte beispielsweise eine eigene Taste für die Hilfefunktion oder stellen eine positive Antwortmöglichkeit („OK“) als Button immer links von einer negativen („Abbrechen“) dar.

  Mögliche Typen sind: BACK, CANCEL, EXIT, HELP, ITEM (bezieht sich auf ein Element des UIs), OK, SCREEN (bezieht sich auf den gesamten angezeigten Schirm) und STOP.

- Eine **Priorität**, die der MIDP Implementierung beispielsweise eine Auswahl direkt darzustellender Kommandos erleichtert, falls nicht alle angezeigt werden können, oder die Reihenfolge der Kommandos in einem Menü bestimmt. Kleinere Zahlen bedeuten größere Priorität.

Damit die Anwendung auf das Auslösen von Kommandos durch den Benutzer reagieren kann, muß sie einen CommandListener beim Displayable registrieren. Der Listener enthält die Methode commandAction(), die dann gerufen wird mit den Parametern command, der angibt, welches Kommando ausgelöst wurde, und displayable, der die Seite bestimmt, auf der dies geschah.

High Level UI

Das High Level UI enthält vordefinierte Elemente, die typischerweise in mobilen Anwendungen benutzt werden und sich auf allen Geräten darstellen lassen. Das konkrete

Look & Feel und das Bedienkonzept hängen dabei vom verwendeten mobilen Gerät ab. Dadurch wird erreicht, daß für den Benutzer eine einheitliche Darstellung und Bedienung möglich ist. Es sollten – soweit sinnvoll – diese vorgegebenen Elemente benutzt werden.

Alle Elemente sind von der abstrakten Klasse javax.microedition.lcdui.Screen abgeleitet. Diese erweitert ein Displayable um einen Titel und einen optionalen Ticker, der einen Text horizontal scrollend darstellt. Falls der Inhalt eines Screens nicht komplett auf den Bildschirm paßt, kann er vertikal gescrollt werden, niemals horizontal. Dieses Konzept bedeutet eine große Einschränkung gegenüber der Desktop-Metapher von Workstations und PCs. Deren in der Größe änderbare, frei verschiebbare und überlappende Fenster mit Menü- und Symbolleisten sind mit der begrenzten Leistungsfähigkeit der Zielgeräte jedoch nicht zu realisieren. Auch ist die dazu benötigte Multitaskingfähigkeit nicht vorhanden.

Konkrete Unterklassen von Screen sind:

TextBox: Ein Schirm zur mehrzeiligen Texteingabe. Dabei kann der erlaubte Zeichenvorrat durch sogenannte Constraints eingeschränkt werden. So können beispielsweise nur numerische Zeichen oder Zeichen für eine E-Mail Adresse zugelassen werden. Dies erleichtert gerade bei Verwendung einer einhändigen Tastatur die Eingabe, da viele Zeichen ausgeschlossen werden und somit das Multitap-Verfahren weniger Möglichkeiten berücksichtigen muß. Auch muß die Eingabe nicht hinterher auf Einhaltung dieser elementaren Bedingungen geprüft werden.

Statt einer normalen Eingabe kann der Typ auch auf Paßwort gesetzt werden, was bei den meisten Implementierungen zur Folge hat, daß eingegebene Zeichen nicht auf dem Bildschirm im Klartext erscheinen.

List: Stellt eine Liste von Auswahlmöglichkeiten auf dem Schirm dar. Diese kann entweder ein Checkbox-Verhalten besitzen, d. h. mehrere Elemente können gleichzeitig ausgewählt werden, ein Radiobutton-Verhalten, d. h. es kann jeweils nur ein Element ausgewählt sein, oder es kann eine implizite Liste sein. Bei letzterer führt die Auswahl eines Elementes dazu, daß ein Select Command erzeugt wird, auf das die Anwendung reagieren kann.

Alert: Ein Alarm zeigt eine Dialogbox mit Text und Bild auf dem Schirm. Er kann modal sein oder für eine bestimmte Zeitspanne eingeblendet werden. Als Alarmtypen stehen Alarm, Bestätigung, Fehler, Info und Warnung zur Verfügung. Auf den meisten Geräten werden je nach Typ unterschiedliche Piktogramme eingeblendet, auf einigen Geräten werden charakteristische Töne gespielt.

