

Universität Stuttgart

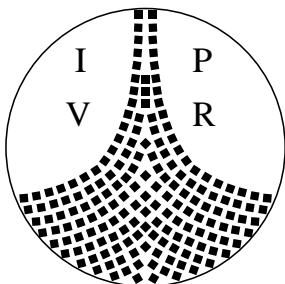
Fakultät Informatik

Prüfer: Prof. Dr. B. Mitschang
Betreuer: Dipl. inf. Daniela Nicklas
begonnen am: 24.6.2000
beendet am: 27.12.2000
CR-Klassifikation: D.2.11, F.2.2, H.2.m.

Studienarbeit Nr. 1794

Konzeption und Entwicklung einer mobilen Datenbankanwendung

Stjepan Grzan



Institut für Parallele und Verteilte
Höchstleistungsrechner (IPVR)
Abteilung Anwendersoftware
Breitwiesenstr. 20-22
70565 Stuttgart

AS

Kurzfassung

Systeme, die Informationen abhängig von momentanen Aufenthaltsort ausgeben, werden durch die Verbreitung mobiler Computer immer wichtiger. Das Nexusprojekt untersucht die Konzepte und Möglichkeiten für die Unterstützung solcher ortsabhängigen Systeme. Das Ziel ist die Entwicklung einer Plattform, die Dienste für ortsabhängige Systeme anbietet.

In dieser Arbeit wird die Entwicklung einer Beispielanwendung für das Nexussystem beschrieben. Diese Anwendung greift dabei noch nicht auf Nexusdienste zu, sondern implementiert die Nexusdienste selbst und greift auf Daten zu, die in einer mobilen Datenbank gespeichert werden. Die Beispielanwendung wird auf einem palmkompatiblen Gerät ausgeführt. Diese Geräte unterscheiden sich durch ihren Einsatzzweck von Laptops durch ihre Größe und Leistungsfähigkeit. Die Datenbank DB2 Eveyplace ist deshalb im Vergleich zu Datenbanken für stationäre Systeme in ihrer Funktionalität eingeschränkt. Als Vorgehensmodell für die Entwicklung wurde ein Phasenmodell mit der Entwicklung von Prototypen eingesetzt. Die Erweiterbarkeit und Bedienbarkeit des Programms müssen hoch sein und die Datenbank sollte mit dem Programm bewertbar sein. Als Ausgangspunkt für die Architektur diente die Nexusplattform, wobei Komponenten zusammengefasst wurden. Der Entwurf der Karten musste die Größe des Palmbildschirms berücksichtigen. Es zeigte sich weiterhin, dass die räumliche Anfrage an die Bedürfnisse des Anwenders angepasst werden musste. Bei der Evaluation der Anwendung stellte sich heraus, dass die Datenbank Probleme mit der Datensicherheit und der Geschwindigkeit der Anfragen hatte.

Inhaltsverzeichnis

1	Einleitung	1
1.1	Motivation	1
1.2	Nexus	1
1.3	Aufgabenstellung	2
1.4	Überblick über die Arbeit	2
2	Technisches Umfeld	3
2.1	Mobile Rechnersysteme	3
2.2	Palmkompatible PDAs	6
2.3	Entwicklung von Palmprogrammen	8
2.4	Programmierung für Palms	11
2.5	DB2 Everyplace	12
2.6	Eingesetzte Entwicklungswerkzeuge	15
2.7	Zusammenfassung	16
3	Das Vorgehen	17
3.1	Anforderungen	17
3.2	Das Vorgehensmodell	17
3.3	Der konkrete Ablauf	17
4	Die Anforderungen an das Programm	19
4.1	Einsatzgebiet des Programms	19
4.2	Zielgruppen der Arbeit	19
4.3	Die Anforderungen	20
4.4	Use Cases	22
4.5	Funktionale Anforderungen	24
4.6	Anforderungen an die Daten	26
4.7	Zusammenfassung der Anforderungen	28
5	Programmbeschreibung	29
5.1	Übersicht über die Fenster	29
5.2	Einschränkungen des Prototypen	34
6	Der Architekturfentwurf	35
6.1	Anforderungen an die Architektur	35
6.2	Die betrachteten Architekturen	35
6.3	Die Architektur des Programms	43
6.4	Aufbau der Datenbank	46
6.5	Die Verteilung der Daten	47
6.6	Mögliche zukünftige Erweiterungen	49
6.7	Zusammenfassung	51
7	Kartenentwurf	53
7.1	Karten	53
7.2	Das Kartenformat	60
7.3	Zuordnung von Kartenobjekten zu Sachdaten	61
7.4	Die Navigation auf den Detailkarten	63
7.5	Zusammenfassung	64
8	Realisierung der räumlichen Anfrage	65
8.1	Semantik: nächster, nearest, next	65
8.2	Ausgabe der find nearest Funktion	66
8.3	Die Berechnung der Entfernung	67

8.4 find nearest für die Ausgabe von einzelnen Mitarbeitern	68
8.5 find nearest für mehrere Mitarbeiter.....	71
9 Tabellen und Anfragen	77
9.1 Tabellen der Datenbank	77
9.2 Die Anfragen an die Datenbank	78
9.3 Zusammenfassung	82
10 Evaluation der Anwendung	83
10.1 Bewertung der Datenbank.....	83
10.2 Optimierungsmöglichkeiten.....	87
10.3 Bewertung der Anwendung.....	96
10.4 Zusammenfassung	97
11 Zusammenfassung und Ausblick	99
12 Literaturverzeichnis	a

Tabellenverzeichnis

Tabelle 2-1: Technische Daten des Visors	7
Tabelle 10-1: Tabellen der Datenbank.....	85
Tabelle 10-2: Statistik zu den SQL Befehlen.....	86
Tabelle 10-3: Ausführungsdauer	87
Tabelle 10-4: Ausführungsdauer bei Indexen	90
Tabelle 10-5: Vergleich der Geschwindigkeit	92

Abbildungsverzeichnis

Abbildung 2-1: Aufbau der Synchronisationsarchitektur	14
Abbildung 4-1: Darstellung der Navigation.....	22
Abbildung 4-2: Klassendiagramm der Daten.....	27
Abbildung 5-1: Die Übersichtskarte.....	29
Abbildung 5-2: Die Detailkarte.....	30
Abbildung 5-3: Das Fenster für global find und find nearest	31
Abbildung 5-4: Fenster für die Mitarbeiter- und Raumdaten.....	32
Abbildung 5-5: Das Fenster für die SQL-Anfragen	33
Abbildung 5-6: Fenster für die Einstellungen	33
Abbildung 6-1: Nexusarchitektur.....	37
Abbildung 6-2: Nexus-Client-Architektur	38
Abbildung 6-3: Nexus-Server Architektur.....	38
Abbildung 6-4: ViLiS-Architektur.....	41
Abbildung 6-5: ViLiS Client Architektur	42
Abbildung 6-6: Programm-Architektur	43
Abbildung 6-7: Aufbau der Datenbank.....	46
Abbildung 6-8: Einfügen einer neuen Datenbank	49
Abbildung 7-1: Eine vereinfachte Karte	55
Abbildung 7-2: Probleme bei der Positionierung	56
Abbildung 7-3: Der Aufbau der Karten.....	63
Abbildung 8-1: Ablauf von find nearest.....	69
Abbildung 8-2: Beispiel für eine nicht optimale Wegwahl.....	70

1 Einleitung

1.1 Motivation

Die Miniaturisierung ermöglichte es Computer so klein zu machen, dass sie von Benutzern bei sich getragen werden können. Hierdurch können die Benutzer ihre Geräte mitnehmen und sind so auch in der Lage ihre Rechner an unterschiedlichen Orten zu benutzen. Es entstanden Anwendungen, die speziell für mobile Benutzer ausgelegt waren. Eine große Gruppe bilden hierbei die ortsabhängigen Systeme, die Informationen abhängig vom momentanen Aufenthaltsort ausgeben. Typische Vertreter dieser Anwendungen sind z.B. Navigationssysteme, die mit Hilfe von Sensoren ihre Position bestimmen. Parallel zu dieser Entwicklung gab es durch den Handyboom eine starke Ausbreitung von Funknetzen. Hierdurch ist es den Benutzern möglich, auf Informationen und Dienste anderer Systeme zuzugreifen. Die Informationen und Dienste können einen Ortsbezug haben und ein Zugriff auf diese könnte über räumliche Anfragen erfolgen. Beispielsweise könnte ein solcher Dienst Karten anbieten. Benötigt ein mobiler Benutzer eine Karte zu einem Gebiet, so sendet er an diesen Dienst seine Position und erhält dann die Karte zu dem Gebiet, die dann von seinem mobilen Gerät angezeigt wird. Mit anderen Diensten könnte der Benutzer dann vielleicht erfahren, wo eine bestimmte Sehenswürdigkeit ist, wo das nächste Restaurant ist oder an welchen Haltestellen die Strassenbahn, in der sich der Benutzer befindet, noch halten wird. Die Dienste müssen allerdings nicht nur für große Gebiete nützlich sein. Museen könnten einen Dienst anbieten durch den ein mobiles Gerät in der Lage wäre Informationen zu einem ausgestellten Objekt anzuzeigen, sobald der Benutzer in der Nähe des Objektes ist und in dessen Richtung sieht. Hierfür müsste das Gerät Angaben zur Position und Richtung, in die es gehalten wird, zum Dienst schicken. Der Dienst ermittelt dann, zu welchem Objekt Informationen verlangt werden, und schickt diese an das mobile Gerät.

1.2 Nexus

Die Nexus-Forschungsgruppe untersucht die Konzepte und Möglichkeiten für die Unterstützung ortsabhängiger Systeme. Das Ziel ist dabei die Entwicklung einer Plattform, die Dienste zur Unterstützung ortsabhängiger Systeme anbietet. Diese Plattform verwaltet dabei ein Modell der realen Welt, das um virtuelle Objekte erweitert wird. Diese virtuellen Objekte können z.B. virtuelle Litfasssäulen sein, die einer geographischen Position zugeordnet sind und Informationen enthalten. Man wird diese virtuellen Objekte auch mit realen Objekten verknüpfen können. Beispielsweise könnte man ein Auto mit einem virtuellen Objekt verknüpfen, das Daten zu dessen Wartung enthält. Hiermit könnten Mechaniker beim Überprüfen des Autos sehr schnell feststellen, was alles gewartet werden muss.

Es wird dabei ein grundlegender Kommunikationsdienst benötigt, der die unterschiedlichen Systeme für die mobile Kommunikation unterstützt und auch die Fähigkeit einer geographischen Adressierung besitzt. Die Plattform erhält Informationen über die Position und die Umwelt des Benutzers durch unterschiedliche Sensorsysteme wie z.B. GPS oder Bildererkennung. Die Informationen werden dabei

in einem verteilten System gespeichert. Die effiziente Speicherung und Verarbeitung der Informationen ist ein weiterer wichtiger Forschungsbereich.

1.3 Aufgabenstellung

Im Rahmen dieser Studienarbeit sollen die Möglichkeiten erforscht werden, mobile Datenbankanwendungen mithilfe von Personal Digital Assistants (PDAs, z.B. Visor) zu realisieren.

Als Beispielanwendung soll ein System entstehen, das aufgrund von Lokationsinformation durch den Benutzer (Stifteingabe auf dem Gebäudeplan der Informatik) räumliche Anfragen an eine Beispieldatenbank erlaubt. Die Datenbank enthält Mitarbeiter im Haus, deren Position und deren Aufgabengebiete. Dadurch soll das System in der Lage sein, den räumlich nächsten Mitarbeiter zu einem bestimmten Aufgabengebiet zu finden.

Arbeitsschritte sind dabei:

- Anforderungsanalyse für die Beispielanwendung
- Spezifikation, Entwurf und Implementierung der Beispielanwendung und der dazugehörigen Datenbank auf einem Visor mit DB2 Everyplace
- Evaluation der Beispielanwendung

1.4 Überblick über die Arbeit

Kapitel 2 beschreibt das technische Umfeld der Studienarbeit. Kapitel 3 beschreibt das Vorgehen bei der Entwicklung der Beispielanwendung. Die Anforderungen an das Programm werden in Kapitel 4 beschrieben. Kapitel 5 beschreibt das Aussehen des Programms. Kapitel 6 bis 8 beschreiben den Entwurf des Programms. Kapitel 6 beschreibt dabei den Architekturentwurf, Kapitel 7 den Kartenentwurf und Kapitel 8 zeigt die Realisierung der räumlichen Anfrage. Für eine Evaluation des Programms benötigt man die Tabellen der Datenbank und die Anfragen an die Datenbank. Diese werden in Kapitel 9 vorgestellt. Kapitel 10 evaluiert die Anwendung. Kapitel 11 fasst die Ergebnisse der Studienarbeit zusammen und gibt Ausblick auf mögliche Erweiterungen des Programms.

2 Technisches Umfeld

Zuerst wird eine Einteilung mobiler Rechner gemacht und die Gruppe vorgestellt, zu denen Palms gehören. Danach wird die Programmentwicklung für diese Geräte beschrieben und die Datenbank DB2 Everyplace vorgestellt. Zum Schluss werden die eingesetzten Entwicklungswerkzeuge beschrieben.

2.1 Mobile Rechnersysteme

Mobile Computer unterscheiden sich in ihrer Zielgruppe, Leistungsfähigkeit, Aussehen und Mobilität. Dieses führt zu Unterschieden in ihrem Einsatz. Eine mögliche Einteilung ergibt sich durch den Einsatzzweck der Systeme. Hierdurch ergeben sich drei Gruppen: Mobile Computer, die Desktop Computer ersetzen sollen; mobile Computer, die als Erweiterung der Desktop Computer dienen; und sonstige Systeme, deren Einsatzzweck auf die Realisierung von Funktionen abzielt, die nicht durch stationäre Computer abgedeckt werden. Desktop Computer bedeutet hierbei ein herkömmlicher stationärer PC. Die Übergänge zwischen den Gruppen sind fließend. Beispielsweise werden Handys mittlerweile um Programme wie Terminplaner erweitert und beherrschen Datenaustausch mit Computern.

Diese Gruppen werden im folgenden vorgestellt. Hierbei wird der Einsatzzweck, die Hardwareeigenschaften und deren Anwendungsentwicklung mit denen bei Desktopsystemen verglichen.

2.1.1 Mobile Computer als Ersatz für andere Systeme

Auf diesen Computern, die man Laptops oder Notebooks nennt, werden typische Computeranwendungen ausgeführt (Textverarbeitung, Tabellenkalkulation, Präsentationsprogramme etc.). Sie besitzen ähnlich viel Speicher und Rechenleistung wie herkömmliche Desktopsysteme und sind kompatibel zu diesen. Für die Ausführung der Anwendungen wird wie bei Desktopsystemen ein großer Bildschirm und eine gute Tastatur benötigt. Man findet auch die herkömmlichen Komponenten für Datenaustausch und der Speicherung der Daten in diesen Systemen. Hierdurch wird die Mobilität eingeschränkt, da der Stromverbrauch dieser Geräte so hoch ist, dass man nur wenige Stunden mit ihnen arbeiten kann, wenn sie nicht an eine Stromquelle angeschlossen sind. Eine weitere Beeinträchtigung der Mobilität ergibt sich durch die Unhandlichkeit dieser Geräte. Da sie einen großen Bildschirm und eine Tastatur benötigen, müssen diese relativ groß sein. Ein weiteres Problem ist, dass durch die große Leistungsfähigkeit der Geräte eine Miniaturisierung der eingebauten Komponenten sehr aufwendig ist und die Systeme dadurch mehrere Kilogramm wiegen können. Sie benötigen für ihre Benutzung deshalb eine stabile Unterlage auf die man sie ablegen kann.

Ein solcher Computer wird meist im ausgeschalteten Zustand bewegt. Für die Benutzung legt man ihn auf eine stabile Unterlage. Da auf ihm herkömmliche Desktopanwendungen benutzt werden, wird er dann auch eine längere Zeit genutzt.

Die Anwendungsentwicklung entspricht der Anwendungsentwicklung für Desktopsysteme. Die Entwicklungswerkzeuge sind die gleichen, wie bei den Desktopsystemen und sind leicht zu erhalten. Den Anwendungen steht viel Rechenleistung und Speicher zur Verfügung. Man muss diese Anwendungen deshalb meist nicht optimieren.

2.1.2 Mobile Computer als Ergänzung zum Desktop

Diese Computer wollen die Desktopsysteme nicht ersetzen, sondern versuchen mit diesen eine Symbiose einzugehen. Der Desktop Computer dient zur Eingabe und Verarbeitung größerer Datenmengen. Der mobile Computer ermöglicht dem Benutzer diese Daten mit sich zu führen. Die mobilen Geräte sollen in erster Linie dazu Daten darstellen und kleine Berechnungen oder Eingaben ermöglichen.

Der Benutzer lädt hierfür die Daten auf sein mobiles Gerät. Möchte der Benutzer wieder mit dem Desktop arbeiten, so werden die veränderten Daten vom mobilen Gerät wieder zum Desktop übertragen. Dies hat große Vorteile gegenüber Systemen, die als Ersatz für den Desktop dienen. Solche mobilen Geräte benötigen nur noch einen Bruchteil der Leistungsfähigkeit der Desktopsysteme. Weiterhin können ihre Maße durch Verkleinerung des Displays und Verkleinerung bzw. Weglassen einer Tastatur beträchtlich schrumpfen, da der Benutzer nur wenige Daten eingeben möchte und eine Anzeige auf einem kleinen Display akzeptabel ist. Durch die Reduktion der Leistungsfähigkeit und der Bildschirmmaße wird der Energieverbrauch der Geräte stark verringert. Die Geräte werden durch diese Maßnahmen im Vergleich zu Laptops sehr leicht. Dies führt zu einer verbesserten Transportfähigkeit, da der Benutzer sie in den Taschen seiner Kleidung transportieren kann. Sie können auch ohne Anschluss an ein Stromnetz längere Zeit betrieben werden. Die kleinen Geräte können in einer Hand gehalten werden, wodurch man diese Geräte auch ohne stabile Unterlage benutzen kann. So kann man auch bei der Bedienung des Gerätes mobil sein.

Die Nachteile der geringen Leistungsfähigkeit und der verschlechterten Eingabemöglichkeiten führen dazu, dass diese Geräte in erster Linie zur Anzeige der Daten und zur Veränderung kleiner Datenmengen benutzt werden können. Das ist für viele Benutzer akzeptabel, da sie rechenintensive oder eingabeaufwendige Aufgaben nur auf ihrem Desktopsystem durchführen.

Typische Vertreter dieser Art Computer sind Personal Digital Assistants (PDA). Sie können Termine und Personendaten verwalten. Man kann auch Notizen auf diesen Systemen machen und später auf den Desktop kopieren.

Manche PDAs können ihre Funktionalität um neue Programme erweitern. Ein Beispiel für solche PDAs sind palmkompatible Geräte. Diese werden in 2.2 "Palmkompatible PDAs" vorgestellt.

Die Anwendungsentwicklung für diese Systeme unterscheidet sich von der Anwendungsentwicklung für Desktopsysteme. Es wird normalerweise nicht auf den Geräten programmiert, sondern auf Desktopsystemen. Die Verfügbarkeit und Anzahl der Entwicklungswerkzeuge ist geringer als bei Desktopsystemen. Auch die dazu verfügbare Literatur ist sehr viel geringer. Die Anwendungen

haben sehr viel weniger Speicher und Rechenleistung zur Verfügung, wodurch Optimierungen des Programms sehr wahrscheinlich sind. Durch die kleineren Bildschirmmaße lassen sich nur wenige Bedienelemente darstellen. Die Erstellung der Benutzungsoberfläche ist dadurch schwieriger als bei Desktopsystemen. Wenn man viele Funktionen in ein Programm einbauen will, ist die Gefahr groß, eine umständliche Bedienung des Programms zu erhalten. Die Entwicklung von Palmprogrammen wird in 2.3 "Entwicklung von Palmprogrammen" genauer dargestellt.

2.1.3 Sonstige mobile Computer

Die bisher betrachteten Gruppen haben gemeinsam, dass der Benutzer die mobilen Geräte für den Einsatz von Computeranwendungen benutzt. Es gibt allerdings auch mobile Geräte mit Prozessoren mit denen man keine typischen Computeranwendungen macht. Der bekannteste Vertreter dieser mobilen Computer ist das Handy. Dieses Gerät wird in erster Linie zum Telefonieren eingesetzt und besitzt hierfür einen oder mehrere Prozessoren, die für die Kommunikation und die Ansteuerung des Displays notwendig sind.

Eine andere Gruppe soll Bücher ersetzen. Mit diesen sogenannten E-Books kann der Benutzer wie bei Büchern Texte überallhin mitnehmen und lesen.

Digitaluhren gehören ebenfalls nicht zu den oberen beiden Gruppen. Sie besitzen allerdings einen Computerchip und einige Funktionen.

Eine weitere Gruppe befindet sich unter den sogenannten Embedded Systems. Diese Systeme werden oft zur Steuerung oder zum Betrieb von Geräten eingesetzt. Beim Einsatz in mobilen Geräten werden diese zu mobilen Rechnern. Beispielsweise werden in heutigen Autos vielerlei Prozessoren zur Steuerung mechanischer Systeme eingesetzt. Dies geschieht ohne das Anwender überhaupt Kenntnis von diesen Systemen haben müssen.

Aufgrund ihres unterschiedlichen Einsatzzweckes kann man erahnen, dass diese Systeme sehr unterschiedliche Eigenschaften haben.

Wie für PDAs gilt, dass die Entwicklung sich von der Anwendungsentwicklung auf einem Desktop unterscheidet. Die Verfügbarkeit und Anzahl der Entwicklungswerkzeuge ist für viele dieser Systeme gering. Die Literatur ist kaum vorhanden, da bei vielen dieser Systeme eine Erweiterbarkeit durch neue Programme nicht vorgesehen ist. Die Anwendungen auf diesen Systemen haben oft noch weniger Speicher und Rechenleistung zur Verfügung wie PDAs.

2.1.4 Eignung der Systeme für die Aufgabe

Aus der Aufgabenstellung kann man entnehmen, dass ein palmkompatibles Gerät verwendet werden soll. Es gehört zu den mobilen Computern, die Desktopsysteme ergänzen sollen.

Die Entwicklung für einen Laptop hätte eine einfachere Anwendungsentwicklung zur Folge gehabt, allerdings sind Laptops für einen Einsatz als Navigationshilfsmittel in einem Gebäude weniger geeignet. Ihre Betriebsdauer ohne Stroman-

schluss ist kurz. Sie sind sehr unhandlich in der Bedienung, da die Eingabegeräte und die Programme von einem stationären Benutzer ausgehen, was für diese Anwendung nicht richtig ist. Möchte der Benutzer den Laptop auf dem Arm tragen, stört dabei das Gewicht und der Aufbau des Gerätes.

Der Vorteil dieser Geräte gegenüber den anderen Systemen besteht in ihrer Rechenleistung, dem zur Verfügung stehenden Speicher und der Anwendungsentwicklung. Die Entwicklung für diese Systeme ist einfacher als für die anderen Systeme.

Die sonstigen Systeme sind ungeeignet, da die Entwicklungswerkzeuge und die notwendige Literatur nicht leicht zu erhalten sind. Die Leistungsfähigkeit der Prozessoren ist z.B. bei Handys noch geringer als sie bei PDAs ist. Ein weiteres Problem stellt die Bedienung der Geräte dar. Diese ist für einen völlig anderen Zweck ausgelegt. Diese Eigenschaften machen diese Systeme ungeeignet für die Studienarbeit.

Die momentan verbreitetste PDA Plattform sind Systeme die auf Palm OS basieren. Sie besitzen Literatur und Entwicklungswerkzeuge für die Entwicklung von Palmprogrammen. Im folgenden Kapitel werden Palms genauer vorgestellt, damit der Leser eine Vorstellung bekommt, welche Eigenschaften ein Palm hat und wie eine Entwicklung für diese Systeme aussieht.

2.2 Palmkompatible PDAs

2.2.1 Beschreibung

Palms besitzen einen berührungssensitiven Bildschirm auf dem man mit Hilfe eines Stiftes klicken kann. Die Texteingabe auf einen Palm erfolgt nicht durch eine Tastatur, sondern durch das Zeichnen von "Graffiti" in einem Feld am unteren Bereich des Geräts. "Graffiti" ist ein Alphabet, das aus vereinfachten Buchstaben und Zahlen besteht. Diese Form der Texteingabe hat Vorteile gegenüber der Eingabe durch eine Tastatur oder der Eingabe durch normale Buchstaben. Eine Tastatur hätte das Gerät stark vergrößert. Die Eingabe normaler Buchstaben ist sehr viel fehleranfälliger als Graffiti, da handgeschriebene Buchstaben nicht so einfach zu erkennen sind.

Palms besitzen im Vergleich zu Desktopsystemen einen sehr langsamen Prozessor. Dieser ist auf Stromsparen optimiert.

Palms besitzen mehrere Standardprogramme z.B. Terminkalender und ein Adressbuch. Es lassen sich weitere Programme auf einen Palm laden. Da viele dieser Programme Daten von Desktopprogrammen darstellen und bearbeiten können, lassen sich diese Daten mit den Daten auf dem Desktop austauschen. Hierbei werden die Daten vom Desktop zum Palm und vom Palm zum Desktop kopiert und eventuelle Inkonsistenzen behoben. Dabei müssen die Daten evtl. auch konvertiert werden, da sie auf dem Palm in einem anderen Format als auf dem Desktop gespeichert sein können.

Palms haben mehrere Möglichkeiten, wie sie Daten austauschen können. Die erste Möglichkeit ist eine Infrarotschnittstelle, die zum Datenaustausch zwischen Palmgeräten dient. Die zweite Möglichkeit ist mit Hilfe der "Cradle". Diese ist ein Gerät mit der ein Datenaustausch zwischen Desktopsystem und dem Palm erfolgen kann. Manche Palms haben die Möglichkeit über TCP/IP zu kommunizieren.

Palms gibt es seit Mitte 1996. Der erste Palm war der Pilot 1000. Er hatte 128 KB Speicher und konnte bereits damals neue Programme laden. Er hatte die typischen PDA Programme. Diese sind ein Adressbuch ein Terminkalender, eine TO DO Liste, ein Memo Pad, ein Taschenrechner und eine Anwendung für Sicherheit.

Die heutigen Palms haben im Vergleich zum ersten Palm viel mehr Speicher (2-8 MB). Viele andere Eigenschaften sind aber in etwa gleich geblieben. Die heutigen Geräte haben in etwa die gleichen Baumaße wie der erste Palm. Auch hat sich die Texteingabe nicht verändert. Die Geschwindigkeit der heutigen Palms ist nicht viel grösser geworden. Es gibt mittlerweile Modelle wie den Palm IIIc die einen farbigen Bildschirm haben.

Für die Studienarbeit wurde ein palmkompatibler Rechner der Firma Handspring benutzt. Im folgenden Abschnitt wird dieser vorgestellt. Andere Palmsysteme unterscheiden sich in erster Linie durch den Speicherausbau.

2.2.2 Der Handspring Visor Deluxe

Allgemeine Beschreibung

Die Daten der Tabelle 2-1 entstammen aus [Handspring 2000].

Tabelle 2-1: Technische Daten des Visors

Element	Visor
Palm OS	3.1 H
Speicher	8 MB
Abmessungen (H, B, T)	12,2 cm x 7,6 cm x 1,8 cm
Gewicht	153,1 g
Bildschirmauflösung	160 x 160 Pixel
Farbdarstellung	schwarz/weiss
Bildschirmgröße	6 cm x 6 cm
Nutzungsdauer der Batterien	ca. 2 Monate
Infrarotschnittstelle	ja

Eine Besonderheit des Handspring Visor Deluxe gegenüber anderen Palmgeräten ist seine Erweiterbarkeit. Mit Hilfe sogenannter Springboard Module kann man

den PDA um Hardware wie GPS Sensoren, zusätzlichen Speicher, Scanner, Kommunikationsmodule und vielem mehr erweitern.

Der Prozessor

Der Prozessor des Gerätes ist ein Motorola MC68328 Dragonball mit 16 MHz Taktfrequenz. Dieser ist kompatibel zum MC68000 Prozessor und ist ein 32-Bit CISC Prozessor. Er besitzt keinen Cache und auch keine Fließkommaeinheit. Auf dem Chip integriert sind Module für die Ansteuerung anderer Komponenten und für das Powermanagement. Diese Komponenten sind das LCD Display, die Serielle Schnittstelle, Soundausgabe, Echtzeituhr und weitere Komponenten. Der Stromverbrauch des Prozessors wird durch mehrere Methoden reduziert. Er kann mit 5 V oder mit 3,3 V arbeiten und wurde in stromsparender HCMOS Technologie gefertigt. Er kann zur Laufzeit seinen Prozessortakt ändern und ist in der Lage die Module für die Ansteuerung der Komponenten abzuschalten. Weiterhin kann die Prozessoruhr (diese bestimmt den Prozessortakt) angehalten werden.

Im Vergleich mit heutigen (Dezember 2000) Desktopprozessoren fällt die niedrigere Taktrate auf. Desktopprozessoren werden zwischen 500-1400 MHz getaktet und sind damit zwischen 31 und 88 mal höher getaktet. Der Geschwindigkeitsunterschied dürfte allerdings noch höher ausfallen, da die Desktopprozessoren mehr Transistoren besitzen und die Anweisungen dadurch effizienter implementieren können. Das zweite Auffällige ist die Integration vieler Ansteuerungskomponenten. Diese sind nicht in Desktopprozessoren integriert. Diese Integration führt zu einer Platzeinsparung und zu einer billigeren Produktion der Handheldgeräte. In heutigen PCs werden viele dieser Funktionen, wie z.B. die Grafikausgabe, durch Zusatzchips erledigt, wodurch die Prozessoren entlastet werden. Der dritte Unterschied besteht in den Stromsparmaßnahmen. Desktopprozessoren werden in erster Linie auf Rechenleistung optimiert und nicht auf die Reduktion des Stromverbrauchs, da den Benutzern der Stromverbrauch nicht wichtig ist. Der MC68328 Prozessor wurde für den Einsatz in mobilen Geräten entwickelt. Die Einsatzdauer dieser Geräte hängt vom Stromverbrauch der Komponenten ab. Die Komponenten müssen deshalb auf die Reduktion des Stromverbrauchs optimiert sein.

2.3 Entwicklung von Palmprogrammen

2.3.1 Überblick

Die Entwicklung von Palmprogrammen unterscheidet sich in einigen Bereichen sehr stark von der Entwicklung von Desktopprogrammen. Es gibt allerdings auch Gemeinsamkeiten. Die Anwendungen sind ereignisgesteuert und die zur Verfügung stehenden Programmiersprachen reichen von Assembler über C bis zu Basic ähnlichen Skriptsprachen. Es gibt auch Entwicklungswerkzeuge bei denen man ähnlich wie bei Visual Basic von Microsoft in erster Linie oberflächenorientiert entwickelt und dabei die Basisfunktionalität des Programms erstellt, ohne eine Zeile Code geschrieben zu haben.

Die Unterschiede entstehen durch die Gerätegröße und den Einsatzzweck. Dieses betrifft die Programmentwicklung, den Speicherbedarf der Anwendungen, den

Aufbau des Anwendungs- und Datenspeichers, die Benutzungsoberfläche und das Zusammenspiel zwischen Gerät und Desktop.

2.3.2 Programmentwicklung

Palmprogramme werden nicht auf Palmrechnern entwickelt. Dafür haben diese Geräte einen zu kleinen Bildschirm, zu wenig Rechenleistung, zu wenig Speicher und eine zu ineffiziente Texteingabe. Deswegen werden Palmprogramme auf einem Desktoprechner entwickelt. Entwicklungsrechner können dabei Windows, Macintosh und Unix/Linux Rechner sein. Diese Systeme unterscheiden sich in der Anzahl der zur Verfügung stehenden Entwicklungswerkzeuge.

Der Test des Programms muss ebenfalls nicht unbedingt auf einem Palmrechner erfolgen. Es gibt die Möglichkeit, einen Emulator zu benutzen. Dieser Palmemulator ermöglicht es den Entwicklern das Programm für unterschiedliche Palms zu testen. So kann man das Verhalten bei unterschiedlichem Speicherausbau und unterschiedlichen Palmbetriebssystemen testen. Der Emulator benötigt für die Emulation eine Datei mit dem ROM des zu emulierenden Palms. Man erhält eine solche Datei, indem man die ROMs von seinem Palm herunterlädt. Ist man bei der Firma 3Com als Palmentwickler registriert, so kann man sich dort diese Dateien zu sämtlichen Palmversionen herunterladen.

Ein Test der Bedienung und der Geschwindigkeit des Programms muss allerdings trotzdem auf einem echten Palm durchgeführt werden.

2.3.3 Programmiersprachen

In diesem Kapitel soll ein Eindruck über die zur Verfügung stehenden Programmiersprachen und Entwicklungswerkzeuge vermittelt werden. Die Auswahl an Programmiersprachen und Entwicklungswerkzeugen war, wie man in [Rhodes & MCKeehan 1999] und internes Dokument IPVR Version 1.2, August 2000 nachlesen kann, bereits 1998 recht groß. Die Spanne reicht dabei von Assembler bis zu Rapid Prototyping/Rapid Application Development Werkzeugen.