Einem Alarm können keine Commands hinzugefügt werden.

Form: Mit Formularen können mehrere Elemente gleichzeitig auf dem Schirm dargestellt werden. Diese werden von der abstrakten Klasse javax.microedition.lcdui.Item abgeleitet. Items können zu einem Formular hinzugefügt, ersetzt und gelöscht werden. Über das Layout der Elemente in einem Formular ist bis auf ihre Reihenfolge keine Kontrolle möglich.

Mögliche Items sind:

- Felder zur Eingabe von Datum und/oder Uhrzeit (DateField),
- Texte (StringItem),
- mehrzeilige Textfelder mit Filteroptionen (TextField; Beschreibung der Filteroptionen oben bei TextBox),
- Bilder im PNG-Format (ImageItem),
- interaktive und automatische Fortschrittsanzeigen (GaugeItem),
- Listen mit exklusiver oder multipler Auswahl (ChoiceGroup).

Für Änderungen an Items gibt es ebenfalls einen Listener, den ItemStateListener. Er enthält die Methode itemStateChanged(), die mit dem veränderten Item als Parameter aufgerufen wird. Dabei ist allerdings zu beachten, daß StringItem und ImageItem keine Events erzeugen und daß kleine Änderungen zusammengefaßt werden können.

Low Level UI

Für einige Aufgaben ist das High Level UI zu eingeschränkt. Beispielsweise wenn es um außergewöhnliche Bedienkonzepte, aufwendige grafische Darstellungen oder um Elemente mit spezieller Präsentation geht. Hier bietet sich das Low Level UI an. Bei diesem ist der Programmierer für die Darstellung verantwortlich.

Die Konzepte dahinter sind teilweise aus der J2SE entnommen. Als Zeichenfläche steht ein **Canvas** zur Verfügung, das eine festgelegte Höhe und Breite besitzt. Darauf kann die Bedienoberfläche gezeichnet werden. Zur Rückmeldung von Benutzeraktionen stehen neben Commands auch Ereignisse auf niedriger Ebene zur Verfügung. Gezeichnet wird mit vielfältigen Zeichenoperationen (Linien, Rechtecke, Bögen, Texte, Bilder, ...) auf ein **Graphics** Objekt.

Im konkreten Canvas Objekt ist die abstrakte Methode paint(Graphics g) der Oberklasse Canvas zu implementieren, in der das eigentliche Zeichnen geschieht. Diese wird durch die Umgebung aufgerufen, wenn ein (Neu)zeichnen nötig ist oder der Anwendungsprogrammierer dies verlangt. Für letzteres kann er die Methode repaint() aufrufen, wahlweise mit Angabe des gewünschten Bereiches. Das Neuzeichnen muß aber nicht sofort erfolgen; mehrere Aufrufe können zusammengefaßt werden. Erst wenn der Programmierer durch serviceRepaints() explizit ein Neuzeichnen verlangt, werden alle Zeichenwünsche abgearbeitet. Dieses Vorgehen dient der Effizienzsteigerung.

Analog zum High Level UI können Commands auch mit einem Canvas verknüpft werden. Die Implementierungen des MIDP auf den verschiedenen Geräten müssen hier einen Kompromiß zwischen leichter Aufrufbarkeit und Raum für die Zeichenfläche suchen.

Für mit Commands nicht zu realisierende Interaktionen stehen Low Level Events zur Verfügung. Es kann festgestellt werden, ob eine bestimmte Taste gedrückt und losgelassen wurde und ob evtl. mehrfache Tastendrücke erzeugt wurden, was nicht auf allen

Geräten möglich ist. Als Tasten werden dabei nur die in der ITU-T Tastatur vorhandenen vorausgesetzt (0 – 9, *, #), weitere werden nicht auf allen Geräten unterstützt. Als zusätzliches, abstrakteres Konzept können sogenannte Gamecodes abgefragt werden, die insbesondere für Spiele gedacht sind. Sie werden von der MIDP Implementierung auf spezielle Tasten abgebildet, so daß z. B. bei einigen Geräten zur Richtungssteuerung Steuerkreuze verwendet werden, bei anderen dagegen die Zifferntasten. Definierte Gamecodes sind UP, DOWN, LEFT, RIGHT, FIRE, GAME_A, GAME_B, GAME_C, GAME_D.