Eine Entwicklung in Assembler ist die schwierigste Art Palmprogramme zu erstellen. Man besitzt allerdings den größten Einfluß auf das Palmprogramm und man ist auch in der Lage das Programm schneller und kleiner zu machen als Programme, die mit Hilfe anderer Programmiersprachen entwickelt wurden. Da der Palmprozessor ein CISC Prozessor ist, lässt er sich im Vergleich zu RISC Prozessoren relativ effizient programmieren. Der Entwicklungsaufwand ist allerdings trotzdem sehr viel höher als bei anderen Programmiersprachen. Diese Sprache wird deshalb in erster Linie zur Optimierung von Programmen eingesetzt.

Nach internes Dokument IPVR Version 1.2, August 2000 ist die am häufigsten verwendete Programmiersprache C. Diese Sprache ist fast genauso mächtig und flexibel wie Assembler. Leider ist das Entwickeln in C auch sehr fehleranfällig.

Es werden auf dem Palm keine C-Standardbibliotheken benutzt, sondern die Betriebssystemfunktionen, die das Palmbetriebssystem zur Verfügung stellt. Als Begründung wird in [Palm Programmer's Companion 2000] angegeben, dass die Programme so klein bleiben, da keine zusätzlichen Bibliotheken zum Programm

hinzugelinkt werden müssen. Dieses führt dazu, dass die Programme nur schlecht zu portieren sind, und außerdem zu einer längeren Einarbeitungszeit auch für Entwickler, die Erfahrung mit C haben. Die Unterstützung für diese Programmiersprache ist gut, da die Firma 3Com C-Bibliotheken und Dokumentation für die Entwicklung von C-Programmen zur Verfügung stellt. Es gibt kommerzielle und kostenlose C Entwicklungswerkzeuge. Unter den kommerziellen dürfte Metrowerks CodeWarrior das wichtigste Entwicklungswerkzeug darstellen, da dieser auch Unterstützung von der Firma 3Com erhält. Eine kostenlose Alternative dazu ist der GNU C Compiler gcc. Es gibt auch die Möglichkeit in C++ zu programmieren, allerdings scheint die Implementierung zumindest beim gcc nach [PRCTools 2000] noch nicht vollständig zu sein.

Es gibt auch höhere Programmiersprachen, die stärker von der Hardware abstrahieren als es in C oder Assembler geschieht. Nach [Palm Development 2000] gibt es Basic Interpreter und Basic Compiler (CASL, NS Basic). Auch Java kann man als Programmiersprache einsetzen. Nach internes Dokument IPVR Version 1.2, August 2000 gibt es unterschiedliche Java Projekte. Es gibt Projekte die aus kompilierten Javaprogrammen MC68000 Code erzeugen (JUMP). Andere versuchen eine Virtual Machine für Palm OS zu entwickeln (Ghost). Eine dritte Gruppe versucht eine Untermenge von Java zu implementieren. Diese Programme lassen sich dann auf allen Computern ausführen, die eine Java Virtual Machine haben (WABA). Mittlerweile gibt es auch kommerzielle Entwicklungsumgebungen, die Java für Embedded Systems anbieten und auch Palm OS unterstützen. Ein Beispiel hierfür ist Visual Age Micro Edition von IBM [VA Micro Edition 2000].

Neben diesen Programmiersprachen gibt es auch die Möglichkeit, Rapid ApplicationDevelopment/Rapid Prototyping Werkzeuge einzusetzen. Ein Beispiel hierfür ist Satellite Forms der Firma Pumatech. Wie man bei [Pumatech 2000] nachlesen kann, erzeugt man Programme durch Erstellen des Layouts und Festlegen der Funktionalität der Oberflächenelemente. Der Benutzer kann dabei auf vorgefertigte Funktionen zugreifen und muss nur solche Funktionen programmieren, zu denen es keine vorgefertigten Funktionen gibt. Diese Art der Programmierung ist für die Entwicklung von Anwendungen interessant, die hauptsächlich Daten in einer standardisierten Form darstellen sollen oder sehr wenig Anwendungslogik ausserhalb der Oberfläche haben. Diese sind beispielsweise Programme die Tabellen oder Datenbankdaten anzeigen sollen. Ein anderer Bereich ist die Entwicklung von Prototypen der Benutzungsoberfläche. Diese Entwicklungswerkzeuge sind recht geeignet um Palmprogramme zu schreiben, da viele Palmanwendungen in erster Linie Daten anzeigen sollen und keine komplexeren Aufgaben ausführen sollen. Die Entwicklungszeit lässt sich in solchen Fällen im Vergleich zu einer Entwicklung mit der Programmiersprache C drastisch verkürzen. In [Rhodes & MCKeehan 1999] wird beschrieben, dass die Autoren die Verkaufsanwendung, die im Buch entwickelt wird, mit Satellite Forms nachprogrammiert haben. Diese Anwendung besteht aus ungefähr 2000 Zeilen C Code. Die Autoren brauchten 3 Stunden, um sich in Satellite Forms einzuarbeiten und die Verkaufsanwendung neu zu erstellen.

2.4 Programmierung für Palms

2.4.1 Der Aufbau des Speichers

Der Handspring Visor Deluxe hat 8 MB RAM. Der Speicher besteht aus zwei Teilen. Der erste ist das "Dynamic Memory", indem die Daten des momentan ausgeführten Programms sind. Der zweite ist das "Storage Memory", das die Daten und Programme enthält. Das Dynamic Memory entspricht in etwa dem normalen RAM bei Desktopcomputern. Das Storage Memory hat die gleiche Funktion wie Festplatten oder Disketten bei Desktopcomputern. Es soll die Programme und Daten persistent speichern. Es wird deshalb im Gegensatz zum Dynamic Memory bei einem Reset auch nicht gelöscht. Da es sich bei beiden Teilen des Speichers um RAM handelt, wird ein Programm im Storage Memory ausgeführt und auch die Daten im Storage Memory müssen nicht in das Dynamic Memory kopiert werden, damit man auf sie zugreifen kann. Das Lesen dieser Daten erfolgt dabei über Zeiger auf die Daten. Allerdings lassen sich die Daten nur durch Betriebssystemfunktionen verändern und sind hardwaremässig schreibgeschützt. Dies verhindert, dass abstürzende Programme Daten anderer Programme beschädigen.

Das Dynamic Memory ist sehr viel kleiner als das Storage Memory. Die exakte Größe hängt vom Speicherausbau und von der verwendeten Betriebssystemversion ab. So beträgt das Dynamic Memory bei einem Palm OS Gerät mit 512 KB RAM und Palm OS 1.0 insgesamt 32 KB. Hierbei ist der Stack 2,5 KB groß und der Speicher für die Anwendungen beträgt ca. 12 KB. Ein Palm mit Palm OS 3.0 und mehr als 1 MB Speicher hat insgesamt 96 KB Dynamic Memory, einen Stack von 4 KB und ca. 36 KB Speicher für die Anwendungen.

2.4.2 Dateiartern auf dem Palm

Palms besitzen zwei Dateiartern. Die erste Dateiartern ist die Ressource Database. Sie wird wegen ihrer Endung (PRC) PRC Datei genannt und enthält Ressourcen d.h. Programmcode, Oberflächenelemente, Zeichenketten etc. Die zweite Dateiartern sind Palm Datenbanken und werden wegen ihrer Endung (PDB) auch PDB Dateien genannt. Sie dienen zur Speicherung von Daten, die dort in der Form von Records abgelegt werden. Diese Dateien entsprechen trotz ihres Namens Dateien auf herkömmlichen Systemen (vgl [Palm SDK reference 2000]).

2.4.3 Synchronisation zwischen Palm und Desktop

Die Synchronisation wird durch sogenannte Conduits gemacht. Entwickler haben die Möglichkeit eigene Conduits zu schreiben, die den Datenaustausch zwischen Palm und Desktop durchführen. Diese Conduits werden aufgerufen, wenn die Synchronisation zwischen dem Palm und dem Desktop aufgerufen wird. Die Kommunikation zwischen den beiden Geräten wird dabei nicht von den Conduits gemacht. Diese rufen nur Funktionen zum Lesen und Schreiben von Records auf. Conduits laufen dabei nur auf dem Desktop. Ihre Aufgabe ist es, die Daten zwischen Palm und Desktop auszutauschen und die Dateien zu konvertieren.

2.5 DB2 Everyplace

2.5.1 Motivation für den Einsatz einer Datenbank

Nach [Kemper & Eickler 1999] sind folgende Probleme beim Einsatz von Dateien Gründe für den Einsatz einer Datenbank:

- Redundanz und Inkonsistenz bei isolierter Datenhaltung
- Beschränkte Zugriffsmöglichkeit, da das Verknüpfen von Daten, die in isolierten Dateien vorhanden sind, schwer bis unmöglich ist
- Probleme des Mehrbenutzerbetriebs, da Dateisysteme nur rudimentäre Kontrollmechanismen für die Nebenläufigkeitskontrolle besitzen
- Gegen den Verlust von Daten haben Dateisysteme nur wenige Sicherungsmaßnahmen
- Integritätsverletzungen, da die Anwendungen die Integritätsbedingungen kennen müssen
- Sicherheitsprobleme beim Zugriff der Benutzer auf die Daten
- Hohe Entwicklungskosten, da bei der Anwendungsentwicklung die oberen Probleme beachtet werden müssen.

Betrachtet man die Probleme, so stellt man fest, dass diese Probleme auch bei den PDB Dateien aufzufinden sind.

Redundanzen und Inkonsistenzen

Redundanzen und Inkonsistenzen der Daten lassen sich auf dem Palm nur schlecht vermeiden, da die Anwendungen ihre eigenen PDB Dateien benutzen und nur selten auf PDB Dateien anderer Anwendungen zugreifen können. Der zweite Bereich der Redundanzen und Inkonsistenzen befindet sich bei den Daten, die zwischen Desktop und Palm synchronisiert werden müssen.

Eingeschränkte Zugriffsmöglichkeiten

Das Verknüpfen der Daten aus mehreren PDB-Dateien muss von der Palmanwendung übernommen werden. Allerdings wird der Anwender durch das Betriebssystem unterstützt, wenn er eine einzelnen PDB-Datei sortieren will oder Elemente darin suchen möchte.

Probleme durch Mehrbenutzerbetrieb

Probleme durch Mehrbenutzerbetrieb entstehen bei vielen Anwendungen nicht, da der Palm immer nur eine Anwendung ausführen kann. Solche Problemen können allerdings beim Zusammenspiel zwischen Palm und Desktopgerät entstehen. Werden die Daten auf dem Desktop und auf einem Palm geändert, so muss dieses bei der nächsten Synchronisation der Daten bemerkt und darauf reagiert werden. Wenn mehrere Palmbenutzer dieselben Daten verändern, so wird dieses bei der Synchronisation auf ein gemeinsames System z.B. einer Datenbank ebenfalls zu Problemen führen.

Verlust der Daten

Der Verlust von Daten ist ein großes Problem, das auch Datenbanksysteme nur schlecht lösen können, da Palms im Normalfall keinen persistenten Speicher haben. Hardwaredefekte oder das Löschen des Speichers, wenn die Batterien leer

sind, kann ein Datenbanksystem weder vermeiden noch beheben. Ein Schutz vor dem Löschen der Daten durch Softwarefehler ist in gewissem Umfang durch den Schreibschutz der Hardware gegeben. Für ein Zurücksetzen der Datenbank auf einen Zustand vor dem Fehler muss die Datenbank zusätzlichen Speicher belegen. Allerdings haben ältere Palmssysteme nur sehr wenig Speicher. Aus Sicht vieler Anwender ist die Belegung zusätzlichen Speichers nicht akzeptabel. Für diese ist das Anlegen einer Kopie der Daten bei der Synchronisation mit dem Desktop bereits ausreichend.

Integritätsverletzung

Das Palmbetriebssystem besitzt keine Funktionen, um Integritätsbedingungen der Daten sicherzustellen. Die Datenintegrität müssen die Anwendungsprogramme selbst sicherstellen. Probleme bei sich ändernden Integritätsbedingungen sind die Folge. Werden die Integritätsbedingungen erst auf dem Desktop überprüft, so kann man diese bei manchen Anwendungen nicht ohne weiteres wiederherstellen, da bereits Aktionen in der realen Welt ausgelöst wurden. Die Probleme entstehen dabei dadurch, dass die Daten über die Geräte mehrerer Anwender verteilt sein können und die Überprüfung der Integrität erst nach einer Verzögerung gemacht werden kann.

Sicherheitsprobleme

Palmgeräte haben ein großes Sicherheitsproblem, da diese Geräte leicht gestohlen werden können und die Daten leicht auf andere Palms übertragen werden können. Ein weiteres Sicherheitsproblem entsteht, wenn die Palmgeräte z.B. über Modems ihre Daten synchronisieren. Der Einsatz einer Datenbank, auf deren Daten man nur mit Hilfe eines Passwortes zugreifen kann und die Daten über sichere und standardisierte Protokolle austauscht, kann dieses Problem beheben.

Hohe Entwicklungskosten

Die Entwicklungskosten können sich durch den Einsatz einer Datenbank aus mehreren Gründen reduzieren. Die Lösung der oben genannten Probleme muss nur einmal erstellt werden und kann von den Entwicklern von Anwendungsprogrammen übernommen werden. Es lassen sich auch Entwicklungskosten einsparen, die typisch für Palmentwicklungen sind. Da für viele Programme Synchronisationsprogramme, sogenannte Conduits, erstellt werden müssen, muss für jede Plattform, mit der das Programm eine Synchronisation durchführen können soll, ein solcher Conduit geschrieben werden. Diese Entwicklungskosten lassen sich einsparen, wenn die Datenbank die Synchronisation der Daten übernimmt. Dadurch hat jede Plattform, mit der die Datenbank eine Synchronisation durchführen kann, einen funktionierenden Synchronisationsmechanismus. Eine weitere Kosteneinsparung ergibt sich durch die leichtere Portierbarkeit der Anwendungen, da zumindest der Zugriff auf die Daten portiert werden kann. Dieses führt auch zu geringeren Schulungskosten, da die Entwickler die unterschiedlichen Dateiverwaltungen für die verschiedenen Zielplattformen nicht kennenlernen müssen.

2.5.2 Bestandteile von DB2 Everyplace

DB2 Everyplace ist der Nachfolger des Produktes DB2 Everywhere. Es besteht aus einer Datenbankkomponente, einem Synchronisationsserver und dem Entwicklungswerkzeug Personal Application Builder. Es existiert kein Administrations-

programm, da mobile Datenbanken ohne Administration auskommen müssen vgl. [Davis 1999]. Ein Datenbankadministrator kann nicht kurz zum Anwender gehen und dort die Datenbank neu konfigurieren, da die Benutzer nicht in der Nähe sind und sich sogar in anderen Ländern aufhalten können. Es werden Bibliotheken und Headerdateien mitgeliefert, die für die Entwicklung einer Datenbankanwendung unter C notwendig sind.

DB2 Everyplace läuft auf folgenden Plattformen:

- Palm OS
- Symbian EPOC
- QNX Neutrino
- embedded Linux
- Windows CE & Pocket PC

Die Datenbankkomponente

Hierbei handelt es sich um eine relationale Datenbank, die 125 Kilobyte groß ist. Sie ist auf Double Byte Character vorbereitet, wodurch sie auch Zeichen ausserhalb des ASCII Zeichensatzes versteht. Sie wird über eine DB2 Everyplace CLI/ODBC Schnittstelle angesprochen. Diese ist eine Untermenge der DB2 CLI. Über diese werden SQL Anweisungen an die Datenbank gesendet. Diese SQL Befehle bilden eine Untermenge von SQL92 und sind stark vereinfacht; z.B. gibt es keine Subqueries, Unions oder Trigger. Sie kann Indexe erstellen, kennt Primärschlüssel und kennt grundlegende Datentypen. Man kann auch Integritätsbedingungen angeben.

Der Synchronisationsserver

Der Synchronisationsserver ist ein Client/Server-Programm, das die Zweiwegesynchronisation zwischen einer Quell- und Zieldatenbank verwaltet. Er ist der Vermittler zwischen DB2 Everyplace auf der mobilen Einheit und einer Datenbank auf einem Quellenserver. Man kann mit diesem Benutzergruppen einrichten und definieren, auf welche Daten diese zugreifen können. Den Aufbau einer Synchronisationsarchitektur zeigt Abb. 2-1. Der Synchronisationsserver kann sich dabei auf einem Mittlersystem befinden oder direkt auf dem Quellensystem laufen. Die Synchronisation zwischen dem mobilen Gerät und dem Mittlersystem kann dabei durch eine Cradle, ein Modem oder eine TCP/IP fähigen Kommunikationseinheit erfolgen.

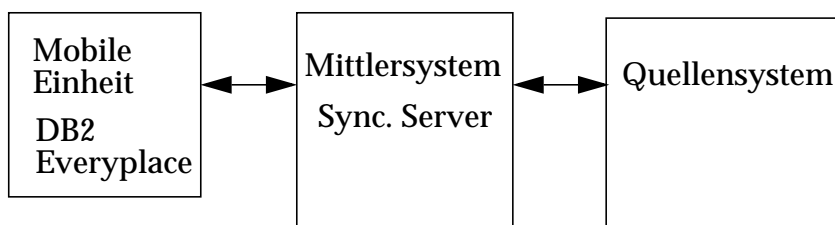


Abbildung 2-1: Aufbau der Synchronisationsarchitektur

Der Personal Application Builder

Personal Application Builder ermöglicht das visuelle Erstellen der Oberfläche eines Palmprogramms. Er ermöglicht die Programmlogik für die Darstellung und Änderung von Tabellenelementen automatisch für Tabellen zu generieren, deren Schema man in das Programm importiert hat. Den einzelnen Oberflächenelementen kann man Standardfunktionen zuweisen. Es besteht auch die Möglichkeit statt Standardfunktionen eigene Programmlogik zu erstellen. Als Programmiersprache wird dabei C benutzt. Der Personal Application Builder erzeugt C Code, der durch den gcc kompiliert werden kann.

2.5.3 Entwicklung von Datenbankanwendungen

Möchte man Anwendungen schreiben, die auf die DB2 Datenbankkomponente zugreifen sollen, so ist die einfachste Möglichkeit den Personal Application Builder zu benutzen. Dieser erstellt ein Makefile, das alle notwendigen Dateien zum Programm hinzulinkt, und ein C Programm, das die für die Ansteuerung notwendigen Headerdateien einliest. Reichen die angebotenen Funktionen, so kann man eine Anwendung entwickeln ohne sich um die Ansteuerung der Datenbank kümmern zu müssen.

Die zweite Möglichkeit ist ein Programm in C zu erstellen. Hierbei muss man dann allerdings selbst dafür Sorge tragen, dass die Bibliothek für die Ansteuerung der Datenbank dem Programm hinzugelinkt wird. Die erstellten C-Dateien müssen die DB2 Everyplace Headerdateien einlesen, die für die Ansteuerung der Datenbank zuständig sind.

2.5.4 Import/Export der Daten

Es gibt drei Möglichkeiten Daten in die Datenbank zu importieren oder zu exportieren. Die aufwendigste Art ist ein Programm zu schreiben, das die Daten in die Datenbank über die ODBC Schnittstelle einträgt beziehungsweise ausliest. Die zweite Möglichkeit ist die DB2 Everyplace Datenbank mit der Quelldatenbank zu synchronisieren. Die dritte Möglichkeit besteht darin CSV Dateien (Comma Separated Values) in die Datenbank zu importieren beziehungsweise zu exportieren. Der Import benötigt zwei Dateien. Die erste enthält das Schema der Tabelle und die zweite enthält die Daten. Aus diesen Dateien wird durch ein Programm auf dem Desktop eine Datei im Format der DB2 Everyplace Datenbank erzeugt. Diese Datei lässt sich auf dem Palm mit Hilfe des Programms DB2eImport in die Datenbank einfügen. Der Export der Daten erfolgt auf dem Desktop über ein anderes Programm, das CSV Dateien erzeugt.

2.6 Eingesetzte Entwicklungswerkzeuge

Als Programmiersprache wurde C benutzt, da die zur Ansteuerung der Datenbank notwendigen Dateien nur für C existieren. C++ wurde nicht eingesetzt, da dieses wahrscheinlich zu Verzögerungen bei der Entwicklung geführt hätte. Da die Beispiele zu DB2 Everyplace in C sind und auch die Beispiele und Dokumente ([Rhodes & MCKeehan 1999],[Palm Programmer's Companion 2000],[Palm Programmer's Companion 2000]) zur Programmierung für Palm von C ausgehen, wäre es wahrscheinlich zu Problemen gekommen, die bei C nicht aufgetreten wären. Da ich einen technischen Durchstich machen sollte, war es wichtig, dass

ich ein Programm erstelle, anhand dem Bewertungen vorgenommen werden konnten. Zeit für Probleme durch Unterschiede zwischen C und C++ gab es nicht. Da ich mich in die Ansteuerung von DB2 Everyplace und in die Palmprogrammierung einarbeiten musste, wäre das Risiko, zu lange an einem Programm entwickeln zu müssen, zu hoch gewesen. Als Compiler benutzte ich die PRC-Tools in der Version 2.0. Diese enthalten den gcc-Compiler. Der Personal Application Builder kam nicht zum Einsatz, da dieser keine Modularisierung unterstützt und von ihm erzeugte Programme deshalb monolithisch sind.

Es wurde zu Beginn der Studienarbeit DB2 Everyplace Personal Edition für die Entwicklung benutzt und erst zur Mitte der Studienarbeit auf eine DB2 Everyplace Enterprise Edition in der Version 7.1 gewechselt. Der Import der Daten erfolgte mit Hilfe von CSV Dateien, da für eine Synchronisation mit Hilfe des Synchronisationsservers eine DB2 Version 7.0 notwendig gewesen wäre, aber nicht vorhanden war. Die Synchronisation wurde im Rahmen dieser Studienarbeit deshalb auch nicht getestet. Da die Anwendung auch keine Änderung der Daten auf dem PDA erlaubt und die Daten sich nicht schnell ändern, wäre eine Bewertung der Synchronisation auch nicht sinnvoll gewesen.

2.7 Zusammenfassung

Die Palms sind Geräte, die Desktops nicht ersetzen, sondern eine Symbiose mit diesen eingehen. Sie überlassen rechenintensive Aufgaben und die Eingabe größerer Eingabemengen dem Desktoprechner. Ihre Aufgabe ist es dem Benutzer Zugriff auf Daten zu ermöglichen, wenn dieser mobil ist. Für eine hohe Mobilität müssen diese deshalb klein sein und sollten möglichst lange ohne festen Stromanschluss funktionieren können. Sie erreichen dies, indem sie auf eine Tastatur verzichten, ein kleines Display haben und nur wenig Speicher und Rechenleistung besitzen. Der RAM-Speicher dieser Geräte ist zweigeteilt. Es existiert Speicher, der nach der Ausführung eines Programmes wieder gelöscht wird und Speicher zur Speicherung der Programmdateien, die nicht gelöscht werden sollen. Veränderungen dieses Speichers lassen sich nur über Betriebssystemfunktionen machen.

Für die Anwendungsentwicklung existiert für Palmrechner eine große Auswahl an Programmiersprachen und Entwicklungswerkzeugen. Die Spanne reicht dabei von Assembler bis zu Rapid Prototyping Werkzeugen, mit denen man Programme fast ohne Eingabe von Code erstellen kann. Das Programm wird in C erstellt.

Der Einsatz einer Datenbank zur Speicherung der Daten ist wegen der Eigenschaften der PDB-Dateien sinnvoll. Als Datenbank wird DB2 Everyplace benutzt. Ein Test des Synchronisationsservers wird im Rahmen dieser Studienarbeit nicht gemacht.

Nachdem dieses Kapitel einen Überblick über das technische Umfeld gegeben hat, wird in den folgenden Kapiteln die Entwicklung und das entwickelte Palmprogramm genauer beschrieben.

3 Das Vorgehen

3.1 Anforderungen

Das Vorgehensmodell für die Entwicklung des Palmprogramms sollte eine Planung erlauben, da die Studienarbeit nach 6 Monaten beendet sein muss. Da weder ich noch die Betreuerin Erfahrung in der Palmentwicklung hatten, waren die typischen Probleme bei Palmentwicklungen nicht bekannt. Es sollte aber Probleme, wie z.B. eine schlechte Bedienung oder Fehler bei der Anforderungsanalyse bereits früh erkannt werden können, da sonst die Korrektur dieser Probleme nicht mehr im Rahmen dieser Studienarbeit möglich gewesen wäre. Dies hätte zur Folge, dass ein Programm entwickelt worden wäre, das die falschen Probleme löst und bei besonders schwerwiegenden Problemen drängt die Behebung dieses Problems andere Tätigkeiten in den Hintergrund. Das Vorgehensmodell sollte die Einarbeitung in die Palmprogrammierung berücksichtigen und geschickt einbauen.

3.2 Das Vorgehensmodell

Als Vorgehensmodell wurde ein Phasenmodell benutzt, das Prototypen für die Bestimmung der Funktionen und dem Aussehen der Benutzungsoberfläche einsetzt. Das Phasenmodell ermöglichte eine gewisse Planung der Tätigkeiten. Es bestand aus den Tätigkeiten Anforderungsanalyse, Spezifikation des Programms, Entwurf der Architektur, Implementierung, Test und Evaluation. Der Einsatz von Prototypen hatte mehrere Vorteile. Der erste war die Möglichkeit die Benutzungsoberfläche festzulegen. Hierdurch konnte auch die von der Betreuerin gewünschte Funktionalität besser veranschaulicht werden. Ein weiterer Vorteil ergibt sich dadurch, dass ich durch Erstellung von Prototypen eine Einarbeitung in die Programmierung hatte. Die Einarbeitung wurde so benutzt, um andere nützliche Ergebnisse für die Entwicklung des Palmprogramms zu erhalten. Die Erfahrungen durch die Prototypenentwicklung konnten auch in das Endprodukt einfließen. Beispielsweise war es nach der Entwicklung klar, wie man die Eventbehandlung auf mehrere Module verteilt. Ein weiterer Vorteil ergibt sich durch den Einsatz von technischen Prototypen, mit denen technische Probleme früh erkannt werden sollten.

3.3 Der konkrete Ablauf

Nach der Erstellung des Projektplans wurde die Anforderungsanalyse gemacht. Hierbei wurden Use Cases erstellt und anhand der Anforderungen wurde ein erster Prototyp entwickelt, der die Benutzungsoberfläche darstellte. Kurz darauf war die Personal Edition von DB2 Everyplace verfügbar. Diese besitzt Beispielprogramme mit denen man DB2 Everyplace ausprobieren kann. Diese Programme dienten als technische Prototypen. Die Geschwindigkeit war dabei ausreichend. Die Spezifikation wurde danach erstellt und anschließend begann der Entwurf des Programms. Am Ende der darauffolgenden Implementierungsphase wurden die Daten für die ganze Fakultät in die Datenbank geladen und festgestellt, dass das Programm ein Geschwindigkeitsproblem hat. Solange dieses Problem bestand, war eine Evaluation des Programmes nicht sinnvoll. Stattdessen musste die Geschwindigkeit der Datenbank optimiert werden.

4 Die Anforderungen an das Programm

In diesem Kapitel wird das Einsatzgebiet des Programms kurz beschrieben. Danach werden die Zielgruppen der Arbeit vorgestellt und deren Anforderungen beschrieben. Use Cases werden den Einsatz des Programms beschreiben und die wichtigsten funktionalen Anforderungen an das Programm werden hieraus abgeleitet. Daraufhin werden die Anforderungen an die Daten und die Daten selbst kurz beschrieben.

4.1 Einsatzgebiet des Programms

Im Rahmen dieser Studienarbeit sollte eine ortsabhängige Datenbank Anwendung entstehen, die aufgrund von Lokationsinformation durch den Benutzer räumliche Anfragen an eine Datenbank stellt und so den nächsten Mitarbeiter zu bestimmten Aufgabengebieten finden soll. Hierdurch wird eine Funktion realisiert, die in vielen Anwendungen benötigt wird. Ein Objekt zu finden, das räumlich am nächsten zu einer Position ist, wird beispielsweise bei Routenplanern (finde die nächste Ausfahrt) benötigt. Ein weiteres Einsatzgebiet sind Messeführer, bei denen man nach den nächsten Ausstellern zu bestimmten Themen sucht. Man kann sich auch neue Anwendungen überlegen, wie z.B. Geräte, die die nächste Notrufsäule finden sollen oder Anwendungen mit denen man das nächste Restaurant finden kann, das ein bestimmtes Menu anbietet. Im folgenden wird diese Funktion nearest Funktion genannt.

Für die Studienarbeit wird ein Programm geschrieben mit dem man sich im Hause orientieren kann und Mitarbeiter zu bestimmten Themengebieten finden kann.

4.2 Zielgruppen der Arbeit

Die erste Zielgruppe sind die Anwender, die das Programm bedienen sollen und die zweite Gruppe sind Entwickler, die das Programm erweitern, verändern oder zum Test anderer Anwendungen einsetzen wollen. Die Gruppe der Entwickler besteht dabei aus Entwicklern des Nexusprojektes und Entwicklern der Abteilung Anwendersoftware. Diese beiden Gruppen sind nicht überschneidungsfrei allerdings haben diese unterschiedliche Interessen an dem Programm.

4.2.1 Anwender

Die Anwender werden das Programm benutzen, um sich im Gebäude zu orientieren und um sich die Position und Daten zu Mitarbeitern anzeigen zu lassen. Das Programm wird auf das Fakultätsgebäude beschränkt sein. Allerdings ist zu erwarten, dass dieses Programm später auch für andere Gebäude und Gebiete benutzt werden soll. Eine Erweiterung um die Suche nach anderen Objekten wie z.B. Hörsäle ist ebenfalls zu erwarten. Typische Anwender könnten Besucher am Tag der offenen Tür sein oder Erstsemester, die noch Orientierungshilfen für das Gebäude brauchen. Personen, die bestimmte Mitarbeiter im Gebäude suchen, könnten dieses Programm ebenfalls benutzen. Hierbei handelt es sich in erster Linie um Studenten. Es können aber auch Familienmitglieder sein.

4.2.2 Nexusentwickler

Die Nexusentwickler haben mehrere Einsatzmöglichkeiten für das Programm. Es dient als Beispiel für eine ortsabhängige Anwendung und könnte zu Demonstrationszwecken eingesetzt werden. Da es sich hierbei um Entwickler handelt, könnten diese das Programm um weitere Funktionen erweitern. Die Erweiterungen könnten beispielsweise Alternativen der Positionierung demonstrieren. Die für die Studienarbeit geschriebene Version des Programms greift noch nicht auf Nexusdienste zu. Dieses könnten die Nexusentwickler ändern, um so die Dienste des Nexusserver zu demonstrieren zu können. Da PDAs und damit auch Palms eine wichtige Zielgruppe darstellen, die auf die Nexusdienste zugreifen sollen, wird dieses Programm nützlich sein, wenn man Protokolle für den Ablauf der Kommunikation zwischen Palms und Nexusserver vergleichen und bewerten soll. Die Nexusentwickler wissen noch nicht wie der Aufbau der Nexuskomponente auf den Clients auszusehen hat. Die Anwendung könnte hierbei hilfreich sein, da man mit ihrer Hilfe die Anforderungen an diese Nexuskomponente analysieren kann.

4.2.3 Abteilung Anwendersoftware

Die Abteilung Anwendersoftware möchte mit dieser Studienarbeit die Anforderungen an Datenbanken für PDAs wie die Palms kennenlernen. Hierbei sollen die Möglichkeiten von bestehenden Datenbanken am Beispiel von DB2 Everyplace betrachtet werden. Das Programm könnte später dafür benutzt werden um auch andere Datenbanken -möglicherweise sogar eine Selbstentwickelte- zu bewerten und zu vergleichen. Eine Eigenschaft der PDAs und der Palms ist das Fehlen von guten Eingabemöglichkeiten und eines großen Bildschirms. Dieses führt dazu, dass herkömmliche Anfragemethoden wie SQL-Anfragen oder Query By Example Probleme haben. SQL-Anfragen sind problematisch wegen des Fehlens einer effizienten Eingabemöglichkeit. QBE, ein mit DB2 Everyplace ausgeliefertes Programm, das Query By Example demonstriert, hat Probleme bei Tabellen mit vielen Spalten oder vielen Zeilen. Abhängig von den Tabellenelementen lassen sich maximal 4 Spalten und 11 Zeilen anzeigen. Es besteht daher in diesem Bereich ein Bedarf an Anfragesprachen, die für die Nutzung unter PDAs optimiert sind.

4.3 Die Anforderungen

4.3.1 Allgemeine Anforderungen an Palmprogramme

An Palmprogramme werden aufgrund der Eigenschaften der Palms bestimmte Anforderungen gestellt, die sich von Desktopprogrammen unterscheiden. Diese entstehen durch die geringe Größe des Geräts, den schlechten Eingabemöglichkeiten von Text, dem geringen Speicher und der geringen Leistungsfähigkeit der Palmgeräte. Diese Anforderungen sind nach [Rhodes & MCKeehan 1999] folgende Punkte:

- Die kleine Bildschirmgröße muss berücksichtigt sein
- Die Eingabe von Text sollte begrenzt sein
- Fähigkeit zur Synchronisation mit einem Desktopcomputer
- Geringer Speicherverbrauch

- Hohe Ausführungsgeschwindigkeit

Die Fähigkeit zur Synchronisation übernimmt bei diesem Programm DB2 Everyplace. Im Rahmen dieser Studienarbeit wurde allerdings keine Synchronisation eingesetzt. Die anderen Eigenschaften sind Anforderungen, die auch die restlichen Programmteile betreffen.

4.3.2 Anforderungen durch Anwender

Neben einer einfachen Bedienung sollte das Programm die Orientierung im Gebäude erleichtern. Der Anwender sollte bei der Ausführung der Funktionen nicht oder nur kurz warten. Die find nearest Funktion wird vom Anwender nicht ständig aufgerufen, wodurch eine kurze Verzögerung bis zur Ausgabe des Ergebnisses akzeptabel ist. Das Maximum für die Verzögerung sollte wahrscheinlich nicht mehr als zehn Sekunden betragen. Die genaue Zahl hängt allerdings vom Anwender und dessen Einsatzzweck für das Programm ab.

4.3.3 Anforderungen der Nexusentwickler

Das Programm muss ortsabhängige Funktionen demonstrieren, da es sonst nicht in den Rahmen des Nexusprojektes passt.

Das Programm soll für Demonstrationszwecke eingesetzt werden und muss deshalb eine leicht verständliche Bedienung haben, die nur einer kurze Einführung benötigt.

Da das Programm weiterentwickelt werden soll, ist eine leichte Erweiterbarkeit des Programms wichtig. Weil das Programm zur Demonstration von Nexusdiensten eingesetzt werden kann, wird es in späteren Versionen auf die Nexusdienste zugreifen können. Diese Funktionen sind momentan im Programm selbst implementiert und müssen für den Zugriff auf Nexusdienste geändert werden, da diese durch Nexusserver bereitgestellt werden. Eine leichte Änderbarkeit des Programms ist deshalb wichtig. Da die Nexusentwickler sich nicht allzu lange für die Entwicklung von Demonstrationsprogrammen beschäftigen wollen, werden Teile des Programms für Demonstrationsprogramme anderer Nexusdienste wiederverwendet. Hierfür sollten die Programmteile gut wiederverwendbar sein und die Kopplung zwischen den einzelnen Programmteilen sollte möglichst niedrig sein.

4.3.4 Anforderungen der Abteilung Anwendersoftware

Die Abteilung Anwendersoftware möchte die Datenbank DB2 Everyplace bewerten und später vielleicht mit anderen Datenbanken vergleichen können. Hierfür sollte man die Datenbank leicht auswechseln können. Weiterhin sollten die Anfragen an die Datenbank leicht geändert werden können. Ein Vergleich zwischen zwei Datenbanken sollte leicht implementiert werden können. Dieses erfordert auch eine klare Trennung zwischen den Algorithmen für das Bearbeiten der Daten und dem Einlesen der Daten.

4.4 Use Cases

Die Use Cases zeigen für welche Aufgaben das Programm eingesetzt werden sollte. Sie entstanden nach der Befragung des Kunden (die Betreuerin).

Der Anwender startet aus unterschiedlichen Gründen das Programm:

- Er möchte das Gebäude anschauen (Use Case 1)
- Er will sich mit Hilfe dieses Programms im Gebäude orientieren (Use Case 2)
- Er will Informationen zu einem Raum haben (Use Case 3)
- Er will Mitarbeiter zu einem Stichwort finden (Globales Find) (Use Case 4)
- Es sucht die von seiner Position nächsten Mitarbeiter zu einem Stichwort (find nearest) (Use Case 5)
- Er sucht einen Mitarbeiter im Gebäude (Use Case 6)
- Er gibt eine SQL-Anweisung ein

4.4.1 Use Case 1: Anschauen des Gebäudes

Der Benutzer will sich mit der eingebauten Karte einen Überblick über das Gebäude machen. Er interessiert sich allerdings nicht für die Räume und möchte auch sonst keine Informationen erhalten.

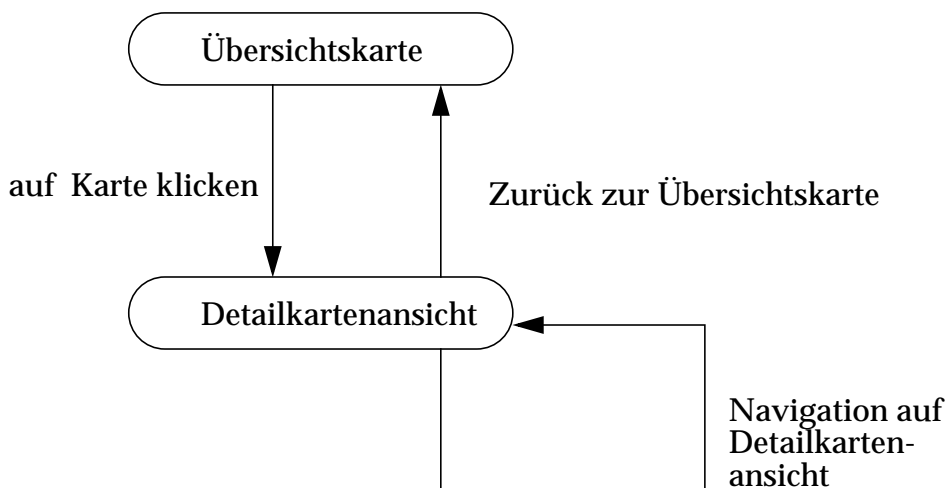


Abbildung 4-1: Darstellung der Navigation

Der Anwender startet das Programm und sieht die Übersichtskarte des Gebäudes. Auf dieser Karte wählt er ein Gebäudeteil aus, für das er die Detailkarte betrachten will. Dies macht er, indem er auf die Karte klickt. Nun erscheint eine Detailkarte dieses Gebäudeteils. Auf der Detailkarte kann er Teile des Gebäudes genauer betrachten. Wenn er einen anderen Teil des Gebäudes ansehen will, kann er dieses auf zwei unterschiedliche Wege machen. Eine Möglichkeit ist das Wechseln der Ansicht auf die Übersichtskarte. Von dieser aus kann er einen anderen Teil des Gebäudes auswählen. Die zweite Möglichkeit besteht darin, mittels Navi-

gationstasten auf angrenzende Gebäudeteile zu wechseln und sich so diese anzeigen zu lassen. Falls Gebäudeteile über oder unter dem angezeigten Gebäudeteil sind, kann man sich auch diese anzeigen lassen.

4.4.2 Use Case 2: Orientierung im Gebäude/ Markierung der Position

Der Benutzer befindet sich irgendwo im Gebäude. Er weiß an welcher Stelle er im Gebäude ist. Er will seine Position an dieser Stelle auf der Detailkarte markieren. Danach kann er find nearest ausführen oder auf der Karte navigieren.

Der Benutzer wechselt in die Detailkarte, die seine Position enthält und markiert seine Position, indem er auf den Teil der Karte klickt, an der er sich befindet. Nun erscheint ein X und markiert so seine Position. Jetzt kann der Benutzer Use Case 5 ausführen oder sich andere Teile des Gebäudes anschauen. Betrachtet er wieder seinen Kartenausschnitt, so sieht er ein X. Der Benutzer kann sich so das Gebäude betrachten und seine nächsten Schritte überlegen ohne Angst davor haben zu müssen, später die Stelle, an der er sich befindet, auf der Karte nicht wiederzufinden. Nach einiger Zeit verschwindet das X wieder, da das Programm dann annimmt, daß sich der Anwender weiterbewegt hat.

Use Case 2 ist Grundlage für Use Case 5.

4.4.3 Use Case 3: Informationen zu einem Raum erhalten

Der Benutzer interessiert sich für die Informationen zu einem Raum.

Der Benutzer wechselt hierfür in die Detailansicht und klickt dann einfach auf den Raum, der ihn interessiert. Als nächstes erscheinen Informationen zu diesem Raum, wie die Mitarbeiter in diesem Raum, die Art des Raums z.B. Mitarbeiterraum. Nachdem der Benutzer die Informationen erhalten hat, kann er wieder auf die Detailansicht wechseln.

4.4.4 Use Case 4: Mitarbeiter zu einem Stichwort finden (global find)

Der Benutzer will Mitarbeiter finden, die im Zusammenhang zu einem bestimmten Thema (Stichwort) stehen. Es soll eine Suche im ganzen Haus stattfinden.

Der Benutzer ruft die Dialogbox für das global find (Suchen im ganzen Gebäude) auf. Nun wählt er ein Stichwort aus einer Liste der möglichen Stichwörter und lässt den Rechner danach suchen. Er erhält eine Liste mit Mitarbeitern der Universität Stuttgart, die mit diesem Stichwort in Verbindung stehen. Wählt man eine Person aus der Liste aus, so wird dessen Position angezeigt. Falls dieser Mitarbeiter auf der Detailkarte ist, dann wird dessen Position dort mit einem O markiert. Wenn die Position des Mitarbeiters außerhalb des Kartenausschnitts ist, dann wird die Übersichtskarte angezeigt und die Position des Mitarbeiters mit O und die des Benutzers mit X markiert.

Es wird auch die Möglichkeit existieren nach mehreren Stichwörtern zu suchen, die durch AND und OR miteinander verknüpft sind. Als Zeichen für AND wird

“+” und für OR wird ein Leerzeichen benutzt. OR ist die Standardverknüpfung zweier Stichwörter.

4.4.5 Use Case 5: Nächsten Mitarbeiter zu Stichwort finden (find nearest)

Dieser Use Case ist ähnlich zu Use Case 4, da er aus Sicht des Benutzers eine ähnliche Aufgabe hat.

Der Benutzer hat seine Position markiert und will nun den nächsten Mitarbeiter zu einem Stichwort finden.

Nach der Markierung seiner Position (Use Case 2) ruft der Benutzer die Dialogbox für find nearest auf. Wie in Use Case 4 kann der Anwender eine Suche nach Stichwörtern machen. Die Anzeige der Mitarbeiter lässt sich aber auf Mitarbeiter beschränken, die in der Nähe zur eigenen Person sind. Die Ausgabe der Mitarbeiter lässt sich auf die beschränken, die innerhalb einer bestimmten Entfernung zum Benutzer sind. Diese Entfernung lässt sich einstellen. Die möglichen Entfernungen sind: Karte, Gebäudeflügel, Stockwerk, ganzes Gebäude. Weiterhin sind die Mitarbeiter nach Entfernung zur Position des Benutzers sortiert. Die Markierung der Mitarbeiter erfolgt wie bei Use Case 4 (global find).

Hat der Benutzer seine Position vor Aufruf von find nearest nicht markiert, so muss er dieses dann nachholen.

4.4.6 Use Case 6: Mitarbeiter mit Namen suchen

Der Benutzer sucht einen Mitarbeiter und kennt den Namen von diesem.

Der Benutzer wechselt hierfür auf die Dialogbox für die Suche. Dort gibt er an, dass er nach Namen suchen möchte. Dann gibt er den Namen in ein Textfeld ein. Nun kann er global find auswählen, um alle Mitarbeiter mit diesem Namen zu finden oder er wählt find nearest aus und erhält dann die Mitarbeiter mit diesem Namen, die von ihm aus räumlich am nächsten sind.

4.4.7 Use Case 7: Eingabe einer SQL-Anweisung

Der Benutzer kann durch Eingabe einer SQL-Anweisung Ergebnisse anzeigen lassen, die er durch die herkömmliche Funktionalität nicht bekommen könnte.

Hierfür wechselt der Anwender in die Dialogbox für SQL-Anweisungen. Nun sieht er ein Textfeld, in dem er seine SQL-Anweisung eingeben kann. Wenn er diese Eingabe gemacht hat, lässt er diese Anweisung ausführen und kann sich das Ergebnis der Anfrage anzeigen lassen.

4.5 Funktionale Anforderungen

4.5.1 Überblick

Die funktionalen Anforderungen wurden anhand der Use Cases und anhand Besprechungen mit dem Kunden erstellt. In diesem Abschnitt werden nur die

wichtigsten funktionalen Anforderungen beschrieben. Eine komplette und detailliertere Aufzählung findet man in [Grzan 2000].

Anforderung 2: Navigation

Der Benutzer hat geeignete Möglichkeiten zur Navigation auf Karten.

Anforderung 2.1: Wechsel zwischen Ansichten

Der Benutzer kann zwischen einer Übersichtskarte und einer detaillierteren Ansicht wechseln. Dieser Wechsel erfolgt durch einfaches anklicken der Übersichtskarte auf die Stelle, die der Anwender in der detaillierten Ansicht sehen möchte. In der detaillierten Ansicht kann man dann wieder auf die Übersichtskarte wechseln.

Anforderung 2.2: Navigation auf der detaillierten Karte

Es besteht die Möglichkeit in alle vier Himmelsrichtungen zu navigieren und es gibt die Möglichkeit den Kartenausschnitt zu sehen, der ein Stockwerk über oder unter dem momentanen Kartenausschnitt ist. Bei der Navigation kann man sich nicht aus dem Gebäude herausbewegen. Sollte der Anwender dieses versuchen, wird weiterhin der momentane Kartenausschnitt angezeigt und somit das Kommando des Benutzers ignoriert.

Anforderung 3: Markierung der Position

Das Programm zeigt so, dass es mit Informationen zur Positionierung umgehen kann. Dieses ist für die Nexusforschungsgruppe die Grundlage, um ihr Themengebiet zu demonstrieren.

Anforderung 4: Erhalt von Informationen eines Raums

Durch Anklicken eines Raumes auf der Detailkarte erscheinen Informationen zu diesem.

Diese Funktion erhöht die Nützlichkeit der Anwendung, da sie so eine bessere Informationsquelle für den Anwender darstellt. Er kann dadurch auch besser entscheiden, welchen Raum er als nächstes betreten will.

Anforderung 5: Anzeige Mitarbeiterdaten

Es lassen sich allgemeine Mitarbeiterdaten anzeigen, wie die Telefonnummer, den Arbeitsraum des Mitarbeiters und dessen Stichworte.

Anforderung 6: Stichwortsuche

Der Benutzer ist in der Lage eine Suche nach Stichwörtern (Mitarbeiternamen oder Themen) durchzuführen. Als Ergebnis erhält er dann eine Liste mit Mitarbeitern, die im Zusammenhang mit diesem Stichwort stehen. Die Stichworte lassen sich dabei verknüpfen. Als Verknüpfungsmöglichkeiten gibt es ein '+' für eine AND-Verknüpfung zweier Stichworte und ein Leerzeichen als OR Verknüpfung zweier Stichworte.

Anforderung 7: Räumliche Anfragen

Hierbei soll der räumlich nächste Mitarbeiter angezeigt werden. Es soll zwei Möglichkeiten für die Ausführung räumlicher Anfragen geben:

- **global find:** Für eine Suche in der kompletten Datenbasis d.h. auch externe Mitarbeiter werden ausgegeben
- **find nearest:** Es werden nur die Mitarbeiter ausgegeben, die innerhalb einer bestimmten Entfernung zum momentanen Ort des Benutzers sind. Es werden keine externen Mitarbeiter angezeigt

Anforderung 8: Mitarbeitersuche über Namen

Die Suche entspricht der Stichwortsuche nur soll nach Namen gesucht werden. Eine Verknüpfung durch AND soll nicht möglich sein.

Anforderung 9: Anzeige der gefundenen Mitarbeiter

Die gefundenen Mitarbeiter werden nach der Entfernung zum Standort des Benutzers sortiert ausgegeben.

Anforderung 10: Anzeige der Positionen von Mitarbeitern und Benutzern

Die Position eines in Use Case 4 oder Use Case 5 ausgewählten Mitarbeiters wird angezeigt. Wenn die Position des Mitarbeiter auf der momentan angezeigten Detailkarte ist, dann wird er dort angezeigt. Sollte er außerhalb der Karte sein, so wird auf die Übersichtskarte gewechselt und die Position des Benutzers und des Mitarbeiters auf der Übersichtskarte markiert.

Anforderung 11: Eingabe einer SQL-Anweisung

Diese Funktion erweitert die Möglichkeiten der Anwendung sehr stark. Ohne diese kann man keine Informationen abfragen, die von den eingebauten Abfragen nicht vorgesehen sind.

4.6 Anforderungen an die Daten

Allgemein

Die Daten werden vom Telefonverzeichnis der Fakultät Informatik übernommen. Es ist für den Prototypen nicht wichtig, ob diese Daten korrekt oder vollständig sind. Der Prototyp muss aber genügend Daten erhalten, damit er evaluiert werden kann. Ein Datenaustausch mit anderen Systemen ist nicht wichtig und dient nur zum Einfügen der Daten in die Datenbank. Wenn das Produkt eingesetzt werden sollte, so werden sehr viel höhere Anforderungen an die Konsistenz der Daten gestellt. Benutzer wären ziemlich verärgert, wenn sie zu einem Zimmer gehen und dort feststellen, dass der gesuchte Mitarbeiter in einem anderen Zimmer arbeitet, nicht existiert oder an völlig anderen Dingen arbeitet als in der Datenbank beschrieben.

Import/Export der Daten in der Datenbank

Die Daten werden durch die Import/Export Programme, die mit DB2 Everyplace geliefert werden, importiert oder exportiert.

4.6.1 Darstellung der Daten in einem Klassendiagramm

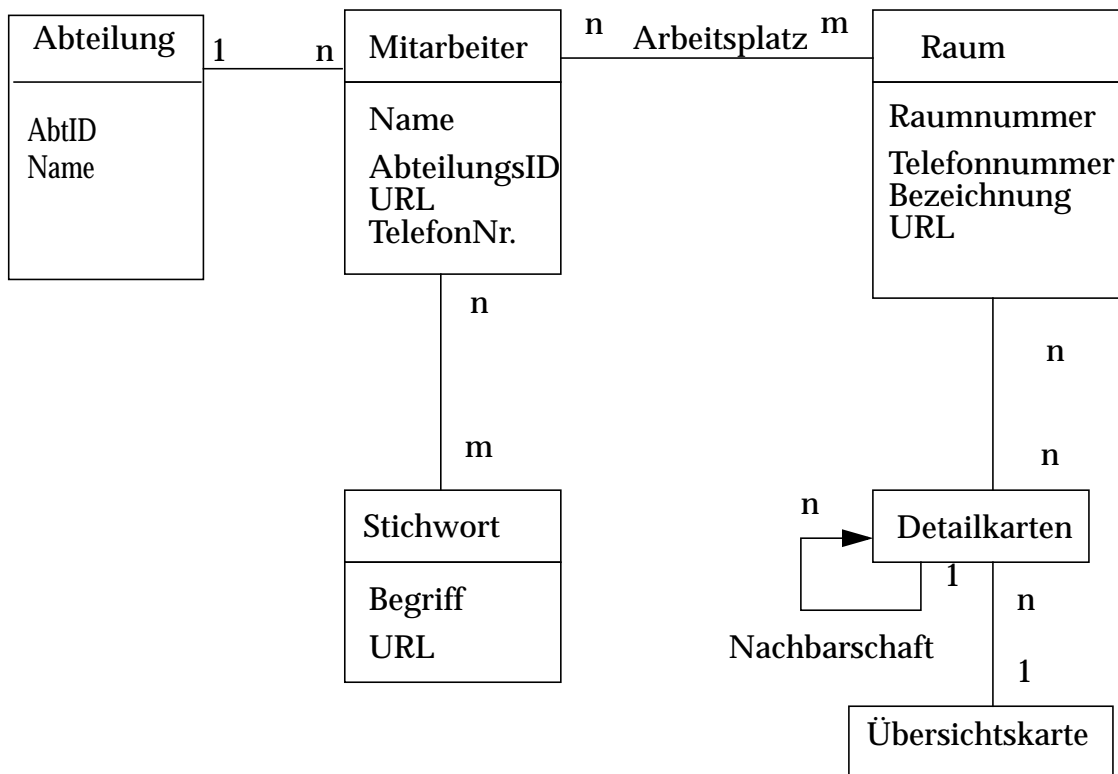


Abbildung 4-2: Klassendiagramm der Daten

4.6.2 Beschreibung der Daten

Mitarbeiter

Mitarbeiter haben einen Namen, eine Raumnummer des Raumes, in dem sie arbeiten, eine oder mehrere Abteilung/en, eine Telefonnummer, Stichwörter, evtl. eine URL. Ein Beispiel für eine Mitarbeiterin, die zu mehreren Abteilungen gehört ist Ursula Mühlbayer. Diese gehört zur Abteilung SE und zur Abteilung PS. (Stand Juli 2000).

Raum

Ein Raum hat eine eindeutige Raumnummer, eine Telefonnummer, Mitarbeiter (0 .. N Stück), eine Bezeichnung (z.B. Hörsaal) und eine URL. Ein Raum ohne Mitarbeiter wäre z.B. ein Hörsaal. Räume mit mehreren Mitarbeitern sind z.B. die Räume der Assistenten.

In der Datenbank werden alle Räume der Fakultät enthalten sein, die ein Telefon haben.

Stichwörter

Stichwörter können beliebige Zeichenketten ohne Leerzeichen sein.

Karten

Es gibt zwei Kartenarten: Übersichtskarten und Detailkarten.

Alle Detailkarten sind durch klicken auf diese Übersichtskarte zu erreichen und alle Detailkarten zusammen sollten einen kompletten Gebäudeplan des Fakultätsgebäudes ergeben.

Übersichtskarte

Die Übersichtskarte des Gebäudes wird alle drei Stockwerke enthalten. Man erhält sie unter [Fakultäts Gebäudeplan 2000]. Es werden aber noch zusätzlich die Treppen eingezeichnet.

Detailkarten

Diese Karten zeigen Gebäudeteile sehr viel genauer als die Übersichtskarte und zeigen die Räume und deren Raumnummer an.

4.7 Zusammenfassung der Anforderungen

Das Programm hat drei Zielgruppen. Die Nexusentwickler und die Entwickler der Abteilung Anwendersoftware sind dabei an technischen Eigenschaften des Produktes interessiert. Die Nexusentwickler benötigen eine gute Änderbarkeit und Wiederverwendbarkeit des Programms. Die Abteilung Anwendersoftware benötigt eine leichte Austauschbarkeit der Datenbank. Zusätzlich sollte ein Benchmark der Datenbank möglich sein. Das Programm sollte leicht um Programmteile erweitert werden können, die mögliche Formen von Anfragen an die Datenbank realisieren. Die dritte Gruppe sind die Anwender. Diese benötigen eine leichte Bedienung des Programms und möchten sich anhand des Programms im Gebäude orientieren können. Sie möchten Informationen zu den Räumen und Mitarbeitern erhalten. Weiterhin sollen sie nach Mitarbeitern suchen können. Sie können diese anhand von Stichwörtern oder Namen suchen lassen. Hierbei soll dann der räumlich nächste Mitarbeiter ausgegeben werden. Weiterhin möchte der Anwender SQL Anweisungen eingeben können, wodurch die Funktionalität des Programms stark erweitert wird. Die Anforderungen an die Daten sind nicht sehr hoch. Es sollten so viele Daten in der Datenbank enthalten sein, dass eine Bewertung der Datenbank sinnvoll ist. Auf eine Konsistenz der Daten muss nicht geachtet werden.

Die folgenden beiden Kapitel sollen zeigen, wie das Programm diese Anforderungen erfüllen soll. Das Kapitel 5 "Programmbeschreibung" beschreibt das Aussehen des Programms und nennt die Beschränkungen des Prototyps. Hierdurch wird die Realisierung der funktionalen Anforderungen gezeigt. Das Kapitel 6 "Der Architekturentwurf" zeigt, wie die technischen Anforderungen durch das Programm erfüllt werden.

5 Programmbeschreibung

In diesem Abschnitt wird das Aussehen der Benutzeroberfläche beschrieben. Es werden die Fenster, die Bedienelemente und die Funktion dieser Bedienelemente beschrieben. Danach werden die Einschränkungen im Prototypen genannt.

5.1 Übersicht über die Fenster

Es gibt 6 Fenster:

- Für die Übersichtskarte
- Für die Detailkarte
- Für die find nearest und global find
- Für die Anzeige von Mitarbeiter- und Raumdaten
- Für die Eingabe von SQL Anweisungen
- Für die Einstellungen des Programms.

Alle Fenster haben ein Menü, mit dem der Benutzer zu jedem anderen Fenster wechseln kann.

5.1.1 Fenster für die Übersichtskarte

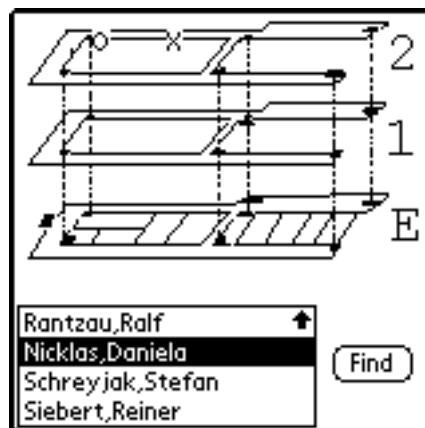


Abbildung 5-1: Die Übersichtskarte

In diesem Fenster sieht man den Gebäudeplan der Fakultät Informatik. Die gestrichelten Linien von einem Stockwerk zum nächsten zeigen die Treppen an. Hat man seine Position markiert, so wird an dieser Position ein X angezeigt. In der unteren Hälfte des Bildschirms sieht man nach einem Aufruf einer der beiden find Funktionen eine Liste der gefundenen Mitarbeiter. Wird einer dieser Mitarbeiter ausgewählt, so wird die Position des Mitarbeiters mit einem O markiert. Man kann auf diesem Bildschirm zum Fenster für die find Funktionen mit Hilfe des 'find' Buttons wechseln. Eine detaillierte Ansicht eines Kartenausschnittes erhält man durch Anklicken eines Kartenausschnittes. Wählt man in der Mitarbeiterliste eine Person aus, so wird deren Position angezeigt.

Dieses Fenster dient zur Erfüllung von Anforderung 2 (Navigation) und Anforderung 10 (Anzeige der Positionen von Mitarbeitern und Benutzern).

5.1.2 Fenster für die Detailkarte

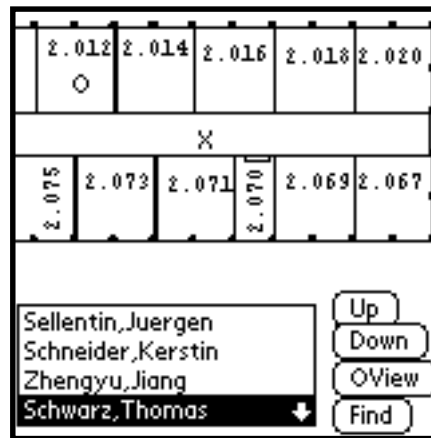


Abbildung 5-2: Die Detailkarte

In dieser Ansicht sieht man in der oberen Bildschirmhälfte einen detaillierten Ausschnitt des Gebäudeplans. Auch hier gilt, dass die eigene Position, falls sie auf diesem Kartenausschnitt ist, mit einem X und die eines Mitarbeiters mit einem O markiert wird. Wenn man einen Mitarbeiter auswählt, dessen Position nicht auf dieser Detailkarte vorhanden ist, so wird auf die Übersichtskarte gewechselt und dort die Position des Mitarbeiters angezeigt.

Wenn man auf einen der Räume auf der Detailkarte klickt, so wird zum Fenster für die Anzeige von Mitarbeiterdaten gewechselt, und die Liste enthält dann die Mitarbeiter in diesem Raum.

Wenn der Anwender auf den find Button klickt, so wird zum Fenster für die Find Funktionen gewechselt.

Man kann auf der Detailkartenansicht auch navigieren. Klickt man auf die Buttons "Up" oder "Down", so wird der Kartenausschnitt ein Stockwerk über bzw. unter dem momentanen Kartenausschnitt gezeigt. Es gibt auch die Möglichkeit in alle vier Himmelsrichtungen zu navigieren. Hierfür muss man an die Ränder der Karte klicken. Bei der Navigation wird überprüft, ob in der gewünschten Richtung eine Karte ist und Falls dort keine Karte ist wird die Anweisung ignoriert.

Mit Hilfe des OView Buttons kann man wieder auf die Übersichtskarte wechseln.

Man kann in diesem Fenster die eigene Position markieren. Hierfür wählt man im Menü Mark Position aus und klickt als nächstes auf die Position auf der Karte. Es werden dabei keinerlei Überprüfungen gemacht, ob diese Position sinnvoll ist.

Dieses Fenster soll Anforderung 2 (Navigation), Anforderung 3 (Markierung der Position), Anforderung 9 (Anzeige der gefundenen Mitarbeiter) und Anforderung 10 (Anzeige der Positionen von Mitarbeitern und Benutzern) realisieren.

5.1.3 Fenster für find nearest und global find



Abbildung 5-3: Das Fenster für global find und find nearest

In diesem Fenster kann der Anwender Suchanfragen für die Funktionen find nearest und global find machen. In der Liste stehen die Stichwörter, die das Programm kennt und den Prefix im oberen Eingabefeld haben. Klickt man auf eines der Stichwörter so erscheint es im unteren Eingabefeld. Hierdurch wird der Aufwand für die Eingabe der Stichwörter reduziert. Gibt der Anwender einen Buchstaben in diesem Prefixfeld ein und drückt den Filter Button, so verändert sich die Liste der Stichwörter und es stehen dort nur noch Stichwörter mit diesem Prefix. Im unteren Eingabefeld stehen die Stichwörter nach denen gesucht werden soll. Der Benutzer kann dort auch eigene Stichwörter angeben. Dort kann man auch die Art der Verknüpfung angeben. Gibt er ein '+' vor ein Stichwort ein, so bedeutet dies, dass nur Mitarbeiter ausgegeben werden sollen, die dieses Stichwort enthalten. Dies entspricht einer AND Verknüpfung. Wenn vor einem Stichwort ein Leerzeichen steht, dann entspricht dies einem 'OR'. Der Mitarbeiter mit diesem Stichwort wird ausgegeben, wenn er alle mit '+' verknüpften Stichwörter enthält.

Neben der Suche nach Stichwörter kann man auch eine Suche nach Mitarbeitern mit einem bestimmten Namen machen. Diese Suchanfrage erstellt man im Prinzip wie die Suchanfrage nach Stichwörtern. Es wird allerdings nur eine OR-Verknüpfung existieren, da das Programm für jeden Mitarbeiter nur einen Namen vergibt.

Die Bedeutung der Worte im Editierfeld legt man durch Anwählen des Pushbutton "names" für Namen oder des Pushbuttons "keywords" für Stichwörter fest.

Die Sucharten lassen sich nicht kombinieren d.h. entweder sucht man nach Stichwörtern oder nach Namen. Beides ist nicht möglich.

Für das Ausführen einer Suchanfrage klickt der Anwender auf find nearest oder den global find Button. Find nearest unterscheidet sich von global find durch die Möglichkeit, die Suche auf Personen bis zu einer bestimmten Entfernung zu beschränken. Nachdem die Anfrage ausgeführt wurde und mindestens einen Treffer hat, wird zu der Detailkarte gewechselt, auf der die Position des ersten Mitarbeiters zu sehen ist.

Dieses Fenster realisiert Anforderung 6 (Stichwortsuche), Anforderung 7 (Räumliche Anfragen) und Anforderung 8 (Mitarbeitersuche über Namen).

5.1.4 Fenster für Daten zu Räumen und Personen

The screenshot shows a window with the following fields and controls:

- Room No. 2.010.....
- Name Schwarz, Holger.....
- Department Anwendersoftware.....
- Phone 244.....
- A text box containing "Anwendersoftware".
- A list box containing "Rantzau, Ralf" and "Schwarz, Holger", with "Schwarz, Holger" selected.
- A "Find" button.

Abbildung 5-4: Fenster für die Mitarbeiter- und Raumdaten

In diesem Fenster werden die Daten zu Mitarbeitern und des Raums ausgegeben. Wie man Abb. 5-4 sieht werden die Raumnummer, der Name des Mitarbeiters die Abteilung, die Telefonnummer und eine Stichwortliste ausgegeben. Die Anzeige verändert sich, sobald man einen anderen Mitarbeiter in der Liste auswählt. Durch Auswählen des Find Buttons wechselt man zum Fenster für global find und find nearest. Die Daten der Mitarbeiter können nicht verändert werden.

Es werden Anforderung 4 (Erhalt von Informationen eines Raums) und Anforderung 5 (Anzeige Mitarbeiterdaten) realisiert.

5.1.5 Fenster für die Eingabe von SQL Anweisungen

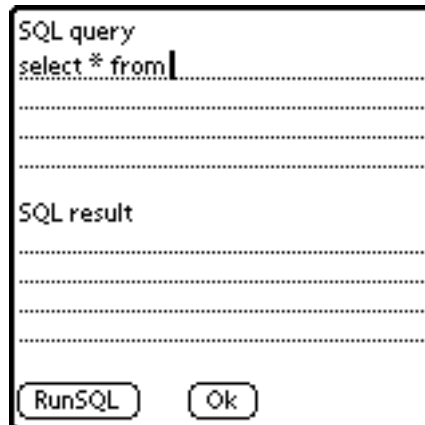


Abbildung 5-5: Das Fenster für die SQL-Anfragen

Im oberen Eingabefeld wird der Benutzer die SQL-Anfrage eingeben können und durch den RunSQL Button ausführen lassen. Im unteren Eingabefeld wird das Ergebnis dieser Query ausgegeben. Das genaue Aussehen dieser Ausgabe ist noch nicht sicher. Die Ausgabe könnte so sein, dass die einzelnen Zellen der zurückgegebenen Tabelle durch das Zeichen ' | ' voneinander abgegrenzt wird. Fehlermeldungen der Datenbank werden dort ebenfalls ausgegeben. Das vorherige Ergebnis wird dabei gelöscht.

Dieses Fenster realisiert Anforderung 11 (Eingabe einer SQL-Anweisung).