Bei Geräten, die dies unterstützen, erzeugen auch Interaktionen von Zeigegeräten, wie Stifte, Mäuse und Trackballs, Events. Es kann festgestellt werden, wenn Zeigegeräte gedrückt, losgelassen und gezogen werden. Dabei wird ihre Position auf dem Schirm durch x- und y-Koordinaten angegeben.

B.6 Record Management System

In mobilen Geräten finden sich eine Vielfalt an unterschiedlichen Speicherkonzepten. Einige besitzen beispielsweise ein Dateisystem mit Verzeichnissen und Dateien, während andere ihre Daten in einzelnen Datenbanken sichern, die Anwendungen zugeordnet sind. Die benötigte Zeit, um Daten zu sichern und zu laden, unterscheidet sich erheblich, ebenso der zur Verfügung stehende Speicherplatz.

In MIDP findet sich ein einheitliches Speichersystem, das Record Management System (RMS), das von den konkreten Systemen abstrahiert. Daten werden dabei in sogenannten RecordStores gesichert, die aus einzelnen Records bestehen, Datensätzen mit Index (RecordID) und entsprechenden Daten. RecordStores sind dabei MIDlet Suites zugeordnet. Jedes MIDlet einer Suite kann auf alle zugehörigen RecordStores zugreifen. Die Anzahl der möglichen RecordStores wird nur durch die Implementierung und den Speicherplatz des Gerätes begrenzt.

Records können hinzugefügt, abgefragt und gelöscht werden. Auf sie kann über ihren Index zugegriffen werden oder über einen RecordEnumerator. Vergleichbar mit Enumerations für Datenstrukturen des Collection-Frameworks der J2SE wird bei letzterem nacheinander jedes Element des RecordStores herausgegeben. Damit kann durch den RecordStore navigiert werden. Es stehen die Methoden nextRecord(), previousRecord(), hasNextElement(), hasPreviousElement() u. a. zur Verfügung.

Records in einem RecordEnumerator können sortiert werden. Dafür ist die Schnittstelle RecordComparator mit der Methode compare() zu implementieren. Die Funktion liefert -1, 0 oder 1 zurück, je nachdem, ob der erste übergebene Record kleiner als, gleich wie oder größer als der zweite ist. Ein RecordEnumerator kann auch gefiltert werden, um nicht benötigte Records eines RecordStores nicht zu berücksichtigen. Dafür dient das Interface RecordFilter mit der booleschen Methode matches().

B.7 Generic Connection Framework

Das Rahmenwerk, das in der J2SE für Netzwerkverbindungen implementiert ist, ist schwergewichtig. Es besteht aus über 100 Klassen mit ungefähr 200 KByte. Für die Konfiguration CLDC der J2ME ist das zu viel. Deshalb wurde eine einfachere Implementierung entwickelt: das Generic Connection Framework. Dieses sieht vor, daß eine Basisklasse, javax.microedition.io.Connector, mit statischen Methoden eine Vielzahl von Verbindungen herstellen kann. Dabei wird anhand der übergebenen Zeichenkette dynamisch eine passende Klasse gesucht. Das Format des Parameters ist {Schema}:[Ziel][Params]. Schema ist dabei der Protokollname (z. B. ftp), Ziel normalerweise die Netzwerkadresse und Params sind Parameter in Gleichungsform, getrennt durch Semikolons (z. B. ;antwort = 42).

Verbindungen werden über eine Hierarchie von Schnittstellen definiert. An ihrer Spitze steht Connection, das eine allgemeine Verbindung charakterisiert. Spezialisiert wird dieses Interface in weiteren wie InputConnection (zum Öffnen eines DataInputStream), OutputConnection (entsprechend für DataOutputStream), StreamConnection (faßt InputConnection und OutputConnection zusammen) und ContentConnection (stellt mehr Informationen über den Verbindungsinhalt bereit). Abbildung B.2 stellt die Hierarchie dar.