5.1.6 Fenster für die Einstellungen des Programms

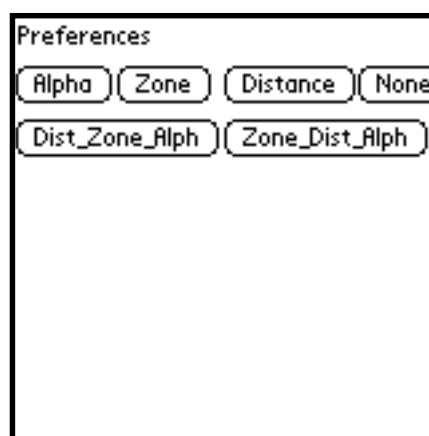


Abbildung 5-6: Fenster für die Einstellungen

Hier kann man auswählen, wie die Mitarbeiter bei einem find nearest sortiert werden sollen. Als Möglichkeiten hat man keine Sortierung, alphabetische Sortierung, Sortierung nach Zone, Sortierung nach Entfernung und komplexere Sortierungen. Bei der Sortierung nach Zonen werden die Mitarbeiter einzelnen Zonen zugeordnet. Die Entfernung dieser Zone zur Zone des Benutzers bestimmt dann die Reihenfolge. Momentan entspricht eine Zone einem Kartenausschnitt. Bei den komplexeren Sortierungen wird beim Vergleich zweier Elemente zuerst das erste Kriterium angewandt. Wenn diese Elemente dabei gleich sind, dann wird das zweite angewandt. Wenn diese Elemente immer noch gleich sind, dann wird das dritte Kriterium angewandt. Es gibt zwei dieser komplexeren Sortierungen. Die erste sortiert zuerst nach Entfernung, dann nach Zone. Die zweite zuerst nach Zone und dann nach Entfernung. Bei beiden werden die Elemente die nicht sortiert werden konnten, noch einmal alphabetisch sortiert. Die Sortierungen zeigen so die unterschiedlichen Möglichkeiten, wie eine find nearest Funktion sich verhalten kann. Mit einer alphabetischen Sortierung oder gar keine Sortierung einzusetzen kann man keine find nearest Funktion realisieren, da der räumliche Bezug fehlt. Es ist so allerdings ein Vergleich der Ergebnisse und Geschwindigkeiten möglich.

Standardmäßig wird beim Aufruf der find nearest Funktion nach Entfernung sortiert.

5.2 Einschränkungen des Prototypen

Der Prototyp unterscheidet nicht zwischen global find und find nearest. Der Unterschied zwischen beiden Funktionen liegt nur darin, dass man das Ergebnis auf Personen innerhalb eines Gebietes einschränken kann. Der Unterschied war für die Bewertung der Datenbank und des Prototypen nicht relevant.

Die Eingabe von SQL-Anweisungen ist ebenfalls nicht möglich. Die Möglichkeit dieses zu testen besteht bereits durch das Programm QBE.

Eine weitere Einschränkung betrifft die Verknüpfungsmöglichkeiten bei der Stichwortsuche. Das '+' bedeutet nicht eine AND Verknüpfung, sondern entspricht der Funktionalität bei Suchmaschinen. Es bedeutet daher, dass das nur Mitarbeiter angezeigt werden sollen, die das Stichwort nach dem Plus-Zeichen haben, und nicht wie bei einer AND-Verknüpfung sowohl das Stichwort vor und hinter dem Zeichen. Dies hat gegenüber einer echten AND-Verknüpfung den Vorteil, dass hierdurch die Semantik der bei Suchmaschinen entspricht und so für viele Anwender intuitiver ist.

6 Der Architekturentwurf

In diesem Kapitel wird die Architektur des Programms vorgestellt. Es werden zuerst die Anforderungen an die Architektur genannt. Danach werden die Architekturen beschrieben, die ich vor der Erstellung des Programms betrachtet habe. Als nächstes wird dann die Architektur des Programms vorgestellt. Daraufhin werden zukünftige Erweiterungen vorgestellt und beschrieben, wie diese die Architektur beeinflussen können.

6.1 Anforderungen an die Architektur

Die Architektur sollte leicht erweiterbar sein, die Funktionen des Programms sollten leicht durch die Funktionen des Nexusservers ersetzt werden können, die Datenbank sollte leicht ausgewechselt werden können und ein Benchmark zwischen unterschiedlichen Datenbanken sollte leicht möglich sein.

6.2 Die betrachteten Architekturen

Es werden zuerst allgemeine Architekturen betrachtet (Palmprogramme, Client/Server, Model-View-Controller) und danach die Nexusarchitektur und die ViLiS-Architektur betrachtet.

6.2.1 Der allgemeine Aufbau eines Palmprogramms

Palmprogramme sind ereignisgesteuerte Anwendungen. Sie erhalten Ereignisse und können auf diese reagieren. Ein Ereignis kann z.B. das Klicken auf den Bildschirm, das Drücken eines Buttons oder das Starten eines Programms sein. Palmrechner besitzen unterschiedliche Arten von Ereignissen, z.B. Systemereignisse, Ereignisse für Menüs etc.

Eine Besonderheit von Palmprogrammen gegenüber Anwendungen auf den Desktops ist, dass Palmprogramme im Vergleich sehr viel weniger Funktionalität besitzen. Ein Grund dafür ist, dass die Programme versuchen den Anwender beim Lösen einer Aufgabe zu unterstützen, und nicht wie z.B. Office Pakete bei vielen unterschiedlichen Aufgaben. Sie müssen darüberhinaus speichereffizient sein. Dieses führt dazu, dass die Zeilenanzahl der Programme relativ gering ist. Aus diesem Grund gibt es oft Programme, die wie die Beispielprogramme zu DB2 Eventplace nur aus einem Modul und einer Headerdatei bestehen.

6.2.2 Das Client Server Konzept

Das Client Server Konzept kennt Server, die Dienste anbieten und Clients, die diese Dienste nachfragen. Die Server können dabei selbst wieder Clients sein, wenn diese zur Realisierung eines Dienstes die Dienste eines anderen Servers benutzen. Die Server kennen dabei ihre Clients nur, solange sie gerade einen Dienst für diese ausführen. Nach der Ausführung des Dienstes vergessen die Server die Clients wieder.

6.2.3 Das Model-View-Controller Konzept

Das Model-View-Controller (MVC) Konzept teilt ein Programm in drei Komponenten auf. Das Model enthält die Daten und alle Funktionen zur Datenverwaltung. Die View ist eine "Sicht" auf Teile dieser Daten. Alle Funktionen für die Darstellung der Daten sind in der View. Die dritte Komponente ist der Controller. Dieser enthält alle Funktionen, die externe Ereignisse (z.B. das Drücken einer Taste, das Auswählen eines Listenelements oder die Veränderungen im Model) überwachen. Diese drei Komponenten kommunizieren miteinander. Drückt der Anwender z.B. einen Button bei dem das Fenster eine neue Karte anzeigen soll, so registriert der Controller den Tastendruck und teilt der View mit, dass diese nun eine neue Karte anzeigen soll. Die View holt sich dann die neue Karte vom Model.

Die Anwendung dieses Konzepts ermöglicht eine leichtere Änderung der Eingabemechanismen und der Oberfläche. Es ist auch sehr viel einfacher neue Fenster für die Darstellung der Daten zu erstellen als bei einem Programm, bei dem Daten und Oberfläche nicht getrennt sind, da eine Schnittstelle zu den Daten vorhanden ist. Ein Problem besteht zwischen der hohen Kopplung zwischen View und Controller. Fügt man ein neues Bedienelement hinzu, so muss im Normalfall der Controller ebenfalls geändert werden. Löscht man dieses Bedienelement wieder, so muss auch die Funktion für dieses Bedienelement im Controller gelöscht werden. Fasst man wie die ViLiS Entwickler View und Controller zusammen, entgeht man dem Problem.

6.2.4 Die Nexusarchitektur

Die hier beschriebene Nexusarchitektur stammt aus [Nicklas 2000].

Allgemeiner Aufbau

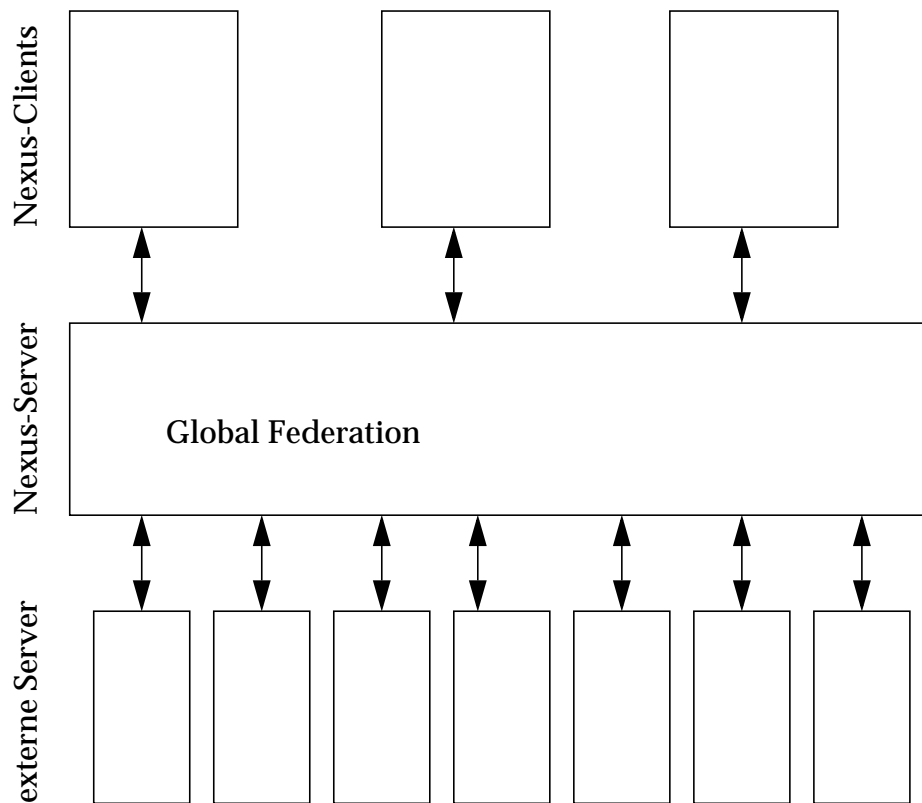


Abbildung 6-1: Nexusarchitektur

Die Nexusarchitektur ist eine Client/Server-Architektur. Sie besteht aus drei Hauptkomponenten: Nexus-Client, Nexus-Server und externe Server. Diese drei Komponenten befinden sich auf unterschiedlichen Rechnern. Nexus-Clients sind Clients, die Nexusdienste benutzen. Alle Nexusdienste zusammen bilden die Global Federation, die sich auf den Nexusservern befindet. Möchte der Client Nexusdienste benutzen, so baut er eine Verbindung zu einem Nexus-Server auf und ruft danach die gewünschten Nexusdienstleistungen auf. Die Nexus-Server holen sich bei Bedarf Daten oder benutzen Dienste externer Server.

Der Nexus-Client

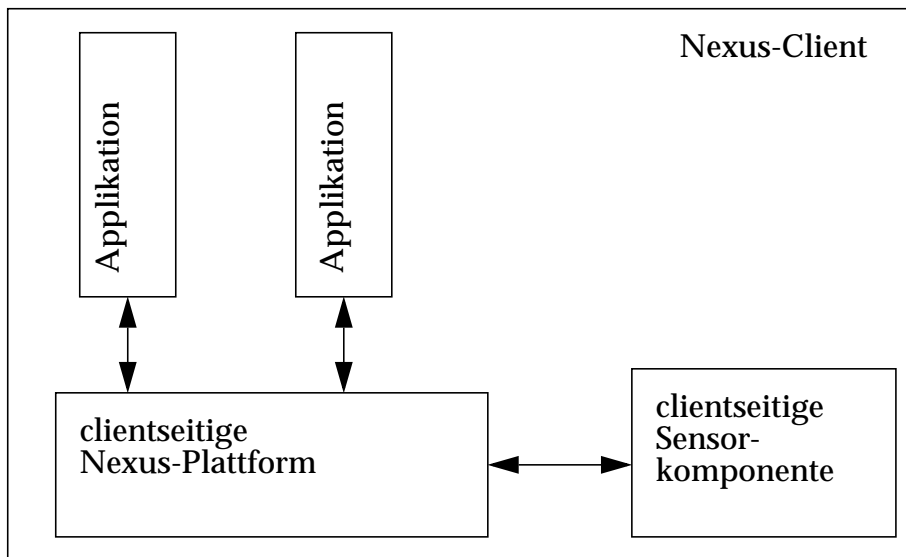


Abbildung 6-2: Nexus-Client-Architektur

Der Nexusclient besitzt Anwendungen, die Nexusdienste nachfragen, eine clientseitige Nexus-Plattform und eine Sensorkomponente. Die clientseitige Nexus-Plattform dient als Schnittstelle der Applikationen zu den Nexusdienstleistungen. Sie ist für die Kommunikation zwischen dem Nexus-Client und den Nexus-Servern zuständig. Die Sensorkomponente dient zur Positionsbestimmung des Clients und kommuniziert deshalb mit der clientseitigen Nexus-Plattform.

Der Nexus-Server

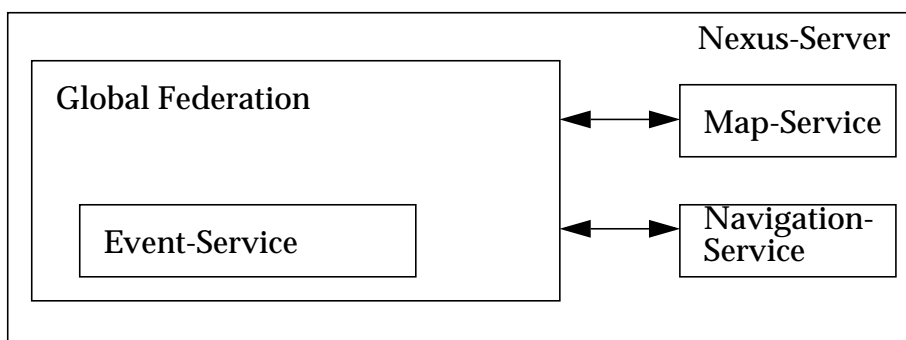


Abbildung 6-3: Nexus-Server Architektur

Der Nexus-Server besteht aus der Global Federation, einem Event-Service, einem Map-Service und einem Navigation-Service.

Die Global Federation bietet alle Nexusdienstleistungen an. Sie kann die Dienstleistungen aber nur durch Hilfe anderer Dienste erbringen, da der Nexus-Server keine Daten speichert. Diese können sich auf dem Nexus-Server befinden wie z.B. Map-Service oder Navigation-Service oder auf externen Servern sein.

Der Map-Service ist für die Erzeugung von graphischen Karten aus Daten zuständig, die von unterschiedlichen Nexusdiensten gesammelt werden.

Der Event-Service ist ein Teil der Global Federation und benachrichtigt Anwendungen von Ereignissen, die z.B. durch mobile Objekte ausgelöst wurden.

Die externen Server

Die externen Server bieten folgende Dienste an:

- Der Lokation Service speichert die Position, die Richtung und andere Daten über mobile Objekte
- Der Spatial Model Service speichert Informationen über statische Realwelt- und virtuelle Objekte
- Der Name Service wandelt herkömmliche Objektnamen in eindeutige Objekt-IDs um. Diese können benutzt werden um Informationen zu bestimmten Objekten von anderen Diensten zu erhalten
- Das Area Service Register speichert, welche Server für welche Gebiete zuständig sind
- Der External Data Service enthält die Daten zu Objekten, die nicht die räumlichen Eigenschaften eines Objektes betreffen
- Die Sensor Komponente bietet Positionsinformationen an. Diese Informationen werden durch Sensoren erstellt.
- Durch die Actuator Komponente können Komponenten der realen Welt gesteuert werden

Eigenschaften der Architektur

Die Architektur ist darauf ausgelegt, dass das System verteilt sein wird. Es ist deshalb ein Client/Server System.

Durch den Nexus-Client ist der Zugriff auf Nexusdienste aus Sicht der Applikationen zugriffstransparent, verteilungstransparent und ortstransparent. Durch die Global Federation ist der Zugriff auf Nexusdienste für den Nexus-Client verteilungstransparent. Dieses führt dazu, dass sich die Nexusarchitektur betreffend der externen Server ändern kann, ohne dass der Nexus-Client davon betroffen ist.

Eignung der Architektur für das Programm

Da das Programm auf diese Architektur migriert werden soll, dient sie als Ausgangspunkt für die Architektur des Programms. Die Funktionalität, die durch das Nexussystem erreicht werden soll, ist allerdings sehr viel größer als die Funktionalität innerhalb des Programms. Hierbei sollte beachtet werden, dass die Nexusarchitektur von einem verteilten System ausgeht, während das Programm sich auf

einem Palm befindet und erst später Programmteile durch Dienste des Nexussystems ersetzen wird. Bei einer kompletten Übernahme der Nexusarchitektur hätte man deshalb viele unnötige Komponenten. Diese unnötigen Komponenten sind das Area Service Register, Sensor Component, Actuator Component, Name Service und Lokation Service. Da der Palm keinen Sensor hat, wird die Positionsbestimmung des Benutzers durch Benutzereingaben gemacht, wodurch auch die clientseitige Sensorkomponente überflüssig ist. Der Event Service ist unnötig, da das Programm keine Nexus-Events erzeugen oder verarbeiten kann. Das führt dazu, dass nur noch der Map Service, der Navigation Service, Spatial Model Service und der External Data Service Aufgaben haben, die das Programm selbst implementieren muss.

Die clientseitige Nexuskomponente und der Nexus-Server sind als getrennte Komponenten nur in einem verteilten System sinnvoll. Wenn man beide in ein Programm einbaut leitet der Nexus-Client die Funktionsaufrufe einfach an den Nexus-Server weiter. Die beiden Komponenten lassen sich deshalb zusammenfassen. Bei einer Migration auf die Nexusplattform muss dieses allerdings wieder rückgängig gemacht werden.

6.2.5 Die ViLiS Architektur

Das ViLiS-System

Virtual Information Towers (VITs) oder auf deutsch virtuelle Litfasssäulen (ViLiS) sind nach [Leonhardi et al 1999] ein Konzept für die Darstellung und den Zugriff auf ortsabhängige Informationen, die an einen bestimmten Ort gebunden sind und eine bestimmte Sichtbarkeit haben. Ähnlich wie bei Litfasssäulen können Anwender diese Informationen nur sehen, wenn diese von ihrer Position aus sichtbar sind.

Das ViLiS-System realisiert dieses Konzept und ist nach [Nicklas 2000] der erste Prototyp der Nexus Plattform.

Interessant ist dabei wie die Architektur aussieht und wie der Aufbau des ViLiS Client ist. Die Informationen zum Aufbau des Systems entstammen aus [ViLiS System Entwurf 2000].

Architektur

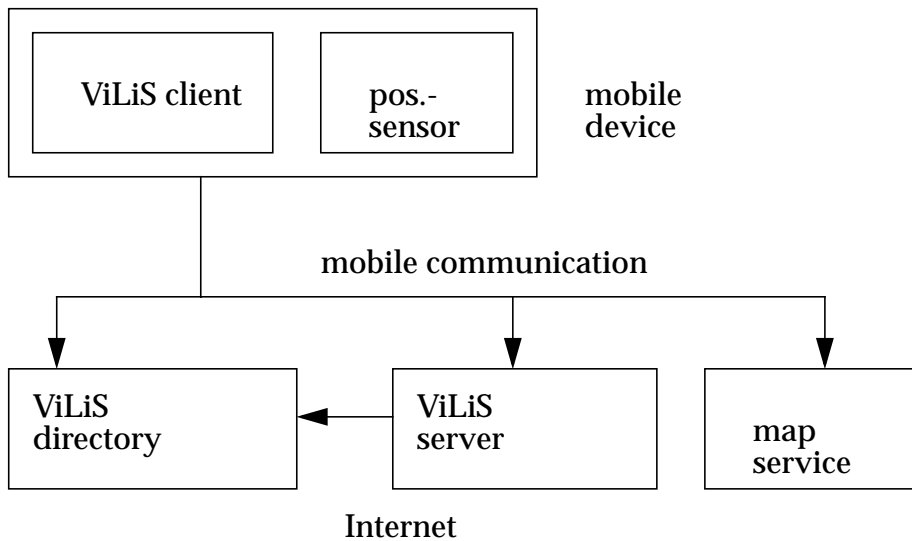


Abbildung 6-4: ViLiS-Architektur

Abbildung 6-4 zeigt die ViLiS Architektur ohne die Komponenten für die Administration des Systems.

Das ViLiS System besteht aus einem ViLiS Client auf einem mobilen Gerät, aus ViLiS Directory Servern, ViLiS Servern und Map Servern.

- Der ViLiS Client läuft auf einem mobilen Gerät und ermöglicht seinen Benutzern den Zugriff auf das ViLiS System. Durch einen Positionierungssensor bestimmt der Client seine Position. Der ViLiS-Client zeigt die sichtbaren VITs an
- Das ViLiS Directory enthält die Links zu allen verfügbaren VITs und wird benutzt, um die VITs zu finden, die von seiner Position aus sichtbar sind.
- Die ViLiS Server basieren auf Webservern. Sie verwalten das Inhaltsverzeichnis und die Plakate für einen oder mehrere VITs. Außerdem registrieren sie die VITs beim ViLiS Directory
- Der Map Service liefert die Karte zu einer Position.

Möchte der ViLiS Client Daten erhalten, so fragt dieser beim ViLiS Directory nach den dafür zuständigen ViLiS Servern. Der Client lädt danach die Daten von den ViLiS Servern und erhält Karten zur Region vom Map Service.

Eigenschaften dieser Architektur

Die Architektur ist eine Client/Server Architektur. Das ViLiS Directory ist aufgrund der Verteilung des Systems notwendig.

Der Client muss den Aufbau des Systems kennen, da es keine Verteilungstransparenz der ViLiS Funktionen gibt. Änderungen der Funktionalität und der Vertei-

lung der Funktionen unter den ViLiS Diensten führen dazu, dass auch der ViLiS Client geändert werden muss. Die Änderbarkeit und die Erweiterbarkeit des Systems ist deshalb im Vergleich zur Nexusarchitektur eingeschränkt.

Eignung der ViLiS Architektur

Da das Programm nicht verteilt ist, ist ein ViLiS Directory unnötig.

Da der Client direkt auf die einzelnen Komponenten zugreifen muss, gibt es bei der Migration auf die Nexusplattform wahrscheinlich Probleme.

Der Client benötigt in der Regel drei Anfragen, um alle Daten zu erhalten, die er benötigt, da er eine Anfrage benötigt, um die Links zu den ViLiS Servern zu erhalten, eine weitere Anfrage, um Daten von diesen Servern zu erhalten und eine Anfrage für die Karte. Es ist allerdings geschickter für die Erstellung eines Programms, wenn man für das Ausführen eines Dienstes nur eine Prozedur aufrufen muss und nicht mehrere. Die Dienste des Programms sollten deshalb so aufgebaut sein, dass sie aus Sicht des Clients keine Kommunikation über mehrere Module erfordern. Diese Dienste können selbst wieder auf andere Dienste zugreifen. Dieses sollte transparent für den Client bleiben.

ViLiS Client Architektur

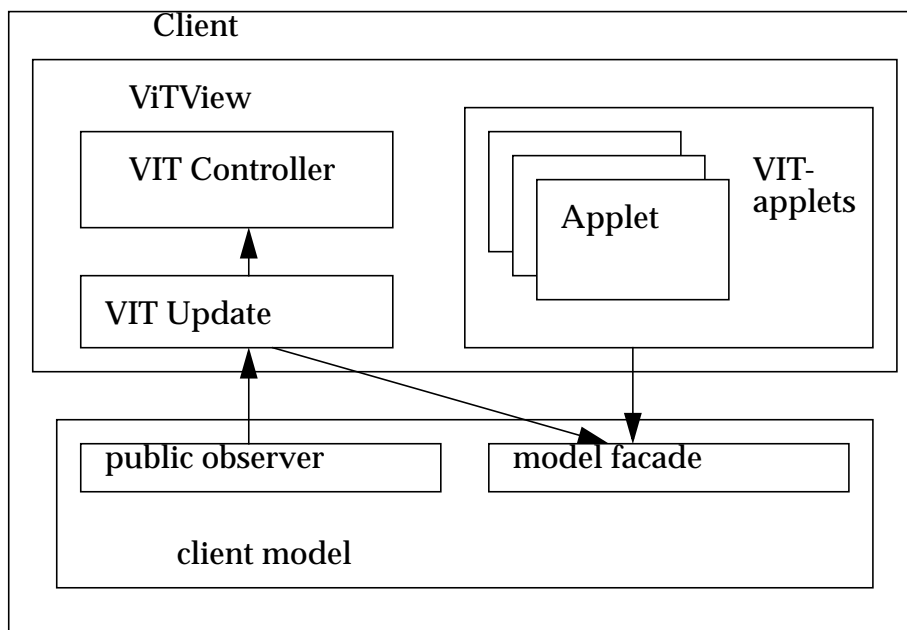


Abbildung 6-5: ViLiS Client Architektur

Die Architektur des Clients ist eine abgewandelte Form einer Modell View Controller Architektur. Der Controller ist Teil der View. Die Darstellung erfolgt in einem Webbrowser, der Java Applets darstellt. Eine besondere Eigenschaft der Architektur ist, dass Multithreading im Client benutzt wird. Dadurch müssen Änderungen durch eine View auch den anderen Views bekanntgemacht werden.

Die Änderung des Modells erfolgt durch die Model Facade. Der Public Observer bemerkt die Veränderungen am Modell und benutzt die VIT Update Komponente, um die Views zu ändern.

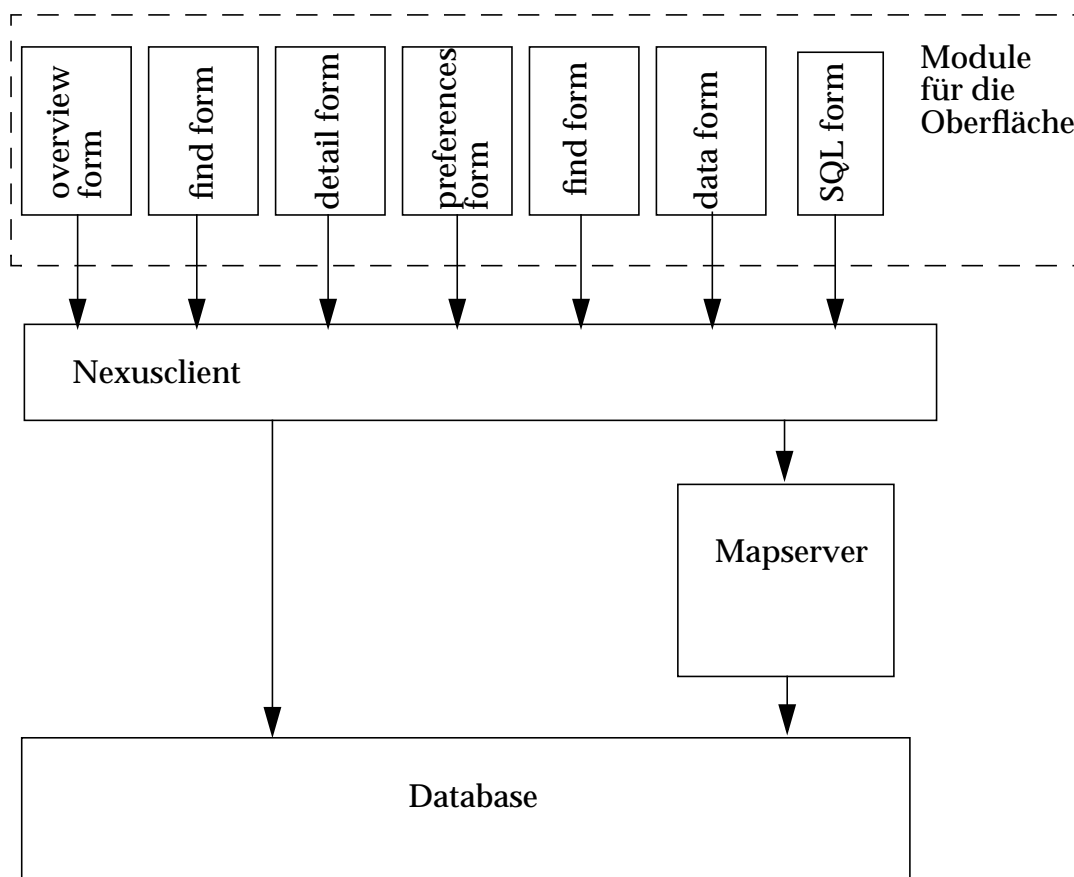
Eigenschaften der ViLiS Client Architektur

Die Oberfläche ist leicht änderbar. Der Client benutzt mehrere Threads und mehrere Views. Dadurch müssen diese Views bei Änderungen des Modells durch eine View aktualisiert werden. Der Controller ist in der View enthalten. Die Komponenten für die Datenspeicherung und Kommunikation werden nicht explizit aufgeführt und sind im Model enthalten.

Eignung der ViLiS Client Architektur

Die Architektur zeigt eine Möglichkeit, wie man die Oberfläche von den Daten trennt und so Daten und Funktionalität von der Darstellung der Daten trennt. Eine Änderung der View ist beim Verschieben von Funktionalität vom Client zum Server nicht nötig, da dieses aus Sicht des Clients völlig transparent geschieht. Das Zusammenlegen von View und Controller erleichtert dabei die Änderbarkeit der Oberfläche, da so Änderungen nur in einem Modul nötig sind.

6.3 Die Architektur des Programms



A —> B A ruft Funktionen von B auf

Abbildung 6-6: Programm-Architektur

In Abbildung 6-1 sieht man die Architektur des Programms. Es gibt Module für die Oberfläche, das Modul Nexusclient, das die Daten der Oberfläche enthält und die Dienste der darunterliegenden Module anbietet, das Modul Mapserver für alle Funktionen, die mit räumlichen Daten zu tun haben, und das Modul Database, das die Daten verwaltet. Nicht auf der Abbildung sind die Module mit den Datentypen für die Kommunikation der Programmteile, da sonst das Bild zu unübersichtlich wird. Das Hauptprogramm, das beim Programmstart aufgerufen wird, ist ebenfalls nicht auf dem Bild, da es nur für die Initialisierung und Deinitialisierung benutzt wird.

6.3.1 Module für die Oberfläche

Jedes Modul für die Oberfläche entspricht einem Fenster im Programm. Da diese Fenster bei Palmprogrammen "Forms" heißen, haben die Module das Wort Form in ihrem Namen. Sie enthalten die View und die Teile des Controllers, die für die Eventbearbeitung dieses Fensters zuständig sind. Die dargestellten Daten sind im Nexusclient gespeichert.

6.3.2 Das Modul Nexusclient

Das Modul stellt die Grenze zwischen Programmteilen, die bei einer Migration auf dem Nexusclient bleiben, und Programmteilen, die bei einer Migration durch Dienste des Nexusserver ersetzt werden sollen.

Diese Modul hat nicht die gleiche Funktionalität wie der Nexusclient in der Nexusarchitektur! Es hat mehrere Funktionen. Es dient zur Speicherung der Daten für die Module der Oberfläche und übernimmt dadurch die Funktion des Models im MVC-Konzept. Es bietet die zur Verfügung stehenden Funktionen an, wodurch es die Aufgabe der clientseitigen Nexuskomponente und der Nexus-Server bei der Nexusarchitektur übernimmt. Wie der Nexusserver erledigt es die Dienste nicht selbst, sondern greift hierfür auf andere Module (Mapserver, Datenbank) zu. Die Integration dieser Funktionen war möglich, da die Funktionalität der clientseitigen Nexuskomponente (Kommunikation mit Nexusserver) und des Nexus-Servers (Aufruf der Dienste externer Server) innerhalb dieses Programms sehr klein ist.

6.3.3 Das Modul Mapserver

Dieses Modul übernimmt die Funktionalität des Mapservice, Navigation Service, Location Service und Spatial Model Service. Es besitzt alle Funktionen, die mit geographischen Daten zu tun haben. Das sind z.B. die find nearest und die global find Funktionen, die Positionsbestimmung für die Mitarbeiter und Funktionen, die den Erhalt und die Navigation auf den Karten betreffen.

6.3.4 Das Modul Database

Die Module Nexusclient und Mapserver rufen Funktionen dieses Moduls auf, um Daten zu erhalten. Der genaue Aufbau der Architektur sieht man in 'Aufbau der Datenbank' auf Seite 46.

6.3.5 Module für die Kommunikation

Die Datentypen für die Kommunikation sind in eigenen Modulen abgespeichert. Der Zugriff auf diese Datentypen erfolgt dabei wie bei abstrakten Datentypen über Prozeduren.

6.3.6 Ursprünge dieser Architektur

Das Vorbild für die Architektur des Programmes ist die Nexusarchitektur. Die einzelnen Teilmodule lassen sich so leichter durch Dienste der Nexusplattform ersetzen. Eine Trennung zwischen Programmteilen, die auch nach einer Migration vom Programm gemacht werden, und Programmteilen, die von der Nexusplattform ersetzt werden, ist so leicht möglich. Hierbei wurden Nexusdienste zusammengefasst, die in der Nexusarchitektur auf mehreren Servern verteilt gewesen sind und Nexusdienste weggelassen, die aufgrund der Verteilung des Systems existieren.

Ähnlich wie beim ViLiS Client wurden die Funktionen der Oberfläche von denen der anderen Programmteile getrennt, damit es eine bessere Änderbarkeit der Oberfläche gibt. Es existiert für jedes Fenster ein eigenes Modul, wodurch man einfacher einzelne Fenster verändern oder in einem anderen Programm wiederverwenden kann. Ähnlich wie beim ViLiS Client wurde die Aufteilung zwischen View und Controller fallengelassen.

Das Modul Nexusclient bietet alle Dienste den Modulen für die Oberfläche an. Es ermöglicht ähnlich wie der Nexus-Server in der Nexusarchitektur eine beliebige Verteilung der Daten und Nexusdienste, die vom Programm benutzt werden, wodurch eine vereinfachte Migration und Änderung der Nexusdienste möglich wird.

6.4 Aufbau der Datenbank

6.4.1 Grobaufbau der Datenbank

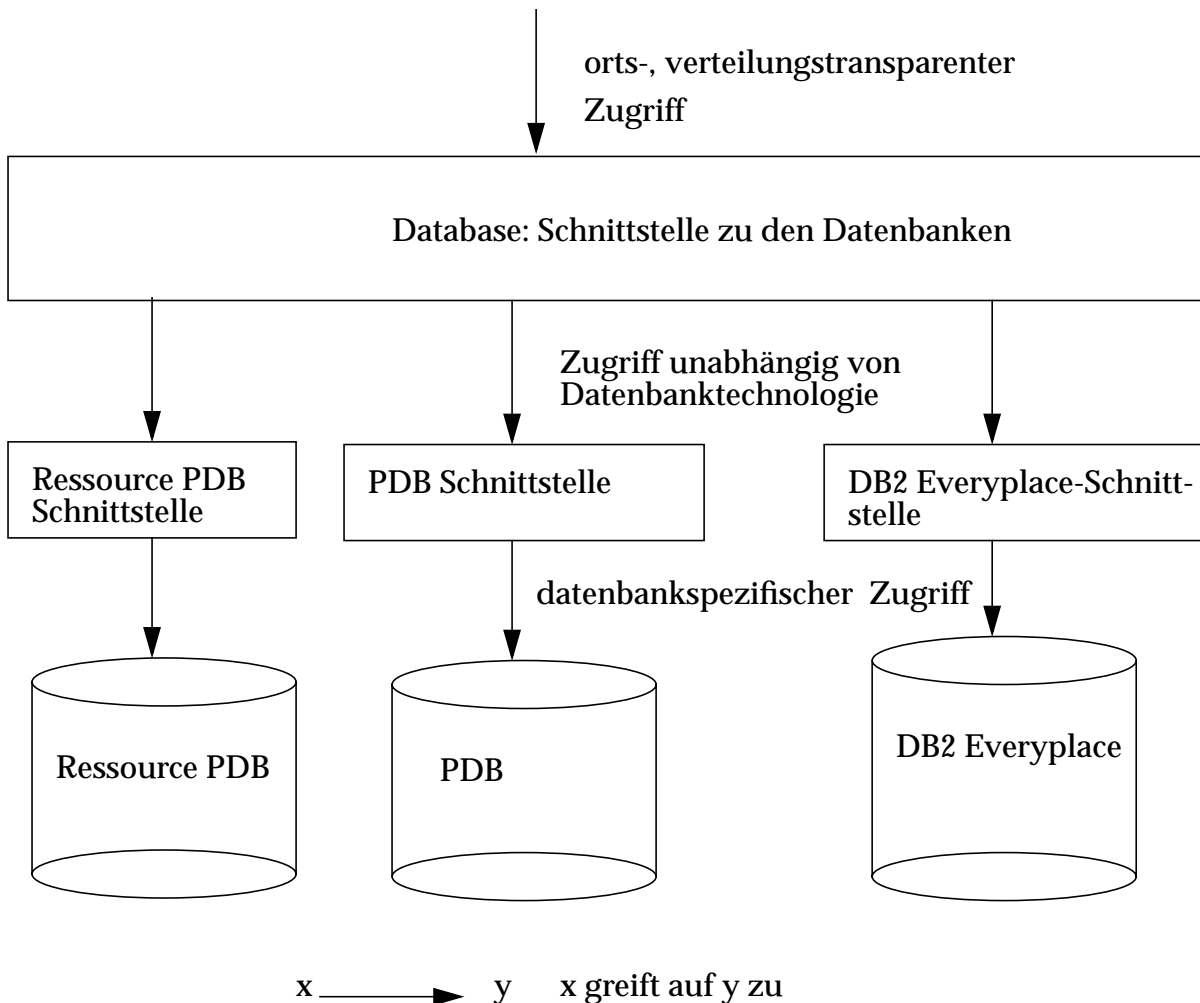


Abbildung 6-7: Aufbau der Datenbank

Die Datenbank besteht aus einer Datenbankschnittstelle, und Schnittstellen zu den möglichen Aufbewahrungsorten der Daten und besitzt daher Schnittstellen zu PDB, Ressourcen PDB und DB2 Everyplace.

Die Datenbankschnittstelle erhält alle Anfragen von externen Komponenten und leitet die Anfragen an die richtige Komponente weiter. Dazu wird die passende Anfrage in der Schnittstelle der Komponente aufgerufen. Die Schnittstelle wandelt die Anfrage in datenbankspezifische Anfragen um.

6.4.2 Datenbankschnittstelle

Sie ist die Schnittstelle anderer Programmteile zu den Daten. Sie kapselt die Daten und stellt Funktionen zum Zugriff auf die Daten zur Verfügung. Die Funktionen orientieren sich an den Bedürfnissen der Anwendung und sind deshalb unabhän-

gig von der eingesetzten Anfragen an die Datenbank. Hierdurch wird eine einfache Veränderbarkeit der Daten und der Anfragen ermöglicht.

Ihre Funktionen können Daten benötigen, die in der DB2 Everyplace Datenbank, der Ressourcen PDB oder in einer PDB und in zukünftigen Erweiterungen sogar in mehreren PDBs enthalten sind. Diese Komponente sorgt bei Anfragen über mehrere Datenbanken für die richtige Sequenz der Anfragen. Aus den Rückgaben der einzelnen Datenbanken wird dann die Rückgabe an die Anwendung erstellt.

6.4.3 Ressourcen PDBs

Die Ressourcen PDB speichert die Ressourcen eines Programms. Ressourcen sind Zeichenketten, Buttons, Menüs, Bitmaps etc. . Diese Daten werden an das Programm gelinkt und können deshalb nur mit Hilfe einer Neukompilation des Programmes verändert werden. Man erzeugt Ressourcen PDBs mit Hilfe von Ressourcencompilern, die eine Textdatei einlesen und daraus eine Ressourcen PDB erzeugen, die von Linkern zum Programm dazugelinkt werden. Da die Ressourcen zum Programm dazugelinkt werden, hängt die Größe des Programms sehr stark von der Größe der Ressourcen ab. Bei jedem HotSync-Vorgang werden die Daten und der Programmcode übertragen, auch wenn sich die Daten nicht geändert haben. Hierdurch kann es zu langen HotSync Vorgängen kommen.

Programme greifen auf die Daten in einer Ressourcen PDB über ähnliche Funktionen wie bei einer normalen PDB zu.

6.4.4 PDB Dateien

Palmprogramme speichern ihre Daten normalerweise in PDB Dateien. PDB Dateien sind leicht austauschbar und können auf einem Desktoprechner durch Synchronisation gesichert werden. Der Zugriff ist im Vergleich zu herkömmlicher Dateibehandlung relativ mächtig: z.B. gibt es Funktionen zur Sortierung einer PDB und Funktionen zur Suche nach Elementen. Anfragen, die mehrere PDBs betreffen, müssen allerdings von der Anwendung implementiert werden.

Der Zugriff auf PDB Datenbanken ist wie bei Dateien nicht verteilungs-, replikations- oder zugriffstransparent.

6.4.5 DB2 Everyplace Datenbankkomponente

Diese ist eine relationale Datenbank. Der Zugriff erfolgt dabei durch SQL-Anweisungen, die über eine ODBC Schnittstelle gesendet werden. Im Vergleich zu den anderen beiden Speicherungsmöglichkeiten hat sie die mächtigsten Zugriffsmöglichkeiten. Zur Geschwindigkeitssteigerung kann man Indexe anlegen, wobei in [DB2E Development Guide 2000] wenig Information zur Art der Indexe vorhanden ist.

6.5 Die Verteilung der Daten

Es werden fast alle Daten in der DB2 Everyplace Datenbank gespeichert. Nur die Bitmaps für die Karten werden nicht in der DB2 Everyplace Datenbank gespeichert. Es gab zwei Gründe hierfür.

Der erste Grund lag an den Programmen zum Import der Daten, die zwar Daten als BLOBs einlesen können, allerdings funktionierte dieses im Praxistest nicht.

Der zweite Grund ist die Tatsache, dass ich die Bitmaps nicht zur Laufzeit in die Datenbank speichern konnte. Hierbei war nicht die Datenbank das Problem, sondern der Zugriff auf Bitmaps unter Palm OS. Man kann Bitmaps relativ leicht in die Ressourcen PDB speichern und so in ein Programm hineinkompilieren. Zum Zugriff werden Pointer eingesetzt. Bitmaps bestehen dabei aus zwei Teilen. Der erste Teil enthält allgemeine Daten zum Bitmap z.B. Höhe, Breite, Anzahl der Bits pro Bildpunkt etc. Der zweite Teil enthält die Daten. Unter Palm OS zeigt ein Pointer nur auf den ersten Teil des Bitmaps. Man kann zwar mit Hilfe des ersten Teils über eine Betriebssystemfunktion auch den zweiten Teil des Bitmaps erhalten. Jedoch fand ich keine Funktion, die aus diesen beiden Teilen wieder ein Bitmap erstellt.

Die Bitmaps konnten auch nicht in einer PDB Datei gespeichert werden. Da ich kein Programm habe, das Bitmaps in PDB Dateien speichert, müsste ich auch hier die Bitmaps zur Laufzeit in eine PDB Datei eintragen.

Die Bitmaps befinden sich deshalb in der Ressourcen PDB. Dies ist für einen Prototypen des Programms akzeptabel. Bei einem Endprodukt sollten die Bitmaps entweder in einer PDB Datei oder in DB2 Everyplace gespeichert werden, damit eine Änderung der Karten zur Laufzeit möglich ist.

6.5.1 Eigenschaften der Datenbank

- Der Zugriff von der Datenbankschnittstelle auf PDB, Ressourcen PDB oder DB2 Schnittstelle ist ortstransparent und verteilungstransparent im Bezug auf die Verteilung der Daten innerhalb einer Datenbank, allerdings verteilungsintransparent im Bezug auf die Verteilung zwischen den Datenbanken. Es gibt keine Zugriffstransparenz beim Zugriff auf die Daten, da die Prozeduren in den Schnittstellen zur PDB, Ressourcen PDB oder DB2 Everyplace keine gleichen Namen haben dürfen, selbst bei gleichem Verhalten
- Der Zugriff der Datenbankschnittstelle auf die einzelnen Datenbanken ist unabhängig von der eingesetzten Datenbanktechnologie
- Neue Datenbanken lassen sich leicht in das Programm einbauen, da diese nicht direkt, sondern über eine Schnittstelle angesprochen werden und die Datenbankschnittstelle diese leicht benutzen kann
- Die Verteilung der Daten lässt sich relativ leicht ändern, indem bei den dazugehörigen Datenbanken nur die Schnittstellen eine neue Funktion erhalten und der Ablauf der Zugriffssequenz in der Datenbankschnittstelle geändert werden muss

6.5.2 Eigenschaften dieser Architektur

Die Verteilung und der Ort der Nexusdienste ist für die Module der Oberfläche transparent. Der Mapserver und die Datenbank lassen sich leicht wiederverwenden. Der innere Aufbau der Datenspeicherung ist nach außen hin transparent und aus Sicht der Anwendung ist der Zugriff auf die Daten zugriffs-, orts- und verteilungstransparent. Der Benchmark von Datenbanken lässt sich leicht implementie-

ren, da die Programmteile außerhalb der Datenbank nicht geändert werden müssen.

6.6 Mögliche zukünftige Erweiterungen

6.6.1 Benchmark der Datenbank

Ein Benchmark der Datenbank lässt sich leicht durchführen. Die Datenbankschnittstelle ruft dabei die Funktionen der zu testenden Datenbank über deren Schnittstelle auf und misst die Zeit für die Ausführung der Funktion. Diese Funktionen haben in normalen Programmen Rückgabewerte, deren Erstellung ebenfalls Rechenzeit kostet. Möchte man die Datenbankgeschwindigkeit ohne die Zeit für die Erstellung der Rückgabewerte messen, so sollten die benutzten Funktionen Anfragen an die Datenbank machen, aber kein Ergebnis erstellen oder zurückliefern. Das Ergebnis des Benchmarks sollte dann ebenfalls ausgegeben werden. Die Ausgabe kann dabei auf dem Bildschirm erfolgen oder in eine Datei oder Datenbank geschrieben werden. Wenn die Ausgabe aufwendig werden sollte, ist es dabei sinnvoll hierfür ein eigenes Modul zu erstellen.

6.6.2 Einfügen anderer Datenbanken

Hierfür muss man eine Schnittstelle zu dieser Datenbank definieren, die einen datenbankspezifischen Zugriff auf die Daten realisiert. Nachdem dieses gemacht wurde, muss man das Modul "Datenbankschnittstelle" so ändern, dass sie die Funktionen der neuen Datenbank aufruft. Die Änderung der Datenbank erfolgt transparent für die restlichen Komponenten des Programms.

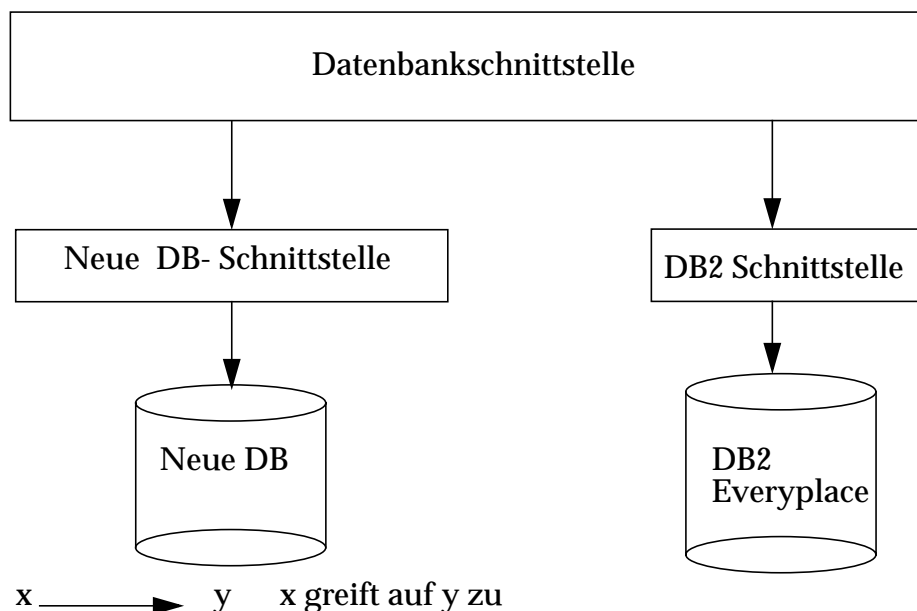


Abbildung 6-8: Einfügen einer neuen Datenbank

6.6.3 Zugriff auf externe Server

Es gibt aus Sicht des Programms unterschiedliche Arten von Diensten, die externe Server anbieten können. Die externen Server können den Zugriff auf externe Daten erlauben, Funktionen anbieten, die Funktionen im Programm ersetzen können oder Funktionen anbieten, die vom Programm zusätzlich zu den bestehenden benutzt werden können. Soll nur auf externe Daten zugegriffen werden, so erzeugt man ein Modul, das die Kommunikation mit dem Server übernimmt. Es wird dann als eine neue Datenbank eingefügt. Sollen Funktionen des Programms ersetzt werden, so kann man den Zugriff auf diese Funktionen über das dafür zuständige Modul machen. Momentan käme dafür nur das Modul Mapserver in Frage. Wenn man die Funktionalität des Mapservers von externen Servern realisiert werden soll und der Zugriff auf diese Funktionen nicht aufwendig ist, so kann man den Nexusclient direkt darauf zugreifen lassen. Der Zugriff auf ergänzende Funktionen des Programms kann dadurch realisiert werden, indem man den Nexusclient diese Funktionen direkt aufrufen lässt. Der Aufruf von Funktionen, die die Funktionalität des Moduls Mapserver ergänzen, kann man auch dort einbauen. Wenn es sich um viele Funktionen handelt, kann man auch ein neues Modul erstellen, das die Dienste dieses externen Servers anbietet.

6.6.4 Einbau eines Augmented World Model

Ein Augmented World Model (= AWM) ist ein Modell eines Gebietes. Dieses Modell enthält Objekte für reale Objekte, aber auch für virtuelle Objekte, die sich in diesem Gebiet aufhalten. Sie definiert, welche Daten zu den einzelnen Objekten gespeichert werden. In der Nexusarchitektur ist diese AWM verteilt über die einzelnen Server. Eine genauere Beschreibung über das Augmented World Model wird man in [Meßmer 2001] erhalten können.

Der Einbau einer AWM in das Programm würde zu einer expliziten Darstellung der benutzten Objekte der realen Welt führen. Der Aufbau der Daten für diese Objekte würde hierdurch ebenfalls sichtbar gemacht. Bei einer geschickten Implementierung könnte so eine leichtere Änderung der Daten der Objekte ermöglicht werden. In der jetzigen Implementierung werden die Objekte der realen Welt nicht explizit dargestellt. Das Wissen darüber verteilt sich über das ganze Programm. Welche Objekte modelliert werden ist nicht offensichtlich und Änderungen fehleranfällig, da dabei mehrere Bereiche des Programms betroffen sind.

6.6.5 Migration auf die Nexusarchitektur

Bei einer Migration auf die Nexusarchitektur wird die Datenspeicherung, die Erzeugung der Karten und die geographischen Algorithmen von den Nexusservern bereitgestellt. Die Datenbankkomponente und die Komponente für die geographischen Funktionen (Map Server) werden überflüssig. Die Kommunikation zwischen dem Client und den Nexus-Servern wird dann von einer Komponente außerhalb des Programms übernommen. Der Zugriff auf diese Komponente wird dann vom Modul Nexusclient übernommen. Dieses wird dabei die Umwandlung der Daten und dem Benutzen der richtigen Funktionen implementieren. Wenn dieses sehr aufwendig ist, kann es sein, dass die Komponente Nexusclient des Programms in zwei Teile zerlegt wird. Das erste für das Model des Programms und das zweite für den Aufruf der externen Komponente.

6.7 Zusammenfassung

In diesem Kapitel wurden mehrere Architekturen betrachtet. Als erstes wurden allgemeine Architekturen betrachtet und danach wurde der Aufbau zweier komplexerer Architekturen betrachtet. Die Nexusarchitektur ist ein Beispiel für eine Plattform, die ortsabhängige Systeme unterstützen soll. Diese Architektur hatte einige Komponenten, die im Programm nicht notwendig sind. Die zweite betrachtete Architektur, war die ViLiS Architektur. Hierbei wurden die Komponenten betrachtet, die Dienste für ortsabhängige Informationen bereitstellen, und der Aufbau der Clients betrachtet, die auf die Dienste zugreifen. Danach wurde die Architektur des Programms vorgestellt und beschrieben, wie die Architektur von den anderen Architekturen beeinflusst wurde. Zum Schluss wurde beschrieben, wie zukünftige Erweiterungen möglich wären und wie diese die Architektur beeinflussen würden.

Betrachtet man das Kapitel, so fällt auf, dass der Aufbau des Programms abgesehen von den Komponenten für die Speicherung und den Namen der Module für die Oberfläche nichts wirklich palmspezifisches enthält. Hierdurch könnte man zum Schluss kommen, dass sich der Entwurf eines Palmprogramms nicht von dem Entwurf eines Desktopprogramms unterscheidet. Dieses liegt daran, dass die Eigenschaften der Palms d.h. Größe, Rechenleistung, Speicherverbrauch, Bedienung etc. nicht den Grobentwurf und damit die Architektur eines Programmes beeinflussen, sondern den Feinentwurf und damit die Algorithmen, den Aufbau der Oberfläche, das Aussehen von Oberflächenelementen etc. Deswegen wird im folgenden Kapitel der Kartenentwurf beschrieben, bei dem sich die palmtypischen Eigenschaften bemerkbar machen.

7 Kartenentwurf

In diesem Kapitel wird der Kartentwurf beschrieben. Zuerst wird der Einsatzzweck und die Unterschiede von Karten beschrieben. Danach werden die Kartenarten beschrieben, die für die Entwicklung der Karte in Frage kamen und verglichen. Danach wird das Format ausgewählt, das zur Speicherung der Karten eingesetzt wird. Als letztes wird beschrieben, wie das Programm die Zuordnung zwischen Kartendaten und Sachdaten realisiert.

7.1 Karten

Karten sind Modelle eines geographischen Gebietes und werden bereits seit Jahrtausenden von Menschen eingesetzt, um sich eine bessere Vorstellung von geographischen Gegebenheiten zu machen. Sie werden sowohl zur Modellierung großer geographischer Gebiete als auch relativ kleiner Gebiete eingesetzt. So gibt es Karten für Länder, Kontinente und sogar für so große Gebiete wie das Weltall, aber auch Karten für Gebäude oder Gebäudeteile.

7.1.1 Einsatzzweck von Karten

Die Eigenschaften eines Modells der realen Welt hängen davon ab, welchen Zweck mit diesem Modell verfolgt wird. Karten zu einem geographischen Gebiet können ein unterschiedliches Aussehen haben, je nach dem wofür diese eingesetzt werden sollen.

Typische Einsatzzwecke für Karten sind:

- Hilfe zur Orientierung z.B. für Wanderer
- Modell für einen geographischen Raum, der verändert werden soll z.B. für den Bau eines Hauses in einem Gebiet
- Entscheidungen sollen anhand von geographischen Daten getroffen werden z.B. bei einer Routenplanung
- nichträumliche Daten wie z.B. die Regenmenge sollen für geographische Gebiete dargestellt werden.

7.1.2 Die Unterschiede bei Karten

Karten existieren in vielerlei Form und unterscheiden sich in verschiedenen Aspekten:

- Dimension: Karten sind normalerweise zweidimensional. Bei Angabe von Höhe spricht man von 2D+1 Karten. Hierbei wird oft Farbe benutzt um Höheninformation darzustellen. Durch den Einsatz von Computern ist auch die Darstellung von dreidimensionalen Karten möglich
- Informationstypen: Karten können sich dabei in Anzahl und Art der Informationstypen unterscheiden
- Maßstab und Detailgrad beeinflussen wieviele Details eines Gebietes dargestellt werden

- Darstellung wichtiger und unwichtiger Information: Karten können wichtige und unwichtige Information gleich behandeln, wichtige Informationen hervorheben oder unwichtige Information weglassen
- Abbildung der Größe. Karten können die Größe der Objekte maßstabsgetreu übernehmen und dabei einen oder mehrere Maßstäbe benutzen. Die Größe der Kartenelemente kann wie bei Skizzen oder Schatzkarten auch völlig ohne Maßstab erstellt sein

7.1.3 Anforderungen an die Karte des Programms

Nachdem die wichtigsten Unterscheidungsmerkmale zwischen Kartenarten aufgezeigt sind, sollten nun die von mir betrachteten Alternativen bewertet werden. Hierfür müssen die Anforderungen an die Karten geklärt werden.

Der Anwender wird mit diesem Programm Informationen zu Räumen erhalten können. Er wird sich die Position von Mitarbeitern anzeigen lassen. Er muß auch seine eigene Position auf der Karte markieren können (für find nearest). Da er zu den Mitarbeitern auch hingehen will, sollte die Karte auch eine Orientierungshilfe für ihn darstellen. Diese Karte soll auf einem Palm dargestellt werden können.

Aus diesen Anforderungen lassen sich folgende Anforderungen ableiten:

Informationen zu bestimmten Räumen erhalten:

- die Räume mit Informationen müssen auf der Karte vorhanden sein
- der Anwender muss auf der Karte erkennen können, welchen Raum er anwählt

Position von Mitarbeitern/Anwender

- die Karte muss groß genug sein, um Positionsmarkierungen zuzulassen d.h die Räume und der Flur müssen groß genug sein
- durch die Karte sollte der Anwender seine eigene Position erkennen
- eine reale Position muss auf die Karte eindeutig abbildbar sein

Orientierungshilfe

- die Position eines Mitarbeiters sollte gefunden werden können
- die wichtigen Objekte müssen gut erkennbar sein, da eine schnelle Orientierung möglich sein muss
- Es sollte ein schnelles Auffinden der Räume möglich sein

Darstellung auf dem Palm

- die Karte sollte wenig Speicher verbrauchen
- die Karte sollte schnell aufgebaut werden können
- der Palm hat einen kleinen Bildschirm, mit geringer Auflösung. Die Karte muss aber trotzdem gut lesbar sein
- die Karte sollte so viel vom Bildschirm ausnutzen, wie möglich
- die Karte ist schwarz-weiss

7.1.4 Die vorhandenen Karten

Die vorhandenen Karten sind im FIG-Format (Standardformat von Xfig) und im GIF-Format. Die Karten im GIF-Format sind im Internet verfügbar und entstan-

den aus den Karten im FIG-Format. Das FIG-Format ist ein Vektorformat. Das GIF-Format ist ein Rasterformat.

Die Karten sind nicht überschneidungsfrei. Die GIF-Dateien haben unterschiedliche Auflösungen. Diese Bilder haben große Auflösungen (ca 900x600 Punkte) und waren für große Monitore bestimmt.

Die Räume sind ziemlich rechteckig und fast alle haben Nummern oder Bezeichnungen. Es werden Details wie Türen, Aufzüge und Kopierer gezeigt.

Es ist kein Maßstab angegeben. Deshalb kann ich nicht sagen, ob diese maßstabsgetreu sind oder nicht.

7.1.5 Alternative für das Aussehen der Karte

Im wesentlichen kamen drei Lösungsmöglichkeiten in Frage: Eine sehr vereinfachte Karte, eine maßstabsgetreue Karte und eine Karte, die so konvertiert wurde, dass sie gut auf den Bildschirm des Palms passt.

Alternative 1: Eine sehr vereinfachte Karte

Die Karte ist eine stark vereinfachte Darstellung des Gebäudes. Es werden nur die interessanten Räume angezeigt. Die Räume sind so groß, dass sie die Raumnummern enthalten können. Ihre Größe hat deshalb auch nichts mit ihrer realen Größe zu tun. Die Abbildung 7-1 zeigt eine solche Karte. Für Durchgänge wird einfach eine Lücke zwischen zwei Räumen gelassen.

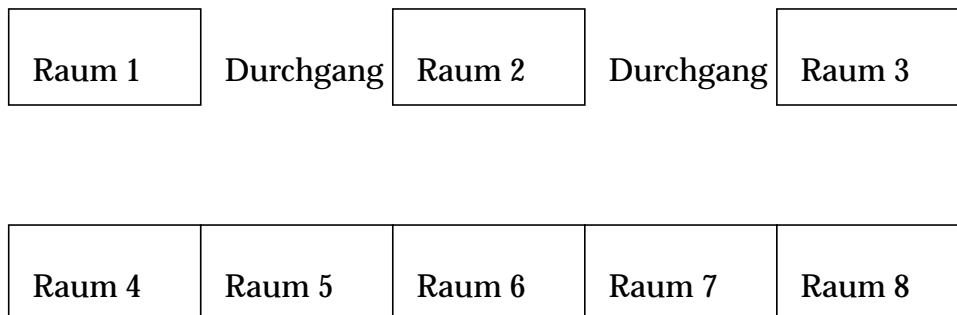


Abbildung 7-1: Eine vereinfachte Karte

Die Eigenschaften einer solchen Karte sind:

- nur die wichtigen Details werden dargestellt
- besonders wichtige Kartenelemente können besonders hervorgehoben werden
- die Größe der Kartenobjekte ist unabhängig von der Größe der realen Objekte
- sehr vereinfachte Darstellung der geographischen Situation
- keine Lücken, die nur Platz verschwenden
- keine eindeutige Positionierung
- relativ kleiner Speicherverbrauch, da für jede Karte nur die Reihe mit den Raumnummern und Durchgängen gespeichert werden müssen

- gut zoombar, auch wenn es kaum Sinn macht
- lassen sich leicht durch einen Computer erstellen
- schnelle Bearbeitung eines Klicks auf die Karte möglich, wenn man die Größe aller Räume und Durchgänge gleich macht
- die Größe der Karte und damit die Anzahl der Kartenteile hängt von der Anzahl der Räume und Durchgänge ab. Es kann dadurch passieren, dass ein Gebäudeteil mehr Kartenelemente enthält als ein gleich großer Gebäudeteil mit wenigen Räumen

Bewertung der Karte:

Die Vorteile der Karte sind in erster Linie technischer Natur. Sie verbraucht fast keinen Speicher, lässt sich gut auf Palms darstellen und Klicks auf der Karte sind aufgrund der Regelmäßigkeiten der Karte sehr schnell ausgewertet. Sie benötigt deshalb auch keinen Speicher für die Information über die Bereiche, die auf der Karte angewählt werden werden können.

Der Anwender würde beim Einsatz einer solchen Karte aber ziemliche Probleme haben. Das erste Problem ist die Schwierigkeit sich mit solchen Karten zu orientieren. Solche Karten können nicht zur Abschätzung von Entfernungen benutzt werden, da ihre Größe ja nichts über den Bereich aussagt, den sie abdecken. Die Navigation auf der Karte kann problematisch werden. Wie wechselt man von einer Karte, die so groß ist, dass sie als Nachfolger zwei Karten enthält? Das größte Problem besteht darin, dass eine eindeutige Positionierung nicht möglich ist. Man betrachte die Abbildung 7-2. Links ist ein Ausschnitt einer maßstabstreuen Karte, auf der die Position des Benutzers zu sehen ist, rechts die Karte, die daraus erzeugt würde. Das X markiert die Position des Benutzers.

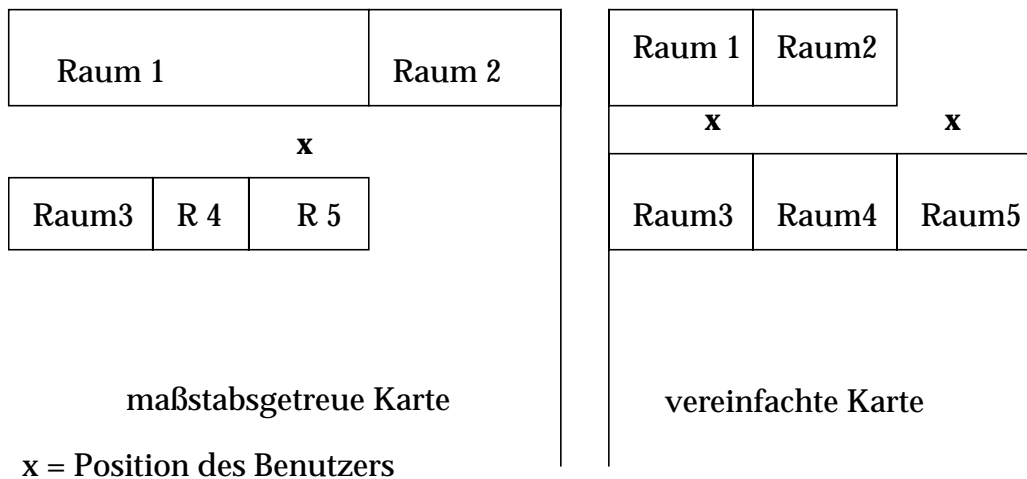


Abbildung 7-2: Probleme bei der Positionierung

Auf der maßstabstreuen Karte erkennt man, dass der Benutzer zwischen den Räumen 1 und 5 steht. Auf der rechten Karte kann man die Position aber nicht eindeutig übertragen. Es gibt dort keine Position zwischen Raum1 und Raum 5. Der Benutzer kann seine Position also vor Raum 1 oder vor Raum 5 markieren.

Für die Anwendung ist es allerdings sehr wichtig, dass der Benutzer seine Position eindeutig positionieren kann. Diese Karten sind deshalb für das Programm ungeeignet.

Alternative 2: Maßstabsgetreue Karte

Die Kartenteile der Karte erhält man, wenn man die ursprünglichen Karten, um einen bestimmten Maßstab verkleinert und sie dann aufsplittet, damit man noch alles erkennen kann. Maßstabsgetreue Karten haben folgende Eigenschaften:

- die Ursprungskarten sollten maßstabsgetreu sein
- der Maßstab der Ursprungskarten oder die Daten des Originals sollten bekannt sein, da sonst Berechnungen mit diesen Karten nicht möglich sind
- durch das Benutzen eines Maßstabes stimmen die Größenverhältnisse
- eine Länge auf der Karte lässt sich in eine reale Länge umwandeln
- enthält alle Räume, aber es bekommen evtl. nicht alle einen Namen
- Entfernungen lassen sich anhand der Punktabstände bestimmen
- gute Orientierung möglich, da Räume unterschiedlich groß dargestellt werden und alle Räume eingezeichnet sind.
- eindeutige Positionierung möglich
- Abstände abschätzbar
- die Navigation ist einfach
- Probleme bei der maßstabsgetreuen Darstellung auf einem Palm
- Probleme bei der Aufteilung der Karten
- Nachbearbeitung der ursprünglichen Karten notwendig z.B. Neuerstellen der Raumnummern durch Raumnummern, Verkleinern der Karte etc.
- Die Auswertung eines Klicks auf der Karte dauert länger
- Speicher für die auswählbaren Gebiete notwendig
- leere Flecken enthalten

Bewertung:

Diese Kartenart ist recht gut geeignet, um in meinem Programm eingesetzt zu werden. Der Anwender kann sich anhand dieser Karte gut orientieren, da sämtliche Räume eingezeichnet sind. Entfernungen lassen sich anhand der Karte nicht nur abschätzen, sondern sogar berechnen. Die Positionierung ist eindeutig. Die Navigation ist leicht zu realisieren, wenn man die Karte in gleich große Teile zerlegt, da es eindeutig ist welcher Kartenteil angrenzt und beim Navigieren angezeigt werden soll.