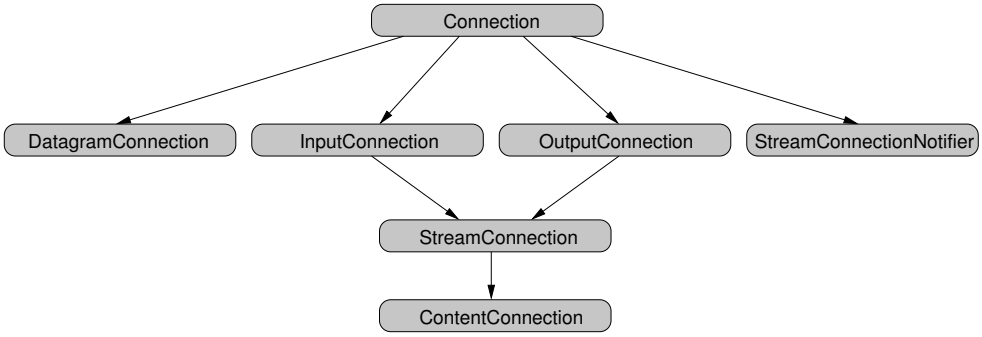


Abbildung B.2: Schnittstellenhierarchie im Generic Connection Framework

In MIDP muß nur das Protokoll http in der Version 1.1 implementiert sein. Es können von den MIDP Implementierungen aber weitere zur Verfügung gestellt werden. Die Netzwerkadresse wird dabei als Uniform Resource Locator (URL) angegeben. Die zugehörige Schnittstelle HttpConnection ist eine Subklasse von ContentConnection.

B.8 Ergänzende Bibliotheken, VMs, Werkzeuge

In die MIDP Spezifikation in der Version 1.0 wurden nur Konzepte aufgenommen, die auf allen Geräten implementiert werden können. Dies bedeutet, daß auf vielen Geräten spezifische Funktionen und erweiterte Möglichkeiten nicht über das API genutzt werden

können. Die Folge ist, daß für verschiedene Geräte Zusatzbibliotheken vom Hersteller oder von Drittanbietern existieren, die diese Bereiche abdecken.

Beispiele hierfür sind APIs für den Zugriff auf Adreßbuch und SMS für Mobiltelefone, APIs für hardwarenahe Programmierung für Spiele, die Implementierung des Abstract Windowing Toolkit der J2SE für Organizer oder XML Parser. Darunter leidet die Kompatibilität und es führt dazu, daß mehrere Versionen eines MIDlets je nach Zielgeräteplattform angeboten werden, falls proprietäre Bibliotheken benutzt werden.

Für MIDP stehen einige kompatible virtuelle Maschinen zur Verfügung, die die Konfiguration CLDC und die MIDP Bibliotheken unterstützen. Von Sun gibt es, wie schon in Abschnitt 6.3 ausgeführt, eine Referenzimplementierung für PalmOS, genannt KVM, und die Hotspot VM.

Für die Entwicklung von MIDlets gibt es Werkzeuge, die sich in bestehende Entwicklungsumgebungen einbinden lassen. Für die Kompilierung kann der Übersetzer der J2SE benutzt werden. Für das Preverifying ist ein eigenes Programm zuständig. Arbeit ersparen Tools, die das Manifest für die JAR-Datei und die JAD-Datei teilweise automatisch erzeugen.

Wichtig bei einer Crossplatform Entwicklung (Zielplattform entspricht nicht Entwicklungsplattform) sind Emulatoren und Simulatoren, da es meist sehr aufwendig ist, die MIDlets auf die Geräte zu übertragen und dort zu testen. Sie stehen für MIDP für verschiedene Geräte zur Verfügung. Die meisten erlauben die Anzeige von Metadaten über die Ausführung, ein Debugging und Leistungsmessungen. Sinnvoll kann auch der Einsatz sogenannter Obfuscator sein. Eigentlich dafür gedacht, ein Reverse Engineering von Java Programmen aus dem Bytecode zu verhindern, erweisen sie sich auch als nützlich, um den Umfang der Klassendateien zu verringern. Hier gibt es Beispiele von Einsparungen von bis zu 60%.