Es gibt allerdings Probleme beim Erstellen der Karten. Will man das Gebäude maßstabsgetreu auf dem Palm darstellen, so braucht man eine Karte mit einem Maßstab. Ob die ursprünglichen Karten maßstabsgetreu sind, ist unklar. Es gibt noch weitere Probleme. Da die Karte auf einem Palmbildschirm dargestellt werden soll, muß man das Seitenverhältnis der Karten an das Seitenverhältnis des Palmbildschirms anpassen und zwar so, dass die Karten nicht verzerrt werden. Hierfür braucht man auch das Seitenverhältnis, in dem die ursprünglichen Karten angezeigt werden sollten, was ebenfalls nicht vorhanden ist.

Ein Nachteil gegenüber Alternative 1 ist, dass es auch viele unbenutzte Gebiete auf der Karte gibt, die im Endeffekt nur Platz verbrauchen. Weiterhin müssen die Karten so konvertiert werden, dass sie maßstabsgetreu dargestellt werden. Dieses führt dazu, dass diese den Bildschirm nicht optimal ausnutzen können, da sie ein anderes Seitenverhältnis haben. Dieses würde Platz auf dem Bildschirm verschwenken. Da der Palmbildschirm sehr klein ist, kann diese Platzverschwendung dazu führen, dass die Raumnummern nicht mehr dargestellt werden können. Man kann nun entweder den angezeigten Ausschnitt vergrößern und erhöht so die Anzahl der Kartenteile oder man lässt die Raumnummern weg. Beide Alternativen haben ihre Nachteile. Die erste sorgt dafür, dass man auf einer Karte nur sehr wenig vom Gebäude sieht, was schlecht für die Orientierung ist. Die andere Alternative ist noch schlechter für die Orientierung des Benutzers. Nun muss sich der Benutzer anhand der Entfernung zu anderen Räumen und anhand der Form des Raumes seine Position bestimmen. In diesem Gebäude ist das eher schwierig, da die Räume sich stark ähneln.

Beim Aufteilen der Karte entstehen ebenfalls Probleme. Es kann passieren, dass genau an der Stelle, an der die Karte aufgeteilt werden soll, ein kleiner Raum ist, der dann auf jeder Karte nur zur Hälfte vertreten ist. Wenn diese Raumteile dann zu klein sind, um eine Beschriftung zu ermöglichen, dann hat man ein ziemliches Problem. Man kann die Größe eines Kartenteils nicht einfach so kleiner machen und das angrenzende Kartenteil dafür größer machen, da dann ein Kartenteil zu groß für die Bildschirmdarstellung wäre. Auch ist es dann nicht mehr eindeutig, welche Kartenteile aufeinander folgen. Verschiebt man die Bruchstellen der Karte, um die Kartenteile auf dem Palmbildschirm anzeigen zu können, so verstärkt sich dieser Effekt evtl. noch durch weitere verschobene Bruchstellen. Das Ergebnis sind überhängende Kartenteile. Lässt man deshalb vorsorglich für solche Fälle Platz auf dem Bildschirm, so hat man wieder Platz auf dem Bildschirm vergeudet.

Alternative 3: Karte, die durch Verkleinerung der Originalkarten mit anschließender Nachbearbeitung entstand

Die Kartenteile erhält man, indem man die Originalkarten auf eine bestimmte Auflösung konvertiert. Danach werden diese Kartenteile aufgesplittet. Hierbei werden diese Kartenteile an Stellen aufgesplittet, die sich für die spätere Orientierung gut eignen könnten. Diese Kartenteile sind in unterschiedlichen Auflösungen vorhanden und werden nun so vergrößert oder verkleinert, dass sie auf dem Palmbildschirm die optimale Größe haben. Die Kartenart hat folgende Eigenschaften:

- enthält alle Räume, es bekommen aber evtl. nicht alle einen Namen
- Entfernungen lassen sich nur noch abschätzen, nicht mehr exakt berechnen, da es keinen Maßstab gibt
- gute Orientierung möglich, solange man die Karten nicht zu stark verzerrt
- Seitenverhältnis stimmt oft nicht, aber der optische Eindruck ist noch ungefähr der gleiche
- die Kartenteile lassen sich relativ einfach erstellen
- das Aufteilen der Karte ist relativ einfach
- eindeutige Positionierung möglich
- die Navigation ist einfach

- die Auswertung dauert länger als bei Alternative 1
- zusätzlicher Speicherverbrauch für die Auswahlflächen
- Nachbearbeitung der Originalkarten notwendig, ist aber einfacher als bei Alternative 2
- Darstellung auf dem Palm ohne Ränder, da die Karten an die Auflösung des Palms angepasst werden
- enthält viel Leerraum, aber weniger als Alternative 2

Bewertung:

Diese Alternative hat gegenüber der Alternative 2 einige Vorteile, aber auch ein paar Nachteile.

Die Vorteile sind die leichtere Erstellung der Karten und die bessere Anpassung an einen Palmbildschirm. Auch sind die Probleme beim Aufteilen einer Karte viel geringer als bei der Alternative 2. Man teilt die Karten so auf, dass der Benutzer alle Räume gut sehen kann. Dadurch wird die eine Karte zwar größer als die andere, da man diese Karten aber an die Auflösung des Palmbildschirms anpasst, gibt es nicht diesen Fortpflanzungseffekt, wie bei Alternative 2. Bei der Navigation auf diesen Karten kann man immer noch das Problem haben, dass die Navigation nicht ganz stimmt.

Als Nachteil ist der fehlende Maßstab anzusehen, da dadurch Entfernungen und das Aussehen der Räume nicht so gut abzuschätzen sind, wie bei Alternative 2. Die Orientierung mit diesen Karten ist deshalb etwas schlechter. Da die Karten aber nicht zu stark verzerrt werden, kann man sich mit diesen Karten immer noch gut orientieren.

7.1.6 Begründung für meine Lösung

Die Alternative 1 ist interessant, da sie von allen Alternativen, die beste Raumausnutzung auf dem kleinen Palmbildschirm verspricht. Da der Palmbildschirm recht klein (6cm x 6cm) ist, ist diese Eigenschaft ein großer Pluspunkt gegenüber den anderen Karten. Zu Beginn der Studienarbeit war es nicht einmal sicher, ob sich die Originalkarten auf dem Palmbildschirm sinnvoll darstellen lassen. Es war unklar, ob sich die vorhandenen Karten so aufspalten liessen, dass man die Räume samt Raumnummer erkennen konnte und dabei ein ausreichend großes Gebiet dargestellt würde. Diese Befürchtung stellte sich als unbegründet heraus. Da diese Kartenart eine eindeutige Positionierung nicht ermöglicht und die Orientierung mit diesen eher schlecht ist, kommen für einen Einsatz nur die Alternativen 2 und 3 in Frage.

Alternative 2 und 3 sind für die Orientierung in etwa gleich gut geeignet. Maßstabsgetreue Karten sind zwar besser als leicht verzerrte Karten für die Abschätzung von Entfernungen, aber der Unterschied ist eher gering. Auch die Abstände sind aus Sicht des Anwenders in etwa gleich gut zu ermitteln, da ein Anwender die Abstände nicht messen wird, sondern nur abschätzen wird. Mit beiden Karten kann man eine eindeutige Positionierung machen.

Die Alternative 2 hat gegenüber Alternative 3 den Vorteil, dass man mit Hilfe dieser Karten Entfernungen exakt berechnen kann. Bei Alternative 3 wechselt der Maßstab zwischen den einzelnen Karten. Eine Entfernungsmessung mit Hilfe der Karte ist deshalb fehlerhaft. Durch die Art der Erstellung wird dieser Fehler bei Alternative 3 klein gehalten, da die aufgesplitteten Karten zusammen wieder die Größe der ursprünglichen Karte erhalten und so alle zwei Karten, der Fehler wird gleich null ist.

Dafür ist der Aufwand für die Nachbearbeitung bei Alternative 2 viel größer als bei Alternative 3, da man die Seitenverhältnisse des Palmbildschirms und der Originalkarten berücksichtigen muss. Die Ergebnisse dieser Konvertierung dürften auch den Aufwand für die Nachbearbeitung der Karten erhöhen. Auch die fehlende Flexibilität beim Aufspalten der Karten könnte zu Problemen führen, die den Nachbearbeitungsaufwand erhöhen.

Da die Platzverschwendung auf dem Palmbildschirm klein gehalten werden sollte und die Karten relativ schnell erstellt werden sollten wurde die Alternative 3 gewählt.

7.2 Das Kartenformat

In diesem Kapitel werden Vektorformate und Rasterformate beschrieben. Dabei gebe ich dann die prinzipiellen Vor- und Nachteile an und werde dann die Anforderungen an das Kartenformat beschreiben. Daraus begründe ich meine Wahl für den Prototypen und gebe an, inwiefern diese Entscheidung bei einem echten Produkt anders hätte fallen können.

Es gibt zwei prinzipielle Formate in denen die Karten abgespeichert werden könnten:

- Das Vektorformat
- Das Rasterformat (Bitmaps)

Beim Vektorformat besteht ein Bild aus einer Menge von Objekten. Wenn man ein Bild ansehen will, das in einem Vektorformat gespeichert ist, so müssen alle Objekte gezeichnet werden. Bilder im Rasterformat, wie z.B. Bitmaps entstehen dagegen aus vielen Bildpunkten, die gemeinsam eine Art Raster bilden. Daher kommt auch der Name für das Format.

Das Programm benutzt das Bitmapformat für die Speicherung der Karten. Ein Vektorformat hat zwar Vorteile wie eine Verwendbarkeit unabhängig von der Auflösung des Ausgabemediums, einen geringeren Speicherverbrauch und eine leichtere Filterung von Objekten auf der Karte, aber dafür benötigt der Aufbau eines Bitmapbildes sehr viel weniger Zeit und wird vom Palm Betriebssystem im Gegensatz zum Vektorformat unterstützt. Ich hätte beim Vektorformat deshalb eine Bibliothek für das Laden und Darstellen der Vektordaten schreiben müssen. Da der Speicherverbrauch der Karten nur 2 KB beträgt und für die Darstellung auf dem Palm optimiert ist, sind die Vorteile eines Vektorformats gegenüber dem Bitmapformat so gering, dass der Aufwand, um ein Vektorformat einsetzen zu können, sich nicht gelohnt hätte.

7.3 Zuordnung von Kartenobjekten zu Sachdaten

Wenn der Benutzer auf der Karte auf ein Objekt klickt (z.B. einen Raum) so muss nun diese Information mit den Sachdaten verknüpft werden.

Verknüpfungsmöglichkeiten

Die möglichen Verknüpfungsmöglichkeiten sind:

- Gemeinsame Speicherung der beiden Informationsarten in einer Tabelle
- Die Informationen werden getrennt gespeichert. Ein "Schlüssel" verweist auf die andere Informationsart und sorgt so für die Verknüpfung

Die Speicherung in einer gemeinsamen Tabelle hat mehrere Nachteile, insbesondere widerspricht es der Anforderung des modularen Aufbaus und der Erweiterbarkeit. Deswegen werden getrennte Tabellen verwendet, die Information wird über Fremdschlüsselbeziehungen verknüpft.

Die Fremdschlüsselbeziehungen lassen sich auf mehrere Arten realisieren:

- Die Schlüssel in der Tabelle für Positionsinformation verweisen auf ein Element in der Sachdatentabelle. Die Sachdatentabelle erhält keine Schlüssel auf die Positionstabelle
- Die Schlüssel in der Tabelle für Sachdaten verweisen auf Elemente in der Positionstabelle. Die Positionstabelle enthält keine Verweise auf die Sachdaten.
- Je ein Schlüssel verweist in den Tabellen auf die jeweils andere Tabelle
- Man benutzt eine Fremdschlüsseltabelle. In dieser werden die Schlüssel der Positionsinformation zu Schlüsseln der Sachdatentabelle zugewiesen.

Möchte man entscheiden was für Verknüpfungen gewählt werden sollen, so muss man die Beziehung zwischen den beiden Tabellen betrachten.

An einer Position können sich auf einer Karte ein oder gar kein Kartenobjekt befinden. Ein Kartenobjekt kann aber durchaus auf mehreren Kartenteilen vorhanden sein (z.B. Hörsaal 20.01).

Enthält die Sachdatentabelle Verweise auf die Positionstabelle, so wäre sie in erster Normalform, da es mehrere Positionen auf den Karten zu einem Raum gibt. Sie sollte deshalb keine Verweise auf die Position enthalten.

Enthält die Positionstabelle Verweise auf die Sachdatentabelle, so hat man Probleme mit der Erweiterbarkeit der Anwendung, sobald eine Position auf mehrere Sachdaten verweisen soll, so muss man eine Denormalisierung vornehmen. Möchte man auf mehrere Sachdatentabellen verweisen z.B. auf eine Tabelle für Personen und eine für Räume, so muss man der Tabelle eine weitere Spalte geben. Man bekommt Probleme mit Nullelementen, wenn es Positionen gibt, auf denen nur Elemente einer Sachdatentabelle existieren. Eine Erweiterung diesbezüglich ist sehr wahrscheinlich, da man sicherlich nicht nur die Position von Räumen speichern möchte.

Der Einsatz einer Fremdschlüsseltabelle ermöglicht es eine n zu n Beziehung zwischen Positionsinformationen und Sachdaten zu realisieren. Durch den Einsatz mehrerer solcher Tabellen, kann man eine Verknüpfung der Positionsinformation mit mehreren Sachdatentabellen erreichen ohne das Schema der Positionstabellen ändern zu müssen. Deswegen werden sie im Programm auch zur Verknüpfung von Sachdaten und Positionsdaten eingesetzt.

Finden der Position eines bestimmten Kartenobjekts

Das Kartenobjekt hat einen Primärschlüssel. Dieser wird benutzt, um in der Fremdschlüsseltabelle den Primärschlüssel der Positionsdaten zu erhalten. In der Positionsinformationstabelle wird dann nach der dazugehörigen Positionsinformation gesucht.

Finden eines Kartenobjektes anhand von Positionsinformation

Beim Anklicken der Karte auf dem Palm wird ein Event ausgelöst. Dieser Event erhält die Koordinaten, an denen der Anwender auf den Bildschirm geklickt hat. Das Programm kann diesen Event auswerten und mit Hilfe dieser X,Y-Koordinaten feststellen, wo der Benutzer auf die Karte geklickt hat.

Es gibt mehrere Möglichkeiten, um diese Kartenobjekte anhand ihrer Positionsinformation zu finden:

- Kartenobjekte könnten so aufgestellt sein, dass man diese anhand einer Berechnungsfunktion finden kann
- Man berechnet das Ergebnis einer solchen Eingabe im voraus. Man kann dieses durch ein zweidimensionales Array erreichen, bei dem die Elemente einer Koordinate entsprechen und die ID des Kartenobjektes enthalten.
- Man betrachtet eine Tabelle in der die Koordinaten der Kartenobjekte vorhanden sind und überprüft, ob eines dieser Kartenobjekte an der eingegebenen Position ist

Da keine Berechnungsfunktion vorhanden ist und der Aufwand für die Erstellung vorberechneter Felder zu hoch ist, ist nur die Suche nach einem Kartenobjekt in einer Tabelle sinnvoll.

Man muss beim Einsatz einer Tabelle mit Kartenobjekten sichergehen, dass diese Berechnung nicht zu rechenaufwendig ist. Es gibt zwei Gründe, weshalb diese Berechnung zu langsam sein könnte. Der erste Grund ist eine evtl. sehr aufwendige Berechnung, ob ein Objekt an diesem Punkt ist. Der zweite Grund besteht in einer zu großen Anzahl der Kartenobjekte. Beide Gründe beeinflussen einander: Je aufwendiger die Berechnungsfunktion, desto weniger Kartenobjekte können durchsucht werden.

Da es sehr viele Kartenobjekte auf dem Gebäudeplan gibt, muss man evtl. sehr viele Berechnungen durchführen bevor man das Kartenobjekt findet. Deshalb muss die Berechnung einfach sein. Da die Kartenobjekte Räume sind lässt sich die Berechnung relativ einfach machen. Jeder Raum ist entweder rechteckig oder lässt sich durch Rechtecke darstellen. Dies führt zu folgender Berechnungsfunktion:

Das Kartenobjekt hat die X,Y Koordinaten des linken oberen und rechten unteren Punkt und eine ID der Karte in der Tabelle eingetragen.

Ein Punkt ist enthalten wenn gilt:

- (1) Kartenobjekt.KartenID = ID der momentanen Karte
- (2) Kartenobjekt.LinksX <= Punkt.X <= Kartenobjekt.Rechts.X
- (3) Kartenobjekt.ObenY <= Punkt.Y <= Kartenobjekt.UntenY

Man braucht fünf Vergleiche, um festzustellen ob der Punkt enthalten ist.

7.4 Die Navigation auf den Detailkarten

Bei der Navigation auf den Karten wird von einer Karte zu einer anderen Karte gewechselt, die in einer der vier Himmelsrichtungen oder ein Stockwerk über oder unter der Momentanen ist. Das Programm muss dabei wissen, welche Karte es anzeigen soll. Für diese Verknüpfung gibt es zwei Möglichkeiten:

- Man erstellt einen Graphen dessen Knoten die Detailkarten darstellen
- Man benutzt eine Berechnungsfunktion

Bei Einsatz eines Graphen müsste der Graph erstellt und in die Datenbank eingetragen werden, was zusätzlichen Speicherverbrauch bedeuten würde. Verwendet man eine Berechnungsfunktion spart man diesen Speicher. Diese kann wie z.B. für das Gebäude leicht erstellt werden und wird deshalb im Programm eingesetzt. Hierbei wurde den einzelnen Karten Zahlen vergeben, bei denen die erste Ziffer das Stockwerk, die zweite Ziffer die Reihe und die dritte Ziffer die Spalte angibt. Möchte man ein Stockwerk hoch oder runter, dann addiert man 100 bzw. -100 zur Zahl der momentanen Karte. Navigiert man in eine Himmelsrichtung so addiert man für Norden 10, für Süden -10, für West -1 und für Ost 1 zur Zahl der momentanen Karte. Anhand der einzelnen Ziffern kann man erkennen, ob es eine solche Karte gibt, da die einzelnen Ziffern nur in einem bestimmten Wertebereich sein können.

Stockwerk 1				Stockwerk2			
101	102	103	104	201	202	203	204
111	112	113	114	211	212	213	214
121	122	123	124	221	222	223	224

Abbildung 7-3: Der Aufbau der Karten

7.5 Zusammenfassung

In diesem Kapitel wurde der Einsatzzweck und die Unterschiede zwischen Karten und die Anforderungen an die Karte des Programms vorgestellt. Danach wurden drei alternative Kartenformen betrachtet und die dritte Alternative gewählt. Als Format für die Speicherung der Karten wurde ein Rasterformat gewählt. Danach wurde die Zuordnung von Kartenobjekten zu Sachdaten mit Hilfe von Fremdschlüsselbeziehungen beschrieben. Hierbei wurden die Verknüpfungsmöglichkeiten der Tabellen beschrieben, gezeigt, wie man die Position eines Kartenobjekts finden kann und wie Kartenobjekte anhand von Positionsinformation erhalten werden können. Als letztes wurde gezeigt, wie das Programm die Navigation auf den Detailkarten realisiert. Es benutzt hierfür keinen Graphen, sondern eine Berechnungsfunktion.

Wie man sehen konnte, beeinflusste die Größe des Palmdisplays die Auswahl der Karte. Ich verzichtete auf die Maßstabstreue der Karten, um das Palmdisplay optimaler ausnutzen zu können. Wenn der Palmbildschirm noch kleiner gewesen wäre, dann hätte ich sogar eine sehr vereinfachte Karte des Gebäudes erstellen müssen. Bei einer Entwicklung für Desktopsysteme hätte ich einfach die vorhandenen Karten benutzt.

Das folgende Kapitel verdeutlicht, dass die Realisierung scheinbar einfacher räumlicher Anfragen Probleme mit sich bringt. Es ist ein Beispiel dafür, wie solche Anfragen verändert werden, damit sie für den Anwender nützlich sind.

8 Realisierung der räumlichen Anfrage

Wie in Kapitel 4.5 "Funktionale Anforderungen" ersichtlich ist, ist eine wichtige Anforderung die Möglichkeit die räumliche Anfrage "Finde den nächsten Mitarbeiter" zu realisieren. In diesem Kapitel werden verschiedene Realisierungsmöglichkeiten diskutiert.

8.1 Semantik: nächster, nearest, next

Find nearest bedeutet wörtlich "finde den räumlich nächsten". Eine Umsetzung von find nearest nach dieser wörtlichen Bedeutung wird aus der Sicht des Anwenders nicht bei jedem Anwendungsfall optimal sein. Dies wird man bei der Beschreibung der Use Cases und bei der Beschreibung der Anforderungen sehen.

Im Deutschen hat das Wort "nächste" neben der Bedeutung des räumlich nächsten auch die Bedeutung des Nachfolgenden in einer Abfolge. Die räumliche Bedeutung entspricht dem englischen Wort "nearest", während die Bedeutung des Nachfolgenden in einer Abfolge im Englischen das Wort "next" ist.

Die Bedeutung des Nachfolgenden in einer Abfolge lässt sich oft als zeitlich nächster ansehen, da diese Reihenfolge oft dazu führt, dass man diese Abfolge dieser Reihe nach "abarbeitet". Bei einer Besuchsreihenfolge hat man beispielsweise diesen Fall. Diese Bedeutung muss aber nicht zeitlich sein, z.B. ist der nächste Buchstabe nach a b, was keinerlei zeitliche Bedeutung hat.

8.1.1 Use Cases

Durch diese Interpretationen ergeben sich folgende Use Cases:

Use Case 1: Finde den räumlich Nächsten (find nearest)

Der Anwender sucht den räumlich nächsten Mitarbeiter zu den eingegebenen Stichwörtern.

Use Case 2.1: Suchen eines weiteren Mitarbeiters (find next nearest)

Der Anwender hat sich nach Use Case 1 mit einem Mitarbeiter über ein Thema ausgiebigst unterhalten und will nun einen weiteren Mitarbeiter zu diesem Thema sprechen. Dieser sollte möglichst nah zum momentanen Aufenthaltsort sein.

Use Case 2.2: Suche nach dem nächsten nicht besuchten Mitarbeiter (find next)

Dieser Use Case ist eine Variante von Use Case 2.1.

Der Anwender sucht die Person, die am wenigsten entfernt zur Position des Anwenders ist, und noch nicht besucht wurde.

Use Case 2.3: Suche nach dem nächsten Mitarbeiter, den man nicht gerade besucht (find nearest but not actual)

Dieser Anwendungsfall ist eine Verallgemeinerung von Use Case 2.1.

Der Anwender sollte als Ergebnis von find nearest nicht den Mitarbeiter erhalten, den er gerade besucht hat (dieser ist am wenigsten entfernt), sondern den zweit wenigsten Entfernten.

Use Case 3: Die kürzeste Tour zum Besuch aller Mitarbeiter (shortest Tour)

Der Anwender möchte mehrere Mitarbeiter besuchen, die zu seiner Anfrage passen. Er will aber nicht unnötig lange unterwegs sein und will deshalb die kürzeste Tour zum Besuch aller Mitarbeiter machen.

Es gibt zwei mögliche Alternativen, wie der Anwender seine Ausgabe haben möchte.

- Als Ausgabe auf die find nearest Anfrage erhält er den als nächstes auf dem Weg zu besuchenden Mitarbeiter
- Es werden alle Mitarbeiter angezeigt und zwar so, dass der Anwender erkennen kann, was der kürzeste Weg ist

Der Anwender läuft dann diesen kürzesten Weg entlang.

8.1.2 Schlussfolgerungen

Wie man sieht hat die find nearest Funktion in den unterschiedlichen Use Cases unterschiedliche Semantiken. Diese Semantikunterschiede entstehen durch die Ergebnisse, die find nearest liefern soll.

Bei Use Case 1 ist für den Anwender nur der einzelne Mitarbeiter wichtig, der am räumlich nächsten ist. Die anderen Mitarbeiter sind bedeutungslos.

In den Use Cases 2.x ist nicht nur ein einzelner Mitarbeiter wichtig. Es wird nun eine ganze Gruppe von Mitarbeitern vom Anwender besucht. Diese Gruppe erhält eine Reihenfolge (Entfernung). Der Anwender gibt dem einzelnen Mitarbeiter aber immer noch eine größere Gewichtung als der ganzen Gruppe. Es ist nicht wichtig, wie man die Gruppe optimal besucht, sondern nur welcher Mitarbeiter als nächstes und damit am schnellsten besucht werden sollte. Das Ergebnis sieht den Mitarbeiter als Teil einer Gruppe.

In Use Case 3 tritt die Bedeutung des einzelnen Mitarbeiters in den Hintergrund, während die Bedeutung der Gruppe in der Vordergrund rückt. Das Ziel des Anwenders ist es auf einem möglichst kurzen Weg alle Gruppenmitglieder zu besuchen und nicht auf einem möglichst kurzen Weg den nächsten Mitarbeiter zu erreichen.

Das Ergebnis ist eine Gruppe deren Einzelteile (die Mitarbeiter) zu sehen sind.

8.2 Ausgabe der find nearest Funktion

An die find nearest Funktion können unterschiedliche Anforderungen gestellt werden. Hierbei ist wichtig, wie man die Anfrage find nearest einsetzt. Betrachtet man als Ergebnis immer nur einen Mitarbeiter, so wird man an solch eine Funktion andere Anforderungen stellen als an eine Funktion, die als Ergebnis eine Gruppe von Mitarbeitern hat. Je größer dabei die Bedeutung der Gruppe gegen-

über dem einzelnen Mitarbeiter ist, desto stärker kann sich die Ausgabe von einem echten find nearest unterscheiden.

Ergebnis: ein Mitarbeiter

Hier ist wichtig, dass die Funktion find nearest tatsächlich die räumlich nächste Person ausgibt.

Es muss keinerlei Rücksicht auf die anderen Mitarbeiter genommen werden, die ebenfalls zu den eingegebenen Stichwörtern passen.

Ergebnis: der Einzelne als Teil einer Gruppe

Neben der Funktion für die Berechnung der räumlich nächsten Person gibt es weitere wichtige Funktionen.

Es ist wichtig, wie die Navigation zwischen den einzelnen Gruppenmitgliedern geschehen soll. Sollen z.B. bereits besuchte Mitarbeiter wieder besucht werden oder nicht?

Es stellt sich bei der Realisierung auch die Frage, ob man die Gruppe nicht in irgendeiner Form anzeigen sollte.

Ergebnis: Gruppe

Hier steht die Betrachtung der Gruppe im Vordergrund. Es ist nun z.B. wichtiger einen optimalen Weg für den Besuch aller Gruppenmitglieder zu wählen, als den Anwender zum räumlich nächsten Mitarbeiter zu bewegen.

Die Ausgabe der Gruppe und damit die Ausgabe aller Mitarbeiter oder zumindest eines großen Teils ist ebenfalls wichtig.

8.3 Die Berechnung der Entfernung

Als Grundlage für die Implementierung einer find nearest Funktion braucht man eine Funktion zur Berechnung der Entfernungen zwischen der Position eines Anwenders und der Position eines Mitarbeiters.

Es werden zwei Varianten zur Berechnung der Entfernung vorgestellt. Die erste Variante ist die Berechnung der Luftlinie und die zweite die Berechnung des kürzesten Weges.

8.3.1 Berechnung der Luftlinie

Die Berechnung der Luftlinie zwischen zwei Punkten $P1(X1,Y1)$ und $P2(X2,Y2)$ ist im Prinzip relativ einfach:

$$\text{Entfernung} = \sqrt{(X2 - X1)^2 + (Y2 - Y1)^2}$$

8.3.2 Berechnung des kürzesten Weges

Für diese Berechnung benötigt man einen Graphen, dessen Knoten die Räume darstellen und dessen Kanten die Entfernung zwischen den beiden Knoten enthalten. Nachdem der kürzeste Weg durch eine Berechnung gefunden wurde, berechnet sich die Entfernung zwischen zwei Punkten wie folgt:

$$\text{Entfernung} = \sum_{\forall \text{Kanten} \in \text{kürzesterWeg}} \text{Kantenlänge}$$

Für die Berechnung des kürzesten Weges gibt es mehrere Algorithmen. Diese unterscheiden sich darin, ob und welche Heuristiken sie verwenden. Diese Algorithmen müssen diesen kürzesten Weg zur Laufzeit berechnen. Algorithmen, die eine Vorberechnung dieser kürzesten Wege machen würden, um zur Laufzeit auf diese Daten zuzugreifen, haben folgendes Problem: Da der Anwender als Startknoten in den Graphen eingetragen werden muss, verändert sich der Graph zur Laufzeit und damit die kürzesten Wege vom Anwender zu den Mitarbeiterräumen. Algorithmen die z.B. den minimalen Spannbaum benutzen, um die kürzesten Wege zu finden, können deshalb diesen Spannbaum nicht vorher berechnen.

Das Eintragen des Knotens für den Anwender in den Graphen ist auch nicht so leicht, da man im Graphen erst die richtige Stelle suchen muss, an der der Knoten eingesetzt werden muss.

8.3.3 Bewertung der Verfahren

Die Berechnung der Luftlinie ist sehr viel schneller möglich als die Berechnung des kürzesten Weges. Sie verbraucht auch weniger Speicher als die Berechnung des kürzesten Weges. Eine Implementierung ist viel leichter und benötigt im Gegensatz zur Berechnung des kürzesten Weges auch weniger Aufwand bei der Erstellung der Datenbasis. Aus der Sicht eines Anwenders ist die Berechnung des kürzesten Weges besser als die Berechnung der Luftlinie. Dieses liegt am Aufbau des Gebäudes, das in den oberen Stockwerken nur wenige Durchgänge zur anderen Gebäudehälfte hat. Hierdurch kann es sein, dass ein Mitarbeiter aufgrund der Entfernungsmessung mit Luftlinie näher ist als ein anderer Mitarbeiter, jedoch der Weg zu diesem länger ist.

Der Prototyp des Programms enthält nur die Berechnung der Entfernung mit Hilfe der Luftlinie, da dieses schneller und leichter ist.

8.4 find nearest für die Ausgabe von einzelnen Mitarbeitern

In diesem Kapitel werden unterschiedliche Find nearest Versionen betrachtet. Hierbei wird das echte find nearest vorgestellt und eine Implementierungsmöglichkeit vorgestellt. Dann werden die Probleme dieser find nearest Version aufgezeigt und eine mögliche Erweiterung vorgestellt, mit der man Probleme des echten find nearest beheben kann. Allerdings hat auch diese Erweiterungen Probleme.

8.4.1 Allgemeiner Ablauf von find nearest

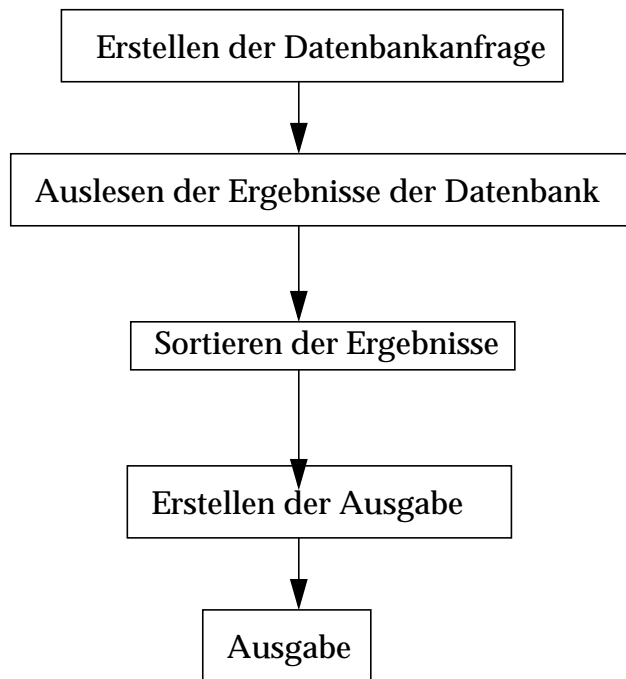


Abbildung 8-1: Ablauf von find nearest

8.4.2 Echtes find nearest

Dies ist eine Implementierung der wörtlichen Bedeutung. Hierbei wird der räumlich nächste Mitarbeiter angezeigt. Ist ein Mitarbeiter 5 Meter entfernt, ein anderer nur 3 Meter, so wird der 3 Meter entfernte angezeigt.

Technische Realisierung

Die Ergebnisse der Datenbankanfrage werden so durchsucht, dass man den am wenigsten entfernten Mitarbeiter erhält.

Ein Möglichkeit hierfür besteht in einer vollständigen Sortierung der Mitarbeiter nach Entfernung. Eine andere Möglichkeit besteht darin die Liste durchzugehen und sich den Mitarbeiter zu merken, der bisher am wenigsten Entfernt ist. Die zweite Möglichkeit ist schneller als die erste, da man die Mitarbeiter nur einmal durchgehen muss, um das Ergebnis zu haben. Die erste Möglichkeit eignet sich allerdings besser für die Realisierung von find nearest Implementierungen, die mehrere Mitarbeiter ausgeben.

Technische Probleme

Ein Problem könnte durch die Ermittlung der Entfernung zwischen Mitarbeiter und Anwender entstehen, da sie unter Umständen zu ineffizient sein kann. Wenn viele Treffer gefunden werden und der Palm diese im Speicher sortieren soll, kann es zu Speicherproblemen kommen.

Probleme beim Einsatz

Diese Funktion erfüllt Use Case 1. Allerdings werden die anderen Use Cases nicht erfüllt, da immer nur der räumlich nächste Mitarbeiter angezeigt wird. Hat man diesen besucht und sucht den nächsten Mitarbeiter zu dem gleichen Thema, so erhält man als Ergebnis wieder den gerade besuchten Mitarbeiter. Man erhält deshalb auch keinen Hinweis, in welche Richtung man gehen sollte.

Bewertung

Diese Funktion ist nur sinnvoll, wenn man explizit nur den am wenigsten entfernten Mitarbeiter sucht z.B. wenn man einen Mitarbeiter sucht, der den anderen etwas ausrichten soll.

Diese Funktion ist unbrauchbar, wenn man nach dem Besuch des räumlich nächsten Mitarbeiters einen anderen Mitarbeiter zum gleichen Thema besuchen möchte. Wenn sich am Tag der offenen Tür ein Besucher mit einem Mitarbeiter eines Projektes unterhalten hat, will der Besucher möglicherweise noch andere Projektmitglieder befragen. Das echte find nearest hilft diesem Besucher dabei nicht, da es den Mitarbeiter anzeigt, den der Besucher gerade besucht hat.

8.4.3 Find nearest mit Speicherung aller besuchten Mitarbeiter

Bei dieser find nearest Variante werden nur Mitarbeiter ausgegeben, die noch nicht besucht wurden.

Probleme beim Einsatz

Diese find nearest Version erfüllt Use Case 1, 2.1, 2.2. Die Erfüllung von Use Case 2.3 ließe sich erreichen, indem der Anwender eingibt, dass das Programm nun alle bis auf den gerade eben Besuchten anzeigen soll. Use Case 3 ist nicht realisierbar. Hierbei gibt es zwei Gründe, die dieses verhindern.

Der erste Grund liegt darin, dass man nicht den optimalen Weg erhält, wenn man sich vom Programm immer den kürzesten Weg angeben lässt. Ein Beispiel hierfür sieht man in Abbildung 8-2.

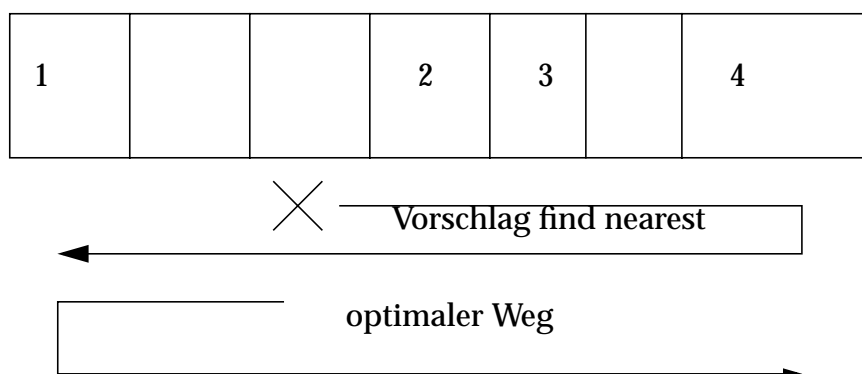


Abbildung 8-2: Beispiel für eine nicht optimale Wegwahl

Der optimale Weg wäre 1, 2, 3, 4 da aber der Benutzer näher zur zwei als zur 1 steht geht er den Weg 2, 3, 4, 1 entlang.

Der zweite Grund ist die Tatsache, dass immer nur ein Mitarbeiter angezeigt wird. Würden mehrer Mitarbeiter angezeigt, so könnte der Anwender sich notfalls den optimalen Weg herausuchen.

Die Funktion hat ein weiteres Problem. Es verstösst gegen das Prinzip der geringsten Verwunderung. Bei gleichen Anfragen kommt es zu unterschiedlichen Ergebnissen, was einen Anwender irritieren kann, wenn dieser das Verhalten der Funktion nicht kennt.

8.4.4 Probleme der find nearest Varianten für einzelne Mitarbeiter

Die find nearest Varianten haben eines gemeinsam: Sie gehen davon aus, dass es immer möglich ist eine Totalordnung über die Mitarbeiter aufzubauen. Es dürfen dafür aber keine gleichweiten Mitarbeiter existieren. Dieses ist beim Einsatz in der Realität aber nicht der Fall.

In vielen Räumen befinden sich mehrere Mitarbeiter. Diese Mitarbeiter besitzen oft auch ähnliche Stichwörter. Eine Auswahl welcher der Mitarbeiter angezeigt werden soll, kann eigentlich nur willkürlich sein. Weiterhin kann man auch nicht sicher sein, dass man nicht mehrere Räume hat, die gleichweit entfernt sind. Hierdurch wird das Problem verschärft, da man aus noch mehr Mitarbeitern auswählen müsste. Dieses Problem dürfte bei einer Anfrage in der Praxis sehr häufig vorkommen, da es viele Zimmer gibt, die einander gegenüber liegen und so gleichweit von einer Anwenderposition entfernt sind.

Als Lösung kann man sich nun Kriterien ausdenken, anhand der man nun einen Mitarbeiter aus diesen auswählt. Eine alphabetische Reihenfolge wäre denkbar oder die größte Relevanz zur Suchanfrage. Es dürfte allerdings schwierig sein mit Hilfe dieser Kriterien immer den Mitarbeiter auszugeben, den der Anwender ausgewählt hätte. Weiterhin erhöht eine Steigerung der Kriterien auch den Rechen- und Speicheraufwand. Einfacher ist es, mehrere Mitarbeiter auszugeben.

8.5 find nearest für mehrere Mitarbeiter

Durch die Ausgabe mehrer Mitarbeiter ergibt sich das Problem, dass man diese Mitarbeiter in einer sinnvollen Weise auf dem Bildschirm darstellen muss. Hierbei ist vor allem die Art der Sortierung wichtig. Es gibt hiervon zwei Gruppen. Die erste ist unabhängig von der Position der Mitarbeiter. Die zweite betrachtet die Position des Mitarbeiters und die Entfernung zum Anwender. Diese beiden Gruppen lassen sich auch kombinieren.

8.5.1 Anforderung an die Sortierung

Die Reihenfolge, in der die Mitarbeiter aufgelistet werden, sollte möglichst günstig sein. Eine günstigste Reihenfolge ergibt sich aus (vgl. [Herceg 1994]):

- der Reihenfolge, wie sie bei der Arbeit benötigt wird

- bestehenden Reihenfolgekonventionen (z.B. bei Personenadressen)
- der Relevanz der Information
- der Häufigkeit der Nachfrage nach der Information
- einer typischen Sortierreihenfolge (z.B. alphabetisch, chronologisch ...)

8.5.2 Sortiermöglichkeiten unabhängig von Positionsinformationen

Diese Sortiermöglichkeiten sind aus zwei Gründen interessant: Der erste Grund besteht darin, dass eine Sortierung mittels Positionsinformation keine Totalordnung erzeugt. Man wird bei der `find nearest` Funktion oftmals mehrere Mitarbeiter haben, die gleich weit entfernt sind. Eine Reihenfolge unter diesen Mitarbeitern sollte aber ebenfalls vorhanden sein und lässt sich mittels Sortiermöglichkeiten erreichen, die unabhängig von Positionsinformationen sind. Der zweite Grund ist, dass man so die Ergebnisse zwischen herkömmlichen Anfragen und ortsabhängigen Anfragen vergleichen kann. Die Sortiermöglichkeiten sind:

- Keine Sortierung ist leicht zu realisieren, aber bietet dem Anwender auch keine Hilfe.
- Häufigkeit der Nachfrage der Information: Es wird hierbei angenommen, dass oft benutzte Informationen wichtiger sind als selten benutzte. Dieses ist für die Sortierung der Mitarbeiter im Programm eher ungeeignet, da der Anwender häufig benutzte Informationen zu den Mitarbeitern, wahrscheinlich bereits kennt.
- Alphabetische Sortierung: Diese Sortierung hat den Vorteil, dass man Elemente deren Namen man kennt schnell finden kann. Ein weiterer Vorteil ist, dass man hierdurch oft eine echte Totalordnung hat, da der Fall, dass zwei Elemente identisch sind, nur sehr selten ist. Ein großer Nachteil ist, dass eine alphabetische Sortierung über Mitarbeiternamen keine Sortierung über etwas bedeutendes ist.
- Sortierung nach Relevanz: Hierbei wird das Ergebnis der Anfrage so sortiert, dass an oberster Position die zur Anfrage relevanteste Person steht. Die Implementierung dieser Sortierfunktion hat ein Problem: Was ist am relevantesten für einen Anwender? Da dieses nirgends definiert ist und Anwender von Anfrage zu Anfrage unterschiedliche Präferenzen haben, ist es nicht leicht, diese Sortierung zu implementieren. Ein wichtiges Relevanzkriterium wäre bei `find nearest` die Entfernung.

8.5.3 Sortiermöglichkeiten abhängig von Positionsinformationen

Da `find nearest` den oder die räumlich nächsten Mitarbeiter anzeigen soll, ist eine Sortierung nach Entfernung sicherlich sehr sinnvoll. Es gibt allerdings auch hier unterschiedliche Möglichkeiten, wie man die Mitarbeiter sortieren kann. Hierbei kann man unterschiedliche Entfernungsberechnungen machen und die Mitarbeiter gruppieren und diese dann als Gruppe ausgeben, indem man die Mitarbeiter nach einzelnen Gebieten sortiert oder sie so auflistet, dass sie in dieser Reihenfolge den optimalen Weg darstellen. Die positionsabhängigen Sortierungen sind:

- **Entfernung:** Hierbei werden die Mitarbeiter nach ihrer Entfernung sortiert. Der räumlich Nächste steht an erster Stelle und der Entfernteste an letzter. Das Ergebnis entspricht den Erwartungen an eine find nearest Funktion. Es gibt allerdings oft Mitarbeiter die gleichweit vom Benutzer entfernt sind. Bei einer Position im Gang in der Mitte des Gebäudes, kann man bis zu 8 Räume auf einer Ebene haben, die gleichweit entfernt sind. Ein Problem ist, dass man anhand dieser Sortierung nur schlecht erkennen kann, wieviele Mitarbeiter in einem Bereich zu einer Anfrage vorhanden sind. Dieses Problem tritt auf, wenn man eine Anfrage macht, die als Ergebnis viele Mitarbeiter aus unterschiedlichen Kartenbereichen hat. Wenn die Mitarbeiter der Kartenbereiche ähnlich weit von Mitarbeitern aus anderen Kartenbereichen entfernt sind, so wird man bei der Anzeige der Mitarbeiterposition, ständig zwischen den einzelnen Kartenteilen hin- und herspringen.
- **Zonen:** Zonen sind Gebiete, in denen alle Positionen als "gleich weit" gelten. Zonen gibt es beispielsweise beim öffentlichen Nahverkehr (zumindest bei VVS und SSB). Hier werden die Gebiete in einzelne Zonen eingeteilt. Entspricht eine Zone einem Kartenausschnitt, so ist das Hin- und Herspringen beim Betrachten der Mitarbeiterpositionen nicht mehr vorhanden. Leider ist die Sortierung durch Zonen nur sehr grob, weshalb man sie nicht alleine benutzen sollte.
- **optimaler Weg:** hierbei werden die Mitarbeiter so angeordnet, dass man den optimalen Weg erhält. Der als nächstes zu besuchende ist dabei an erster Position. Ein Problem ist der große Rechenaufwand. Ein weiteres Problem besteht darin, dass bei einer Anfrage auch Mitarbeiter angezeigt werden, die der Anwender nicht besuchen möchte. Diese beeinflussen das Ergebnis wodurch der Anwender dann einen nicht so optimalen Weg entlanggeht. Da ein optimaler Weg dargestellt wird, gibt es kein Hin- und Herspringen zwischen den Kartenteilen und die Sortierung der Mitarbeiter ist relativ fein, da nur noch Mitarbeiter eines Raumes nicht sortiert werden können. Ihre Reihenfolge kann allerdings beliebig gewählt werden.

8.5.4 Kombinationsmöglichkeiten

Alle Sortiermöglichkeiten haben Schwachpunkte. Die Sortiermöglichkeiten, die unabhängig von Positionsinformationen sind, können nicht alleine eingesetzt werden, da sie nicht zu find nearest passen, da dieses eine positionsabhängige Sortierung suggeriert. Die positionsabhängigen Sortierungen sind entweder nicht fein genug (Entfernung, Zonen), sind zu aufwendig (optimaler Weg) oder verursachen Probleme bei der Nutzung (optimaler Weg: Weg über ungewollte Mitarbeiter; Entfernung: Hin- und Herspringen auf der Karte). Eine Kombination dieser Sortiermöglichkeiten kann hierbei zu Verbesserungen führen, was durch Kombination der Sortiermöglichkeiten Entfernung, Zone und alphabetische Reihenfolge demonstriert wird.

Die wichtigsten Kriterien sind Entfernung und Zone, da diese eine positionsabhängige Sortierung erzeugen. Das Kriterium der alphabetischen Reihenfolge sollte als letztes angewandt werden, da dieses am wenigsten zum Nutzen aus Sicht des Anwenders beiträgt und bei Anwendung im Normalfall eine Totalordnung erzeugt. Es sortiert dann die Mitarbeiter, die wie Mitarbeiter im gleichen Raum,

nicht anhand der Entfernung oder der Zugehörigkeit einer Zone sortiert werden können.

Die Reihenfolge in der mit Entfernung und Zone sortiert wird, führt zu zwei unterschiedlichen Ergebnissen.

- Reihenfolge: Entfernung, Zone: Es werden alle Mitarbeiter sortiert bis auf die gleichweit Entfernten. Die Mitarbeiter in der gleichen Zone z.B. Mitarbeiter im gleichen Raum oder in gegenüberliegenden Räumen werden, da sie in der gleichen Zone sind, hintereinander geschrieben. Es kann aber so keine Totalordnung entstehen, da erstens die Zonen untereinander sortiert sein müssen und zweitens Mitarbeiter in der gleichen Zone untereinander keine Sortierung haben. Mitarbeiter in gleichen Räumen oder gegenüberliegend müssen immer noch sortiert werden. Man hat bei dieser Sortierreihenfolge immer noch das Problem mit dem Hin- und Herspringen auf der Karte. Das Problem ist allerdings kleiner als bei einer Sortierung nach Entfernung, da die gleichweit entfernten Mitarbeiter nach Zonen sortiert werden und so die Mitarbeiter in der dargestellten Zone zuerst angezeigt werden. Dies ist eine Verbesserung gegenüber der Sortierung nach Entfernung.
- Reihenfolge: Zone, Entfernung: Die Mitarbeiter werden in Zonen aufgeteilt, wobei es über den Zonen eine Totalordnung gibt. Die Mitarbeiter in den jeweiligen Zonen werden, dann nach Entfernung zum Anwender sortiert. Diese Reihenfolge behebt das Herumspringen und hat eine Sortierung, die man bei einer find nearest Implementierung als sinnvoll ansehen kann.

8.5.5 Zusammenfassung

Das "echte" find nearest ist nur brauchbar, wenn man nur einen einzigen Mitarbeiter sucht. Bei mehreren Mitarbeitern zur gleichen Suchanfrage ist echtes find nearest für den Anwender als Suchhilfe unbrauchbar. Die find nearest Version, in der alle besuchten Mitarbeiter gespeichert werden, ist ein Fortschritt. Sie verstößt allerdings gegen das Prinzip der geringsten Verwunderung. Da die Mitarbeiter im gleichen Raum oft gleiche Stichwörter besitzen, gibt es die Notwendigkeit einen auszuwählen. Durch Angabe mehrerer Mitarbeiter kann man dieses Problem umgehen. Man muss dann eine sinnvolle Sortierreihenfolge festlegen.

Es gibt zwei grundlegende Sortiermöglichkeiten: positionsunabhängig und positionsabhängig. Beide Arten alleine sind für den Einsatz bei einer find nearest Funktion nicht ideal. Mit Hilfe von Kombinationen der Sortiermöglichkeiten kann man eine Verbesserung des Ergebnisses erzielen. Hierbei sind die Sortierungen nach Entfernung, Zone und die alphabetische Sortierung betrachtet worden. Die alphabetische Sortierung kann nur als letztes angewendet werden, da sie zu stark vorsortiert. Sie ermöglicht es eine Totalordnung über die Mitarbeiter zu erhalten.

Dieses Kapitel zeigt, wie die räumliche Anfrage an die Bedürfnisse des Anwenders angepasst werden musste. Das Problem liegt dabei darin, dass der Anwender eine Funktion braucht, die beide Bedeutungen des Wortes "nächster". Die räumliche Bedeutung steht dabei sehr stark im Vordergrund verliert allerdings

sobald mehrere Mitarbeiter besucht werden sollen an Bedeutung. Je nach dem wie stark der Anwender die Gruppe oder den einzelnen Mitarbeiter gewichtet, tritt die zweite Bedeutung des Wortes nächste in den Vordergrund.

Das nächste Kapitel zeigt die Tabellen der Datenbank und beschreibt die Realisierung der Anfragen an diese.

9 Tabellen und Anfragen

Dieses Kapitel zeigt die Tabellen und die Anfragen, die in DB2 Everyplace verwendet werden.

9.1 Tabellen der Datenbank

Die Tabellen lassen sich in drei Kategorien anordnen: Sachdatentabellen, Tabellen für Positionsinformationen auf den Karten und Fremdschlüsseltabellen. Im folgenden werden die Tabellen kurz vorgestellt und ihr Schema gezeigt. Hierbei sind unterstrichene Attribute Primärschlüssel und doppelt unterstrichene Fremdschlüssel.

9.1.1 Tabellen für Positionsinformationen

Hierbei gibt es zwei Arten von Positionsinformation: Positionsinformation über Räume auf den Detailkarten und der Position von Detailkarten auf der Übersichtskarte.

Beide Tabellen sind ähnlich aufgebaut. Sie enthalten beide Positionsinformation und einen Primärschlüssel auf das Objekt an dieser Position. Die Positionsinformationen sind die linke obere und rechte untere Ecke eines Rechtecks.

Die Tabelle für die Positionsinformation der Räume auf den Karten benötigt als Positionsinformation auch die Karten-ID, der dazugehörigen Karte. Die Karten-ID ist der Primärschlüssel der Karte. Es wird eine Raum-ID für den Raum an dieser Stelle der Karte gespeichert. Dies ergibt folgende Tabelle für Raumpositionen:

Roompositions = {x1 Integer, y1 Integer, x2 Integer, y2 Integer, mapid Integer, roomid Integer}

Die Tabelle für die Kartenpositionen der Detailkarten auf der Übersichtskarte ist etwas anders aufgebaut. Da es nur eine Übersichtskarte gibt, ist keine Karten-ID für die Bezeichnung der Übersichtskarte nötig. Es ist aber eine Karten-ID für die Detailkarte an der Position nötig. Diese Karten-ID ist der Primärschlüssel der Karte. Die Tabelle sieht folgendermaßen aus:

Mappointions = {mapid Integer, x1 Integer, y1 Integer, x2 Integer, y2 Integer}

9.1.2 Sachdatentabellen

Es gibt Tabellen für Mitarbeiterdaten (Persondata), Raumdaten (roomdata), Stichworte (keywords), und Abteilungsdaten (departmentdata). Die Tabellen haben folgende Schemata:

Persondata = {personid Integer, name Varchar(30), departmentid Integer, phone Varchar(30), url Varchar}

Roomdata = {roomid Integer, roomno Varchar(10), description Varchar(30), phone Varchar(30), url Varchar(30)}

Departmentdata = {departmentid Integer , name Varchar(40)}

Keywords = {keywordid Integer, keyword Varchar(30)}

Fremdschlüsseltabellen

Die Tabelle Persontokeywords weist den Mitarbeitern Stichworte zu:

Persontokeywords = {personid Integer, keywordid Integer}

Die Tabelle Persontoroom weist den Mitarbeitern ihre Räume zu:

Persontoroom = {personid Integer, roomid Integer}

9.2 Die Anfragen an die Datenbank

9.2.1 Detailkarte auf Position(x,y) der Übersichtskarte

Es wird zu einer Position auf der Übersichtskarte, die zu diesem Gebiet dazugehörige Detailkarte gesucht.

```
SELECT MapID
FROM MapPositions
WHERE X1 <= ? AND X2 >= ? AND Y1 <= ?' and Y2 >= ?'
? = X Position, ?' = Y Position
```

9.2.2 Erhalt aller Stichwörter

Diese Funktion liefert alle Stichwörter, die die Datenbank gespeichert hat..

```
SELECT keyword
FROM Keywords
ORDER BY keyword
```

9.2.3 Erhalt der Namen aller Mitarbeiter

Man erhält hierdurch die Namen aller gespeicherten Mitarbeiter.

```
SELECT DISTINCT name
FROM PersonData
ORDER BY name
```

9.2.4 Mitarbeiterposition auf Detailkarte

Es wird die Position eines Mitarbeiters zurückgegeben. Da es keine explizite Position gibt, wird das Gebiet des Raumes zurückgegeben, in dem sich der Mitarbeiter befindet.

```
SELECT X1 ,Y1 ,X2 ,Y2 ,MapID
FROM Roompositions r, Persontoroom t
WHERE t.PersonID = ? AND t.RoomID = r.RoomID
? = MitarbeiterID
```

Diese Anfrage kann zu mehreren Ergebnissen führen, da Räume auf mehreren Karten vorkommen können oder Räume aus mehreren solchen Quadraten bestehen.

Das Programm wird das erste Rechteck aus der Ergebnistabelle zurückgeben. Die Umwandlung dieses Gebietes in einen Punkt wird nicht in der Datenbank, sondern im Modul Mapsver gelöst, der diese Anfrage gestellt hat. Dieses berechnet im Programm einfach den Mittelpunkt dieses Gebietes.

9.2.5 Daten zu Mitarbeiter

Die Daten zu einem Mitarbeiter werden zurückgegeben.

Es sind zwei Anfragen notwendig.. Mit der ersten Anfrage erhält man die allgemeinen Daten (name, phone, id, department, roomno). Durch die zweite Anfrage erhält man die Stichworte des Mitarbeiters.

Allgemeine Daten

```
SELECT p.name,p.phone,d.name,roomNo
FROM PersonData p,DepartmentData d,PersonToRoom t, RoomData r
WHERE p.personID = ? AND p.personid = t.personid
AND t.roomid = r.roomid AND d.departmentid = p.departmentid
? = ID des Mitarbeiters
```

Stichwörter

```
SELECT keyword FROM Keywords k, Persontokeywords t
WHERE t.personid = ? AND k.keywordid = t.keywordid
? = Person ID
```

9.2.6 Raum an Position einer Detailkarte

Es sollen die Raumdaten des Raumes an Position x, y auf einer Detailkarte gefunden werden.

Es werden zwei Select Anweisungen benötigt. Die erste für die Raumdaten , die zweite für die Mitarbeiter im Raum.

Raumdaten

```
SELECT roomid, roomno, description, phone, url
FROM roomposition p, roomdata r
WHERE (p.x1<= ? AND p.x2 >= ?)
AND (p.y1 <= ?' AND p.y2 >= ?') AND mapid = ?''
AND p.roomid = r.roomid
```

Bei der Ausführung wird ? durch die X-Position ersetzt, ?' durch die Y-Position und ?'' durch die mapID.

Mitarbeiter

```
SELECT personid, name
FROM persondata
```

```
WHERE roomid = ?
```

? wird durch die RoomID ersetzt, die man durch die vorherige Anfrage erhalten hat.

9.2.7 Position der Detailkarte auf der Übersichtskarte

Die Anweisung hierfür sieht folgendermaßen aus:

```
SELECT x1,y1,x2,y2
FROM Mappositions WHERE mapid = ?
```

? ist die ID der gesuchten Detailkarte

9.2.8 Mitarbeiter zu Stichworten

Man erhält die Mitarbeiter, die gesuchte Stichwörter haben. OR Stichwörter sind Stichwörter die durch ein Oder miteinander verknüpft sind. Ein Mitarbeiter wird in das Ergebnis aufgenommen, wenn er mindestens eines dieser Stichwörter hat. AND Stichwörter sind Stichwörter, die ein Mitarbeiter haben muss, wenn er in das Ergebnis der Anfrage aufgenommen werden soll. Enthält eine Anfrage AND und OR Stichwörter so muss nur nach AND Stichwörtern gesucht werden.

Stichwortsuche mit OR Stichworten

Da ich für die Stichwortliste im SQL Befehl nicht einfach so ein oder mehrere ? stehen lassen kann, wird der SQL-Befehl zur Laufzeit zusammengebaut. Die Anweisung sieht dann folgendermaßen aus:

```
SELECT DISTINCT p.personid, p.name
FROM Persondata p, Keywords k, Persontokeywords t
WHERE p.personid = t.personid AND t.keywordid = k.keywordid
AND keyword = "keyword1" OR keyword = "keyword2"
OR keyword = ...
```

Stichwortsuche mit AND Stichworten

Bei mehreren AND Stichwörtern muss man anders Vorgehen als bei OR-Stichwörtern. Der Befehl

```
SELECT personID, name
FROM PersonData p, Keywords k Persontokeywords t
WHERE p.personid=t.personid AND t.keywordid = k.keywordid
AND keyword = "keyword1" AND keyword = "keyword2"
AND keyword = "keyword3" ...
```

führt zu einem leeren Endergebnis, da ein Stichwort nur dann allen Stichworten entspricht, wenn nur nach einem Stichwort gesucht wird oder diese Stichworte alle gleich sind.

Die Anfrage muss auch die restlichen Stichwörter in der Tabelle betrachten. Eine einfache Lösung wäre folgende Anweisung, in der man für jedes Stichwort ein Join mit der Stichworttabelle macht:

```

SELECT personID, name
FROM PersonData p, Persontokeywords t, Keywords k1,
Keywords k2...
WHERE p.personid = t.personid AND t.keywordid = k1.keywordid
AND t.keywordid = k2.keywordid AND ...
AND k1.keyword = "keyword1" AND k2.keyword = "keyword2"...

```

Ein Problem dieser Anfrage besteht in den vielen Joins, die sie benutzt. Wenn nach vielen Stichworten gesucht wird, dann muss die Datenbank Joins sehr effizient beherrschen oder die Anfrage so optimieren, dass sie keine Joins einsetzen muss.

Ein Alternative zur obigen Anfrage ergibt sich, wenn man die Operation INTERSECT benutzt, mit der man die Schnittmenge zweier Tabellen erhält. Die Anweisung hierfür würde folgendermaßen aussehen:

```

(SELECT personID,Name
From PersonData p, Persontokeywords t, Keywords k
WHERE p.personid = t.personid AND t.keywordid = k.keywordid
AND k.keyword = "keyword1")
INTERSECT
(SELECT pesonID,Name
FROM PersonData p, Persontokeywords t, Keywords k
WHERE p.personid = t.personid AND t.keywordid = k.keywordid
AND k.keyword = "keyword2") ...

```

DB2 Everyplace kennt kein Intersect. Deshalb muss man die Anweisung selbst programmieren. Für jedes Stichwort wird folgende Anweisung ausgeführt:

```

SELECT personid, name
FROM Persondata p, Keywords k, Persontokeywords t
WHERE p.personid = t.personid AND t.keywordid = k.keywordid
AND keyword = ?
ORDER BY personid
? = Stichwort

```

Danach werden die Personen übernommen, die in dem Ergebnis jeder Anfrage enthalten sind. Im Programm wird dafür folgender Algorithmus benutzt:

Erhöhe den Cursor, der auf das Element mit der niedrigsten personid zeigt, solange bis entweder alle Cursor auf die gleiche personid zeigen oder der zu erhöhende Cursor am Ende der Ergebnistabelle angekommen ist.

Zeigen alle Cursor auf die gleiche Personid, so wird sie in das Ergebnis übernommen. Danach werden alle Cursor eins erhöht.

Der Algorithmus wird beendet, sobald der zu erhöhende Cursor am Ende der Ergebnistabelle angekommen. Danach werden die Ergebnisse zurückgeliefert.

9.2.9 Mitarbeiter zu einer Namensliste

Es werden Daten der Mitarbeiter zurückgegeben, deren Namen in der Namensliste enthalten sind.

Aus der Namensliste wird folgender Suffix erzeugt: Alle Namen werden mit einem OR verbunden und der Name in der Liste wird mit dem Namen in der Tabelle verglichen. Der Suffix sieht dann so aus:

```
Name = Name1 OR Name = Name2 OR Name = Name3 ...
```

```
SELECT PersonID, Name FROM PersonData where + Suffix
```

9.3 Zusammenfassung

Es gibt drei Gruppen von Tabellen. Die Fremdschlüsseltabellen verknüpfen die Daten in der Sachdatentabellen mit den Tabellen für die Positionsinformationen. Die Anfragen werden durch ein oder zwei SQL-Anweisungen gemacht. Die Anfragen für die Stichwortsuche müssen zur Laufzeit erstellt werden.

Im folgenden Kapitel wird die Datenbank evaluiert. Hierbei wird man die Geschwindigkeit der Datenbank bei der Ausführung der Anfragen sehen können.

10 Evaluation der Anwendung

In diesem Kapitel werden die Datenbank und das Programm bewertet.

10.1 Bewertung der Datenbank

Zuerst wird die Geschwindigkeit der Datenbank betrachtet. Hierfür wird mit Hilfe eines Benchmarks die Zeit gemessen, mit der die Datenbank die Anfragen an das Programm ausführt. Als nächstes werden Optimierungen besprochen und die Geschwindigkeit mit Optimierungen bewertet. Danach wird betrachtet, ob sie die Probleme löst, die in Abschnitt 3.6.1 'Motivation für den Einsatz einer Datenbank' auf Seite 14 genannt werden.

10.1.1 Der Aufbau des Benchmarks

Die Erstellung eines Benchmarks ist eine schwierige Aufgabe, da die Auswahl der Tests und deren Bewertung nicht einfach ist. Da im Rahmen dieser Studienarbeit kein Benchmark zum Testen mobiler Datenbanken erstellt werden sollte, zeigt der folgende Benchmark deshalb in erster Linie, wie geeignet DB2 Everyplace für das Programm ist. Eine Verallgemeinerung der Bewertung auf den Nutzen der Datenbank für andere Anwendungen sollte nicht gemacht werden.

Da der Benchmark zumindest Aussagekraft für das Programm haben sollte, werden die Anfragen des Programms an die Datenbank benutzt. Als Daten werden die Daten zur Fakultät benutzt.

Der Benchmark kann mit mehreren Programmen durchgeführt werden. Das erste ist das entwickelte Programm, das zweite das Programm QBE und das dritte ist ein selbstgeschriebenes Benchmarkprogramm.

Ein Problem bei Benchmarks ist die Bewertung der Ergebnisse. Da der Benchmark in erster Linie über die Brauchbarkeit des Einsatzes im Programm Aussagekraft haben soll, müssen die Anfragen innerhalb der für das Programm tolerierbaren Zeit erledigt werden.

Als Rechner kam ein Handspring Visor Deluxe mit 8 MB RAM und Palm OS 3.3 zum Einsatz.

10.1.2 Betrachtung der möglichen Programme

Eigenschaften des entwickelten Programms

Das Programm benutzt die SQL-Anweisungen. Allerdings werden diese oft zusammen ausgeführt z.B. um die Personendaten und die Schlüsselwörter zu holen. Auch werden manche Befehle mehrmals hintereinander ausgeführt. Beispielsweise werden bei einem Find Nearest für jede Person in der Ergebnismenge die Daten zur Position geladen. Ein weiteres Problem besteht darin, dass die Ergebnisse mehrere Algorithmen durchlaufen bevor sie gesehen werden können. Probleme entstehen auch durch Beschränkungen des Programms. Es kann z.B. nur maximal zehn Personen einlesen. Die Zeit für die Ausführung der Befehle wird

nicht vom Programm gemessen. Personen, die das Programm nicht haben, können die Ergebnisse nur sehr schlecht nachvollziehen.

Die Ausgabe des Ergebnisses hat einen geringen Einfluss auf die Messung der Ausführungsdauer der Anfragen.

Eigenschaften von QBE

Das Programm QBE wird mit DB2 Everyplace mitgeliefert. Man kann mit diesem Programm die Tabellen der Datenbank betrachten und einfache Filterungen und Sortierungen machen. Eine weitere Komponente ist der CLP (Command Line Processor). Mit diesem ist es möglich SQL-Befehle einzugeben und sich die Ergebnisse anzeigen zu lassen.

Ein Problem von QBE ist die relativ langsame Ausgabe der Ergebnisse auf dem Bildschirm. Hierdurch wird die Ausführungszeit bei Ergebnissen mit vielen Elementen stark verfälscht. Ein weiteres Problem besteht darin, dass die Eingabe der SQL Befehle ohne Tastatur ziemlich aufwendig ist. Da die Ausführungszeit der Befehle nicht gemessen wird, muss man das beim Benchmark selbst machen.

Eigenschaften des selbstentwickelten Benchmarkprogramms

Das selbstentwickelte Benchmarkprogramm sollte die Ausführung des Tests vereinfachen, die Ausführungsdauer selbst messen, um auch Befehle bewerten zu können, die sehr schnell ausgeführt werden müssen und störende Einflüsse auf die Ausführungsdauer minimieren.