B.9 MIDP 2.0

Aufgrund der oben erwähnten Probleme mit zu wenig leistungsfähigen oder fehlenden Bibliotheken und aus anderen Gründen wurde eine Überarbeitung der MIDP Spezifikation nötig. Die Version 2.0, auch als MIDP Next Generation bezeichnet, liegt seit November 2002 als Final Release vor.[1] Neuerungen darin sind hauptsächlich:

- Ein verbessertes Sicherheitsmodell. Das Sandboxmodell von MIDP 1.0 wird grundlegend erweitert durch ein neues Sicherheitsrahmenwerk. Dieses unterstützt u. a. Übertragungen mit dem https Protokoll. Es gibt bei MIDlets eine Unterscheidung zwischen trusted und untrusted Code. Ersterer kann selbständig (unter Umständen kostenpflichtige) Verbindungen aufbauen, während bei letzterem beim ersten Zugriff oder individuell für jeden Zugriff der Benutzer eine Erlaubnis geben muß.

[1] Die Spezifikation ist unter `http://jcp.org/aboutJava/communityprocess/final/jsr118/` einsehbar.

- Die Bedienschnittstelle wird erweitert. Die eingeschränkte Funktionalität des alten LCDUI-APIs wird durch Unterstützung für vom Anwendungsprogrammierer entworfene Elementklassen und eine bessere Kontrolle des Formularlayouts erweitert. Elemente werden in Formularen in Reihen angeordnet. Sie verfügen über eine Ausrichtung und eine minimale und eine bevorzugte Größe, die bei der Plazierung berücksichtigt werden. Mit Platzhaltern kann das Layout weiter beeinflußt werden.

  Commands können nicht mehr nur Displayables hinzugefügt werden sondern auch einzelnen Items. Items besitzen auch ein Default Command, das mit einer festgelegten Eingabe des Benutzers aufgerufen wird.

- Das neue Game API, nicht zu verwechseln mit dem Game Profile für CDC, adressiert einen Haupteinsatzbereich von MIDlets: Spiele für mobile Geräte. Gerade für diese Programme wurden oft proprietäre APIs verwendet. Das Game API soll hier Abhilfe schaffen und stellt umfangreiche Neuerungen zur Verfügung. Auf verschiedenen Layern können Sprites angeordnet werden, die über Animationen verfügen und bewegt werden können. Mit Soundfunktionen können Melodien und Töne abgespielt werden.

  Auch für geeignete Geschäftsanwendungen könnte sich der Einsatz des Game APIs lohnen, um Sachverhalte grafisch zu veranschaulichen. Das Game API ist fester Bestandteil von MIDP 2.0.

Erste Geräte mit eingebauter MIDP 2.0 Unterstützung werden erst für Sommer 2003 erwartet.

Anhang C

Abkürzungsverzeichnis

ACM	Association for Computing Machinery
API	Application Programming Interface
AWT	Abstract Windowing Toolkit
B2B	Business to Business
B2C	Business to Consumer
BREW	Binary Runtime for Wireless
CDC	Connected Device Configuration
CLDC	Connected Limited Device Configuration
DRM	Digital Rights Management
EC	Exekutivkomitee
EJB	Enterprise JavaBeans
FK	Funktionskomponente
FTP	File Transfer Protocol
GIF	Graphics Interchange Format
GIS	Geographisches Informationssystem
GPS	Global Positioning System
GSM	Global System for Mobile Communication
GUI	Graphical User Interface
HCI	Human-Computer-Interaction
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IAF	Interaktionsformen
IAK	Interaktionskomponente
IEEE	Institute of Electrical and Electronic Engineers
ITU	International Telecommunication Union
J2EE	Java 2 Enterprise Edition
J2ME	Java 2 Micro Edition
J2SE	Java 2 Standard Edition
JAD	Java Application Descriptor
JAR	Java Archive
JCP	Java Community Process
JDK	Java Development Kit
JSML	Java Swing Markup Language

JSP	Java Server Pages
JSR	Java Specification Request
LBS	Location Based Services
LCDUI	Liquid Crystal Display User Interface
MCI	Mensch-Computer-Interaktion
MIDI	Musical Instrument Digital Interface
MIDP	Mobile Information Device Profile
ML	Markup Language
MMAPI	Mobile Media API
MMS	Multimedia Messaging Service
MPEG	Motion Pictures Expert Group
OCR	Optical Character Recognition
OS	Operating System
OTA	Over the Air Provisioning
PC	Personal Computer
PDA	Personal Digital Assistant
PF	Präsentationsformen
PIM	Personal Information Management
PMO	Program Management Office
PNG	Portable Network Graphics
POP3	Post Office Protocol 3
RI	Reference Implementation
RMI	Remote Method Invocation
RMS	Record Management System
ROM	Read Only Memory
SDK	Software Development Kit
SIM	Subscriber Identity Module
SMS	Short Message System
SMTP	Simple Message Transfer Protocol
TCK	Technology Compatibility Kit
UI	User Interface
URL	Uniform Resource Locator
VM	Virtuelle Maschine
WAM	Werkzeug – Automat – Material
WAP	Wireless Application Protocol
WMA	Wireless Messaging API
WML	Wireless Markup Language
XML	Extensible Markup Language
XP	Extreme Programming
XUL	XML User Interface Language

Literaturverzeichnis

[AbowdMynatt 00] G. Abowd, E. Mynatt: *Charting Past, Present, and Future Research in Ubiquitous Computing*, in: ACM Transactions on Computer-Human Interaction, Vol. 7, Nr. 1, 2000, S. 29ff

[Balas.Pierce 98] S. Balasubramaniam, B. Pierce: *What is a File Synchronizer?*, in: Proceedings of the ACM/IEEE MOBICOM 98 Conference, 1998, S. 98ff

[Beck 99] K. Beck: *Extreme Programming Explained: Embrace Change*, Addison-Wesley, Reading MA 1999

[Bleek et al. 99a] W.-G. Bleek, G. Gryczan, C. Lilienthal, M. Lippert, S. Roock, W. Strunk, H. Wolf, H. Züllighoven: *Frameworkbasierte Anwendungsentwicklung (Teil 2): Die Konstruktion interaktiver Anwendungen*, in: OBJEKTSpektrum 2/99, 1999, S. 78ff

[Bleek et al. 99b] W.-G. Bleek, C. Lilienthal, H. Züllighoven: *Frameworkbasierte Anwendungsentwicklung (Teil 4): Fachwerte*, in: OBJEKTspektrum 5/99, 1999, S. 75ff

[Bohlmann 00] H. Bohlmann: *Thin Clients in JWAM*, Diplomarbeit am Fachbereich Informatik, Universität Hamburg, 2000

[BrewsterCryer 99] S. Brewster, P. Cryer: *Maximising Screen-Space on Mobile Computing Devices*, in: Summary Proceedings of ACM CHI'99, ACM Press, New York 1999, S. 224f

[Buchanan et al. 01] G. Buchanan, S. Farrant, M. Jones, H. Thimbleby, G. Marsden, M. Pazzani: *Improving Mobile Internet Usability*, in: Proceedings WWW'10, Hong Kong 2001, S. 673ff

[DunlopCrossan 99] M. Dunlop, A. Crossan: *Dictionary based text entry method for mobile phones*, in: A. Brewster, M. Dunlop (Hrsg.): Proceedings of Second Workshop on Human Computer Interaction with Mobile Devices, 1999

[Floyd 93] C. Floyd: *STEPS - A Methodical Approach to PD*, in: Communications of the ACM 36(6), 1993

[FormanZahorjan 94] G. Forman, J. Zahorjan: *The Challenges of Mobile Computing*, in: IEEE Computer, Vol. 27, No. 4, 1994, S. 38ff

[Gamma et al. 95] E. Gamma et al.: *Design-Patterns – Elements of Reusable Object-Oriented Software*, Addison-Wesley, 1995

[Gomes et al. 01] P. Gomes, S. Tostão, D. Gonçalves, J. Jorge: *Web Clipping: Compression Heuristics for Displaying Text on a PDA*, in: M. Dunlop, S. Brewster (Hrsg.): Proceedings of Mobile HCI 2001: Third International Workshop on Human Computer Interaction with Mobile Devices, Lille, 2001

[Havenstein 00] A. Havenstein: *Unterstützung kooperativer Arbeit durch eine Software-Registratur*, Studienarbeit am Fachbereich Informatik, Universität Hamburg, 2000

[Hinckley et al. 00] K. Hinckley, J. Pierce, M. Sinclair, E. Horvitz: *Sensing Techniques for Mobile Interaction*, in: ACM UIST 2000 Symposium on User Interface Software and Technology, CHI Letters 2 (2), 2000, S. 91ff

[ISO9241-10 96] *Ergonomische Anforderungen für Bürotätigkeiten mit Bildschirmgeräten: Grundsätze der Dialoggestaltung (ISO 9241-10)*, Berlin, 1996

[JCP 01] *The Java Community Process Program – JCP Procedures – JCP2: Process Document*, `http://www.jcp.org/procedures/jcp2/`, Sun Microsystems, 2001

[JWAM 03] *JWAM Framework Website*, `http://www.jwam.de`, it-wps GmbH, 2003

[Kamba et al. 96] T. Kamba, S. Elson, T. Harpold, T. Stamper, P. Sukaviriya: *Using small screen space more efficiently*, in: Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI'96), ACM Press, New York 1996, S. 383ff

[Kleinrock 95] L. Kleinrock: *Nomadic Computing – An Opportunity*, in: ACM SIGCOMM, Computer Communications Review, Vol. 25, Nr. 1, ACM Press, New York 1995, S. 36ff

[Lippert 99] M. Lippert: *Die Desktop-Metapher in Systemen nach dem Werkzeug- und Material-Ansatz*, Diplomarbeit am Fachbereich Informatik, Universität Hamburg, 1999

[MacKenzie et al. 01] S. MacKenzie, H. Kober, D. Smith, T. Jones, E. Skepner: *LetterWise: Prefix-based Disambiguation for Mobile Text Input*, in: Proceedings of the 14th annual ACM symposium on User interface software and technology, 2001

[MacKenzieSoukoreff 02] S. MacKenzie, W. Soukoreff: *Text Entry for Mobile Computing: Models and Methods, Theory and Practice*, in: Human-Computer Interaction, 17, 2002, 147ff

[MacKenzieZang 99] S. MacKenzie, S. Zhang: *The Design and Evaluation of a High-Performance Soft Keyboard*, in: Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI'99), ACM Press, New York 1999

[Magerkurth 02] C. Magerkurth: *Entwicklung und Evaluation eines alternativen Texteingabesystems für Persönliche Digitale Assistenten*, in: M. Herczeg, W. Prinz, H. Oberquelle (Hrsg.): Mensch & Computer 2002: Vom interaktiven Werkzeug zu kooperativen Arbeits- und Lernwelten, B. G. Teubner, Stuttgart, 2002, S. 205ff

[Marcus 01] A. Marcus: *Babyface Design for Mobile Devices and the Web*, in: M. Smith, G. Salvendy (Hrsg.): Proceedings, Vol 2. Human-Computer Interface Internat. (HCII), New Orleans 2001, S. 514ff

[Marcus et al. 98] A. Marcus, J. Ferrante, T. Kinnunen, K. Kuutti, E. Sparre: *Baby Faces: User-Interface Design for Small Displays*, in: Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI'98), ACM Press, New York 1998

[MarcusChen 02] A. Marcus, E. Chen: *Designing the PDA of the Future*, in: interactions 9, Ausgabe 1, ACM Press, New York 2002, S. 35ff

[Masui 98] T. Masui: *An Efficient Text Input Method for Pen-based Computers*, Proceedings of the ACM Conference of Human Factors in Computing Systems (CHI'98), ACM press, New York 1998, S. 328ff

[Mattern 01] F. Mattern: *Pervasive/Ubiquitous Computing*, in: Informatik Spektrum 24, Ausgabe 3, Springer-Verlag Berlin Heidelberg, 2001, S. 145ff

[Muchow 02] J. Muchow: *CoreJ2ME Technology & MIDP*, Sun Microsystems Press, Palo Alto 2002

[Müller-Wilken 02] S. Müller-Wilken: *Mobile Geräte in verteilten Anwendungsumgebungen: Ein Integrationsansatz zwischen Abstraktion und Migration*, Dissertation am Fachbereich Informatik, Universität Hamburg, 2002

[Nilius 02] F. Nilius: *Anbindung und Kapselung existierender Anwendungssysteme*, Diplomarbeit am Fachbereich Informatik, Universität Hamburg, 2002

[Norman 98] D. A. Norman: *The Invisible Computer*, MIT Press, Cambridge 1998

[Oberquelle 99] H. Oberquelle: *Neue Herausforderungen und Wege für die Gestaltung von Information, Interaktion und Kooperation*, in: Information Management & Consulting, 14, 3, 1999, S. 19ff

[OttoSchuler 00] M. Otto, N. Schuler: *Fachliche Services: Geschäftslogik als Dienstleistung für verschiedene Benutzungsschnittstellen-Typen*, Diplomarbeit am Fachbereich Informatik, Universität Hamburg, 2000

[Palen et al. 00] L. Palen, M. Salzman, E. Youngs: *Going Wireless: Behavior & Practice of New Mobile Phone Users*, in: Proceedings of the 2000 ACM conference on Computer supported cooperative work, Philadelphia 2000, S. 201ff

[Palm 01] M. Dugger et al.: *Zen of Palm – Designing Products for Palm OS*, Palm, Inc., Document Number 3100-001, Santa Clara, 2001

[RhodesMcKeehan 01] N. Rhodes, J. McKeehan: *Palm OS Programming. The Developer's Guide*, O'Reilly, 2001

[RistBrandmeier 02] T. Rist, P. Brandmeier: *Customizing Graphics for Tiny Displays of Mobile Devices*, in: Personal and Ubiquitous Computing 6 (2002) 4, Springer-Verlag, London, S. 260ff

[Rosenberg 98] R. Rosenberg: *Computing without Mice and Keyboards: Text and Graphic Input Devices for Mobile Computing*, Dissertation, Dept. of Computer Science, University College, London 1998

[Roth 02a] J. Roth: *Mobile Computing – Grundlagen, Technik, Konzepte*, dpunkt.verlag, Heidelberg, 2002

[Roth 02b] J. Roth: *Patterns of Mobile Interaction*, in: Personal and Ubiquitous Computing (2002) 6, Springer-Verlag, London, 2002, S. 282ff

[Sauer 01] J. Sauer: *Generierung von Fachwerten aus XML-Beschreibungen*, Studienarbeit am Fachbereich Informatik, Universität Hamburg, 2001

[Shneiderman 97] B. Shneiderman: *Designing the User Interface*, Dritte Auflage, Addison-Wesley, 1997

[Spolsky 01] J. Spolsky: *User Interface Design for Programmers*, APress, 2001

[StajanoJones 98] F. Stajano, A. Jones: *The Thinnest Of Clients: Controlling It All Via Cellphone*, in: ACM Mobile Computing and Communications Review, ACM Press, New York 1998, S. 46ff

[Sun 00] *Applications for Mobile Information Devices – Helpful Hints for Application Developers and User Interface Designers using the Mobile Information Device Profile*, White Paper, Palo Alto 2000

[Weiser 91] *The Computer for the 21st Century*, Scientific American 265 (3), 1991, S. 94ff

[XUL 01] D. Hyatt (Hrsg.): *XML User Interface Language (XUL) 1.0*, `http://www.mozilla.org/projects/xul/xul.html`, Mozilla.org, 2001

[Zuberec 00] S. Zuberec: *The Interaction Design of Microsoft Windows CE*, in: E. Bergman (Hrsg.): Information Appliances and Beyond – Interaction Design for Consumer Products, Morgan Kaufmann, San Francisco 2000, S. 103ff

[Züllighoven 98] H. Züllighoven: *Das objektorientierte Konstruktionshandbuch*, dpunkt.verlag, 1998

Erklärung:

Hiermit versichere ich, diese Arbeit selbständig und unter ausschließlicher Zuhilfenahme der in der Arbeit aufgeführten Quellen und Hilfsmittel erstellt zu haben.

Hamburg, den 21. Juli 2003

Joachim Sauer
Grot Sahl 72
D-22559 Hamburg
Tel. (040) 81 17 80