Das Benchmarkprogramm wurde wie in Abschnitt 6.6.1 'Benchmark der Datenbank' auf Seite 49 beschrieben erstellt. Es werden nur die Komponenten für die Datenbank benutzt und eine neue Oberfläche entwickelt. Die Datenbankschnittstelle ruft bei diesem Benchmark die Funktionen der DB2 Everyplace Datenbank auf und misst wie lange die Ausführung dauert. Daraufhin wird die Ausführungszeit angezeigt. Die Funktionen führen immer nur eine SQL Anweisung aus, haben keine Eingabewerte und auch keine Rückgabewerte. Die Parameter der SQL-Anweisungen werden nun in der Funktion definiert. Bei SQL-Anweisungen, die erst zur Laufzeit erstellt werden müssen, wie z.B. bei der Stichwortsuche nach OR-Stichwörtern, wird der SQL-Befehl bereits erstellt sein. Hat eine SQL-Anweisung mehrere Ergebnisse, so werden diese Ergebnisse von der Datenbank geladen, aber nicht verarbeitet d.h. es wird nur der Cursor auf der Ergebnistabelle ein Element weiterbewegt. Dadurch wird nur die Geschwindigkeit der Datenbank bewertet, da die Zeit für die Verarbeitung des Ergebnisses minimal ist.

Die Verbreitung dieses Benchmarkprogramms ist klein, aber die Erstellung eines vergleichbaren Programms ist leicht möglich, da es nur die SQL-Anweisungen ausführt, die Ergebnisse ausliest und die hierfür benötigte Zeit misst.

Bewertung der Programme

QBE und das Programm sind für den Einsatz als Benchmarkprogramm nur bedingt geeignet. Beide Programme führen keine Messung der Ausführungsdauer durch und eine Durchführung des Benchmarks ist aufwendig. Störeinflüsse

sind bei beiden vorhanden. Die Ausgabegeschwindigkeit von QBE ist relativ gering, wodurch die Ausführungszeit einer SQL-Anweisung bei Ergebnissen mit vielen Elementen verfälscht wird. Das Programm kann nur wenig Elemente einlesen und führt auch mehrere Algorithmen auf diesen aus bis diese angezeigt werden.

Es ist deshalb notwendig gewesen ein Benchmarkprogramm zu schreiben. Die Entwicklung ist nicht aufwendig, da einfach die Datenbank des Programms umgebaut werden kann. Die Erstellung des Benchmarkprogramms hat deshalb ca. drei Stunden gedauert.

10.1.3 Der Aufbau der Datenbank

Tabellen der Datenbank

Tabelle 10-1 zeigt die Tabellen in der Datenbank. Hierbei wird angegeben wieviele Zeilen und Spalten die Tabellen haben und ob sie einen Primärschlüssel besitzen. Die Tabellen haben alle relativ wenig Zeilen.

Tabelle 10-1: Tabellen der Datenbank

Tabelle	Zeilen	Spalten	Primärschlüssel
Departmentdata	25	2	Ja
Keywords	22	2	Ja
Mappositions	36	5	Nein
Persondata	180	5	Ja
Persontokeywords	186	2	Nein
Persontoroom	181	2	Nein
Roompositions	231	6	Nein
Roomdata	203	5	Ja

Überblick über die SQL-Anfragen

Die Tabelle 10-2 enthält alle Anfragen an die Datenbank und gibt die Anzahl der als Ergebnis erwarteten Elemente an, welche Tabellen betroffen sind, welche Werte die Parameter im Benchmark haben und welche Anforderungen an die Ausführungsdauer gestellt werden

Tabelle 10-2: Statistik zu den SQL Befehlen

Anfrage	Ergebnismenge	Betroffene Tabellen	Parameter	Ausführungsdauer (Soll) (Sekunden)
Anfrage 1: Detailkarte an Pos. der Übersichtskarte	1	Mappositions	x1,x2 = 40, y1,y2 = 50	<2
Anfrage 2: Alle Schlüsselwörter	22	Keywords		<=1
Anfrage 3: Alle Namen	180	Persondata		2-3
Anfrage 4: Position einer Person	1	Roompositions, Persontoroom	personid = 40	<1
Anfrage 5: Daten zu Person	1	Persondata, Departmentdata, Persontoroom, Roomdata	personid = 40	<2
Anfrage 6: Schlüsselwörter einer Person	1-2	Keywords, Persontokeywords	personid = 40	<2
Anfrage 7: Daten zu Raum	1	Roompositions, Roomdata	x1,x2 = 40, y1,y2 = 40, mapid = 204	<2
Anfrage 8: Personen in Raum	0-3	Persondata, Persontoroom	roomid = 47	<1
Anfrage 9: Detailkartenposition	1	Mappositions	mapid = 201	<2
Anfrage 10: Stichwortsuche (OR)	Je nach Anfrage, 17 bei "Anwendersoftware"	Persondata, Keywords, Persontokeywords	keyword = "Anwendersoftware"	<10
Anfrage 11: Stichwortsuche (AND)	Maximal 1	Persondata, Keywords, Persontokeywords	keyword1 = "Anwendersoftware" keyword2 = "VerteilteSysteme"	<10
Anfrage 12: Suche mit Namen	1 pro Namen	Persondata	name = "Nicklas,Daniela"	<10

10.1.4 Die Resultate des Benchmarks

Die Tabelle 10-3 zeigt das Ergebnis des Benchmarks. Hierbei wird zu jeder Anfrage die Anzahl der betroffenen Tabellen, die maximale Ausführungsdauer, die tatsächliche Dauer der Ausführung und eine Bewertung angegeben

Tabelle 10-3: Ausführungsdauer

Anfrage	Anzahl Tabellen	Ausführungsdauer (Soll) (Sekunden)	Ausführungsdauer (Ist) (Sekunden)	Bewertung
Anfrage 1:	1	<2	0,56	ok
Anfrage 2:	1	<=1	0,83	ok
Anfrage 3:	1	2-3	0,63	ok
Anfrage 4:	2	<1	36,28	zu langsam
Anfrage 5:	4	<2	0,91	ok
Anfrage 6:	2	<2	3,89	etwas langsam
Anfrage 7:	2	<2	0,37	ok
Anfrage 8:	2	<1	27,26	zu langsam
Anfrage 9:	1	<2	0,1	ok
Anfrage 10:	3	<10	38,17	zu langsam
Anfrage 11:	3	<10	109,49	zu langsam
Anfrage 12:	1	<10	0,41	ok

10.1.5 Bewertung des Ergebnisses

Die Geschwindigkeit vieler Anfragen ist zu langsam. Etwas zu lang ist die Ausführungsdauer für die Schlüsselwörter einer Person.

Besonders problematisch sind die Anfragen zur Position eines Benutzers, den Personen in Raum und zur Suche mit Schlüsselwörtern. Die Anfragen zur Position einer Person und zur Suche mit Schlüsselwörtern, werden zur Demonstration von Find Nearest benötigt. Bei der Ausführung dieser Find Nearest Funktion muss der Anwender beim Stichwort Anwendersoftware 38,17 (Anfrage 10)+ 17 * 36,28 (Anfrage 4) Sekunden = 654,93 Sekunden warten. In dieser Form ist die Datenbank ungeeignet für den Einsatz mit dem Programm und Optimierungen notwendig sind.

10.2 Optimierungsmöglichkeiten

Als erstes werden die Ergebnisse des Benchmarks betrachtet und die Anfragen ausgewählt, die optimiert werden sollen. Als nächstes werden Optimierungsmöglichkeiten vorgestellt und bewertet. Es werden dann sinnvolle ausgewählt und die Ausführungsdauer der Anfragen mit Optimierungen dargestellt.

10.2.1 Betrachtung des Ergebnisses

Eine Optimierung sollte bei vier Anfragen gemacht werden. Besonders schlecht ist dabei die Anfrage 4: "Position einer Person". Diese sorgt dafür, dass die find nearest Funktion praktisch unbrauchbar wird, da sie für jeden Mitarbeiter im Ergebnis einmal aufgerufen wird. Die Anfrage 8: "Personen in Raum" ist ebenfalls zu langsam. Die beiden Anfragen zur Schlüsselwortsuche (Anfrage 10: "Stichwortsuche (OR)", Anfrage 11: "Stichwortsuche (AND)") sind auch zu langsam. Da beide Anfragen sehr ähnlich sind, wird bei den folgenden Optimierungen nur die Anfrage 10: "Stichwortsuche (OR)" betrachtet.

Auffällig am Ergebnis ist, dass nur Tabellen Geschwindigkeitsprobleme haben, die Joins benutzen. Anweisungen, die nur auf einer Tabelle arbeiten sind dagegen sehr schnell und benötigen weniger als 1 Sekunde.

10.2.2 Optimierungsmöglichkeiten

Indexe

DB2 Everyplace ermöglicht das Anlegen von Indexen. Dieses sollte zu einer Geschwindigkeitssteigerung bei Anfragen führen, die auf Elementen arbeiten, auf denen ein Index angelegt wurde. Hierdurch sollten sich sowohl Anfragen auf einer Tabelle als auch Anfragen über mehrere Tabellen beschleunigen lassen.

Das Anlegen eines Indexes ist sehr einfach und ist auch eines der interessantesten Funktionen in DB2 Everyplace.

Denormalisierung

Die Denormalisierung wird eingesetzt, um die Geschwindigkeit mancher Anfragen zu erhöhen. Hierbei wandelt man die Tabellen in eine niedrigere Normalform um und hofft so Joins bei den Anfragen einzusparen. Dies führt zu einer Erhöhung der Redundanz in der Datenbank und zu sehr viel längeren Tabellen und sollte deshalb nur gemacht werden, wenn es keine Alternative dazu gibt.

Eine Denormalisierung würde dazu führen, dass die Daten für die Datenbank noch einmal erstellt werden müssten und wurde wegen des hohen Aufwands nicht gemacht.

Aufbrechen der Joins

Bei der Betrachtung des Ergebnisses fällt auf, dass vor allem Anfragen mit Joins langsam sind. Betrachtet man allerdings die Anfragen auf nur einer Tabelle, so sind diese Anfragen sehr schnell. Durch diese Beobachtung kann man zum Schluss kommen, dass in erster Linie die Joins die Geschwindigkeitsprobleme verursachen. Eine Abschätzung der Ausführungszeit für eine Anfrage, bei der das Programm den Join macht, ist deshalb sinnvoll.

Die betrachtete Anfrage ist die Anfrage 4: "Position einer Person". Soll das Programm die Joins selbst machen, so kann es dies machen, indem es in der Personroom Tabelle nach den roomids sucht, die in den Zeilen der gesuchten personid sind. Dieses ergibt fast immer nur ein Ergebnis. So auch bei personid = 40. Die dazugehörige roomid ist 200 und ist in der Tabelle Roompositions nur einmal vor-

handen. Es werden also ganze zwei Anfragen benötigt. Jede dieser Anfragen kann in weniger als einer Sekunde ausgeführt werden. Der Join ist so schnell, dass er in vernachlässigbar kurzer Zeit ausgeführt ist. Dieses führt zu einer geschätzten Ausführungsdauer von zwei Sekunden. Tatsächlich benötigt die Anfrage mit Joins aber 36 Sekunden. Ein Aufbrechen der Anfragen scheint sich zu lohnen.

Dieses ist erstaunlich, wenn man bedenkt, dass diese Optimierung in herkömmlichen Datenbanken nicht sinnvoll ist und zu langsameren Anfragen führt.

Benutzen von PDBs

Unter der Annahme, dass die Datenbank nicht effizient arbeitet, kann es sinnvoll sein auf das Dateiformat der Palms zu wechseln und die Anfragen selbst zu programmieren. Dieser Wechsel kann bei der bestehenden Softwarearchitektur leicht durchgeführt werden. Die Daten müssen allerdings in PDBs gespeichert werden. Dieses bedeutet einen ziemlichen Aufwand, da ein weiteres Programm hierfür geschrieben werden müsste.

Diese Optimierungsmöglichkeit ist im Rahmen dieser Studienarbeit nicht sinnvoll, da eine Bewertung der Datenbank ohne Datenbank nicht möglich ist.

10.2.3 Ergebnisse durch Einsatz von Indexen

Auswahl der Indexe

Zur Beschleunigung sollten in erster Linie auf Elemente ein Index gelegt werden, mit Hilfe denen Joins durchgeführt werden oder deren Elemente durchsucht werden. Ausserdem werden Indexe auf Primärschlüssel automatisch von DB2 Everyplace angelegt.

Folgende drei Anfragen werden optimiert:

- Anfrage 4: "Position einer Person"
- Anfrage 8: "Personen in Raum"
- Anfrage 10: "Stichwortsuche (OR)"

Bei der Anfrage 4 sollten Indexe auf folgende Elemente gelegt werden (dazugehörige Tabelle in Klammern):

- personid (Persontoroom)
- roomid (Persontoroom)
- roomid (Roompositions)

Bei der Anfrage 8:

- roomid (Persontoroom)
- personid (Persontoroom)

Bei der Anfrage 10:

- personid (Persontokeywords)

- keywordid (Persontokeywords)

Es können 5 Indexe angelegt werden

- I1: roomid (Roompositions)
- I2: roomid (Persontoroom)
- I3: personid (Persontoroom)
- I4: keywordid(Persontokeywords)
- I5: personid(Persontokeywords)

Ausführungsdauer bei Einsatz der Indexe

Tabelle 10-4 zeigt die Ausführungsdauer nach Einsatz von Indexen. In Klammern stehen die angelegten Indexe.

Tabelle 10-4: Ausführungsdauer bei Indexen

Anfrage	Ausführungsdauer (Sekunden)
Position einer Person	34,73 (I1), 2,03 (I1,I2), 2,03 (I1,I2,I3)
Personen in Raum	0,99(I1,2), 1,01 (I1,2,3)
Suche mit Schlüsselwörtern (OR)	15,71(I1,I2,I3,I4), 15,71 (I1,I2,I3,I4,I5)

Nebeneffekt durch Einsatz von Indexen

Die Indexe beschleunigten nicht nur die Ausführung der Anweisungen. Sie benötigen Speicher im Storage Ram aber anscheinend auch im Dynamic Ram. Beim Programm QBE wurden Anfragen mit dem Fehler "Out of Memory" abgebrochen.

10.2.4 Ergebnisse durch Aufbrechen der Joins

Die Aufgebrochenen Anweisungen

Aus der Anweisung zur Personenposition

```
SELECT x1,y1,x2,y2,mapid
FROM Roompositions r, Persontoroom t
WHERE t.personid = ? and t.roomid = r.roomid
```

werden zwei Anweisungen:

```
SELECT roomid
FROM persontoroom
WHERE personid = ?
```

```
SELECT x1,y1,x2,y2,mapid
FROM roompositions
WHERE roomid = ?
```

Aus der Anweisung für die Personen eines Raumes

```
SELECT personid,name
FROM Persondata p,Persontoroom r
WHERE roomid = ? AND p.personid = r.personid
```

werden zwei Anweisungen:

```
SELECT personid FROM Persontoroom WHERE roomid=?
SELECT personid,name FROM Persondata WHERE personid=?
```

Aus der Anweisung für die Personen zu Schlüsselwörtern:

```
SELECT DISTINCT p.personid,p.name
FROM Persondata p, Keywords k, Persontokeywords t
WHERE p.personid = t.personid
AND t.keywordid = k.keywordid AND keyword="
```

werden drei Anweisungen:

```
SELECT keywordid FROM Keywords WHERE keyword=
SELECT personid FROM Persontokeywords WHERE keywordid=?
SELECT personid, name FROM Persondata WHERE personid=?
```

Der Join durch das Programm

Dieser Join ist eine sehr intuitive Form der Realisierung eines Joins:

Als erstes werden sämtliche Join-Elemente geholt, die in der gleichen Zeile sind, wie die Elemente nach denen in der Anfrage gesucht wird.

Im zweiten Schritt wird für jedes Join-Element eine Anfrage an die zu joinende Tabelle gemacht und das Ergebnis wird einer Ergebnismenge hinzugefügt.

Wenn diese Ergebnismenge für den Join mit einer anderen Tabelle benötigt wird, dann dienen diese Elemente als Join Elemente und es wird verfahren, wie im zweiten Schritt. Sonst sind diese Elemente das Ergebnis der Anfrage.

Ein Beispiel:

Tabelle 1: Keyword (kid,keyword) = {(1,"Anwendersoftware"), (2,"Betriebssoftware")}

Tabelle 2: Persontokeywords (pid,kid) = {(1,1), (2,1),(3,1), (3,2),(4,2)}

Tabelle 3: Persondata (pid, name): {(1,"Petra"),(2,"Daniela"),(3,"Cora"),(4,"Christine")}

Sucht man nach den Namen aller Personen mit dem Schlüsselwort "Anwendersoftware" wird folgendes gemacht:

Suche in Tabelle 1 nach allen kid mit keyword "Anwendersoftware". Ergebnis {1}. Danach Suche in Tabelle 2 nach allen pid bei denen kid = 1. Ergebnis: {1,2,3}. Danach werden drei Anfragen nach dem Namen zu einer pid auf Tabelle 1 gemacht. Die Anfrage 1 sucht dabei nach pid = 1, Anfrage 2 sucht nach pid = 2 und Anfrage 3 sucht nach pid = 3.

Ergebnis {"Petra", "Daniela", "Cora"}.

An diesem Beispiel lässt sich gut erkennen, dass ich bei diesem Join sehr viele Anfragen benötige. In diesem Fall sind es bereits 4.

Ausführungsdauer der Befehle

Die Geschwindigkeitssteigerung ist groß. Verwendet man zusätzlich noch Indexe, sind die Anweisungen schnell genug. Ein Find nearest nach dem Schlüsselwort Anwendersoftware dauerte mit aufgebrochenen Joins und Indexen nur noch 4,3 Sekunden.

10.2.5 Vergleich

In der Tabelle 10-5 wird die Ausführungsdauer der Befehle unoptimiert, mit Indexen, aufgebrochen ohne Indexe und aufgebrochen mit Indexen dargestellt. Man sieht, dass die Indexe die Ausführungsdauer bei den Anfragen mit Joins sehr stark reduzieren. Allerdings ist die Ausführungsdauer der Anfragen bei aufgebrochenen Joins deutlich geringer. Kombiniert man beide Möglichkeiten, so verringert sich die Ausführungsdauer weiter. Auf den ersten Blick überraschend ist Anfrage 10: "Stichwortsuche (OR)". Der Einsatz eines Indexes beschleunigt die Anfrage nicht, wenn man bereits aufgebrochene Joins benutzt. Die Erklärung hierfür liegt in der Art, wie das Programm Joins durchführt. Es werden dabei 19 SQL Anfragen ausgeführt. Eine Anweisung für die Keywordid des Stichworts "Anwendersoftware" in der Tabelle Keywords, eine Anweisung für die Personenids in der Tabelle Persontokeywords, deren keywordids der von "Anwendersoftware" entsprechen, und 17 Anweisungen, um die Daten zu jedem Mitarbeiter mit einer gefundenen personid in der Tabelle Persondata zu erhalten. Die Anwendung der Indexe aus Abschnitt 10.2.3 "Ergebnisse durch Einsatz von Indexen" beschleunigt nur einen Befehl. Die 17 Anweisungen, die die Mitarbeiterdaten erhalten, lassen sich nicht durch Anlegen von zusätzlichen Indexen beschleunigen, da personid in der Tabelle Persondata ein Primärschlüssel ist und DB2 Everypplace auf Primärschlüssel automatisch Indexe anlegt.

Tabelle 10-5: Vergleich der Geschwindigkeit

Anfrage	unoptimiert	mit Indexen	Aufgebrochene Joins ohne Indexe	Aufgebrochene Joins mit Indexen
Anfrage 4: "Position einer Person"	36,28	2	0,43	0,17
Anfrage 8: "Personen in Raum"	27,26	3,5	0,23	0,07

Tabelle 10-5: Vergleich der Geschwindigkeit

Anfrage	unoptimiert	mit Indexen	Aufgebrochene Joins ohne Indexe	Aufgebrochene Joins mit Indexen
Anfrage 10: "Stichwortsuche (OR)"	37,17	16	1,43	1,41
find nearest nach "Anwendersoftware"	654,93	50	8,74	4,3

10.2.6 Bewertung des Ergebnisses

Das Ergebnis der Optimierungen ist überraschend. Die Optimierung durch aufbrechen der Joins ist bei diesen Tabellen besser als der Einsatz von Indexen, obwohl das Aufbrechen der Joins bei herkömmlichen Datenbanken keine Beschleunigung, sondern eine Verlangsamung zur Folge hat. Das Aufbrechen der Joins beschleunigt die Anfragen so stark, dass sie die notwendige Ausführungsgeschwindigkeit erreichen. Der Einsatz von Indexen scheint sich nicht zu lohnen, da die Geschwindigkeitssteigerung von einem Anwender kaum bemerkt werden kann. Für die einzelnen Anweisungen ist die Aussage richtig, aber wenn die Anweisungen mehrmals hintereinander ausgeführt werden müssen, kann es sich doch lohnen. Beispielsweise wird bei einem find nearest zuerst eine Stichwortsuche durchgeführt und dann wird die Position eines jeden gefundenen Mitarbeiters geladen. Dieses führt, wenn man die Anfrage 4: "Position einer Person" und die Anfrage 10: "Stichwortsuche (OR)" beim Stichwort "Anwendersoftware" dazu, dass die Anfrage 4 mehrmals aufgerufen wird. Der Unterschied beträgt dann 4,4 Sekunden.

Benutzt man nur Indexe ist die Geschwindigkeit der Anfragen zu niedrig. Die Ausführungsdauer eines find nearest nach dem Stichwort Anwendersoftware benötigt 50 Sekunden. Dieses dürfte für die meisten Anwender nicht mehr akzeptabel sein.

10.2.7 Bewertung der sonstigen Einsatzgebiete

In diesem Abschnitt wird beschrieben, wie DB2 Everyplace die einzelnen Probleme löst, die in Abschnitt 2.5.1 "Motivation für den Einsatz einer Datenbank" als Gründe für den Einsatz von Datenbanken beschrieben sind.

Redundanzen und Inkonsistenzen

Durch den Einsatz von DB2 Everyplace können Redundanzen und Inkonsistenzen der Daten auf dem Palm verhindert werden. Inkonsistenzen der Daten zwischen Palm und Desktop lassen sich allerdings nicht verhindern, da Daten nur zum Synchronisationszeitpunkt abgeglichen werden können. DB2 Everyplace unterstützt das Entdecken dieser Inkonsistenzen, indem es bei der Synchronisation der Daten Inkonsistenzen zwischen den Daten auf dem Desktop und auf dem Palm aufdecken versucht. Es stehen auch Werkzeuge zur Verfügung, die Inkonsistenzen der Daten automatisch beheben können.

Eingeschränkte Zugriffsmöglichkeiten

Durch den Einsatz von SQL, ermöglicht es die Datenbank komplexere Verknüpfungen der Daten durchzuführen als sie für PDB Dateien zur Verfügung stehen. Die Zugriffsmöglichkeiten sind durch den geringeren Speicher und die geringere Rechenleistung der Palmgeräte gegenüber Datenbanken in herkömmlichen Rechnersysteme, wie z.B. DB2 sehr stark eingeschränkt. Betrachtet man die Anfragen des Programms, so stellt man allerdings fest, dass die einzige wirklich notwendige Erweiterung eine Funktion zum Bilden der Schnittmenge mehrerer Anfragen ist.

Leider ist die Geschwindigkeit in der vorliegenden Version zumindest für das im Rahmen dieser Studienarbeit geschriebene Programm nur ausreichend, wenn man die Verknüpfungen durch das Programm machen lässt und damit auf komplexe Verknüpfungsmöglichkeiten verzichtet.

Probleme des Mehrbenutzerbetriebs

Da Palms keinen Mehrbenutzerbetrieb und auch keine parallele Ausführung mehrerer Programme ermöglichen, gibt es das Problem auf dem Palm nicht. Das Problem entsteht erst, wenn die Daten mit dem Desktop synchronisiert werden müssen und mehrere Personen diese Daten verändern können. Hierbei entstehen Inkonsistenzen zwischen den Daten auf dem Palm und dem Desktop.

Verlust der Daten

Der Verlust von Daten ist für DB2 Everyplace in der Version 7.1 ein großes Problem. Den Verlust der Daten durch Hardwaredefekte kann die Datenbank nicht verhindern. Der Schutz der Daten vor Softwarefehler d.h abstürzende Programme ist noch geringer als bei PDB Dateien. Bei der Entwicklung des Programms habe ich oft die Situation gehabt, dass die Daten in der Datenbank nach einem Absturz gelöscht waren oder ein Zugriff auf diese nicht mehr möglich war. Dies ist umso überraschender, wenn man bedenkt, dass das Programm nur lesen Zugriff auf die Daten benutzt. Hätte das Programm PDB-Dateien benutzt wären die Daten aufgrund des hardwaremäßigen Schreibschutzes sicher gewesen. Dieses Verhalten ist für eine Datenbank absolut inakzeptabel.

Integritätsverletzung

DB2 Everyplace erlaubt es Integritätsbedingungen zu setzen. Hierdurch ist nicht sichergestellt, dass die Integritätsbedingungen auch wirklich eingehalten werden. Das Problem liegt bei Integritätsbedingungen, die man über mehrere Geräte überprüfen müsste. Aufgrund der Verzögerung durch die Zeit zwischen den Synchronisationsvorgängen wird die Verletzung der Integritätsbedingung erst bei der Synchronisation erkannt. Ein Beispiel zeigt dieses Problem:

Die Integritätsbedingung die eingehalten werden soll, lautet, dass die Personen A und B gemeinsam nicht mehr als 100 DM ausgeben dürfen. Die Überwachung sollen dabei Palms übernehmen. Die Daten werden nun synchronisiert. Da beide noch nichts ausgegeben haben, beträgt die Geamtausgabe 0 DM.

Person A möchte 70 DM ausgeben und überprüft auf seinem Palm, ob dieses möglich ist. Der Palm stellt fest, das Person A und Person B 0 DM ausgegeben

haben. Person A kann also die 70 DM ausgeben. Person B möchte 50 DM ausgeben. Er lässt seinen Palm überprüfen, ob dieses möglich ist. Der Palm hat immer noch die Daten des letzten Synchronisationszeitpunktes und stimmt zu.

Die Verletzung der Integritätsbedingung fällt erst auf, wenn beide Personen ihre Daten synchronisiert haben. DB2 Everyplace kann die Integritätsbedingung zwar nicht sicherstellen, aber zumindest feststellen.

Sicherheitsprobleme

Die Datenbankkomponente besitzt keinerlei Sicherheitsmechanismen, um den Zugriff der Daten vor Unbefugten zu schützen. Der Synchronisationsserver besitzt dagegen Sicherheitsfunktionen. Er überprüft bei einer Synchronisation die Zugriffsrechte des Benutzers auf die Desktopdaten und verweigert die Synchronisation bei fehlenden Zugriffsrechten.

Der Grund für die fehlenden Sicherheitsfunktionen in der Datenbankkomponente liegt in der Art der Benutzung der Palms. Man startet sie, damit man schnellen und einfachen Zugriff auf seine Daten hat. Viele Benutzer würden eine Passwortabfrage bei der Bedienung als störend empfinden. Die Sicherheit musste deshalb der Bedienbarkeit weichen.

Hohe Entwicklungskosten

Die Entwicklungskosten von Palmprogrammen dürfte weit unter denen von Desktopprogrammen liegen, da sie viel weniger Funktionalität besitzen. Trotz allem ist eine Reduktion der Entwicklungskosten sinnvoll. Die Datenbank reduziert die Entwicklungskosten, indem sie die Entwicklung und Wartung der Synchronisationsprogramme überflüssig macht. Der Zugriff auf die Datenbank über eine ODBC Schnittstelle ermöglicht einen relativ leichten Zugriff auf die Daten. Da der Zugriff auf allen von DB2 Everyplace unterstützten Plattformen gleich ist, reduzieren sich die Kosten einer Portierung.

10.2.8 Schlussfolgerungen zur Datenbank

Da das Programm seine Daten nicht synchronisieren muss, habe ich keine praktische Erfahrung mit der Synchronisationskomponente von DB2 Everyplace. Ich beschränke mich deshalb auf Aussagen zur Datenbankkomponente von DB2 Everyplace.

Die Datenbankkomponente hat momentan zwei große Probleme:

- Die Geschwindigkeit beim Einsatz von Joins
- Der mögliche Verlust von Daten bei abstürzenden Programmen.

Diese Probleme sollten aber in den Griff zu bekommen sein und zeigen in erster Linie, dass das Gebiet der mobilen Datenbanken relativ jung ist. Weiterhin muss man bedenken, dass die Datenbank mit sehr wenig Speicher und Rechenleistung auskommen muss. Deshalb muss man auch Abstriche im Bereich der Zugriffsmöglichkeiten machen. Da die Daten auf den Palms im Normalfall nicht verknüpft werden, ist dieses allerdings für viele Anwendungen akzeptabel.

10.3 Bewertung der Anwendung

Nachdem die Datenbank bewertet wurde, werden in diesem Kapitel die Anwendung bewertet. Hierbei wird die Navigation auf den Detailkarten, die Positionierung und die find nearest Funktion betrachtet. Für eine Bewertung des Programmes ist es auch wichtig Funktionen zu erwähnen, die implementiert werden sollten. Wenn unnötige Funktionalität implementiert wurde, sollte auch dieses erwähnt und begründet werden, warum diese Funktionalität unnötig ist. Eine Betrachtung der Möglichkeiten, die sich durch eine Migration auf die Nexusplattform ergeben, zeigt welche Probleme des Programms gelöst werden und welche neuen entstehen.

10.3.1 Bewertung der Navigation

Die Navigation ist für ein kleines Gebiet ausgelegt, da nur auf den Karten zu einem Gebäude navigiert werden soll. Ein Problem ist, dass sichtbare Navigationstasten Platz verbrauchen. Deshalb werden die Navigationstasten nicht dargestellt. Leider erkennt man nun nicht, dass sie existieren. Die Benutzer müssen deshalb in die Bedienung des Programms eingewiesen werden. Die Navigationstasten sollten deshalb doch sichtbar sein.

10.3.2 Bewertung der Positionierung

Die Positionierung mit Hilfe der Karten hat Probleme. Die Benutzer versuchen sich anhand der Raumnummern im Gebäude zu orientieren und werden versuchen den Raum mit dieser Raumnummer auf den Detailkarten zu finden. Da viele Benutzer nicht wissen, wie die Räume nummeriert sind und in welchem Gebäudeflügel sie sich aufhalten, werden sie die Detailkarten eines Stockwerks nach dem Raum durchsuchen. Bei der Benutzung des Prototypen kann der Benutzer dieses umgehen, indem er nach der Position eines Mitarbeiters in dem Raum sucht. Da im Prototyp die Position der Mitarbeiter identisch mit der Position ihres Arbeitsraumes ist, kann der Benutzer so den Raum finden. Sobald die Position der Mitarbeiter nicht mehr identisch mit der Position des Raumes ist, funktioniert dies nicht mehr.

10.3.3 Bewertung der find nearest Funktion

Die find nearest Funktion ist für Besucher eine nützliche Anfrage. Die Implementierung ist für die momentane Datenbasis ausreichend schnell. Die Darstellung des Ergebnisses in einer Liste ist für kleinere Ergebnismengen akzeptabel. Die Eingabe der Stichwörter, indem man auf die Elemente einer Liste klickt, ist geschickt und ermöglicht eine schnelle Eingabe der Stichwörter. Ein Problem ergibt sich, wenn die Daten aktuell gehalten werden müssen z.B. wenn die aktuelle Position der Mitarbeiter benötigt wird, da keine ständige Synchronisation der Daten möglich ist.

10.3.4 Erweiterungen des Programms

Eine sinnvolle Erweiterung des Programms wäre die Möglichkeit der Raumsuche. Diese Funktion wird von vielen Besuchern der Fakultät benötigt, da diese oft nach Hörsälen suchen. Hierdurch würde sich auch die Positionierung vereinfachen. Weiterhin könnte man das Programm so ausbauen, dass es auch andere

Gebiete darstellen kann und weitere Informationen anzeigen kann. Beispielsweise könnte man sich Informationen zu einzelnen Stichwörtern anzeigen lassen wollen. Ein weiterer Bereich betrifft die Datenbank. Diese kann durch eine andere ersetzt werden oder um ein Vergleich dieser Datenbanken zu erhalten könnte man sie auch zusammen in das Programm einbauen und einen Wechsel zur Laufzeit ermöglichen.

10.3.5 Nicht benötigte Funktionen

Der SQL Parser ist nicht im Prototypen implementiert und sollte auch nicht innerhalb des Programms implementiert sein. Sinnvoller ist es, ein Programm zu entwickeln, dessen einzige Aufgabe es ist, dem Anwender die Eingabe von SQL-Anweisungen zu ermöglichen. Hierdurch muss der Anwender nicht mehr das Programm starten, wenn er mit SQL-Befehlen auf die Datenbank zugreifen möchte.

10.3.6 Änderungen bei einer Migration auf die Nexusplattform

Bei einer Migration des Programms auf die Nexusplattform ergeben sich Änderungen am Programm. Die Daten werden in externen Servern gespeichert und können auch aktuell gehalten werden. Man ist beispielsweise durch den Einsatz von Sensoren in der Lage die aktuelle Position der Mitarbeiter zu speichern. Ein weiterer Vorteil ergibt sich dadurch, dass die gespeicherten Datenmengen sehr viel größer sein können als bei einer Speicherung auf dem Palm. Hierdurch kann man Daten über ein größeres Gebiet speichern. Die Geschwindigkeit der externen Server dürfte auch weit über der Geschwindigkeit des Palms sein und ermöglicht so komplexere Anfragen an die Datenbank. Die Kommunikation wird über Funknetze erfolgen. Da der Palm keine große Leistungsfähigkeit hat, müssen die Kommunikationsprotokolle vom Palm effizient verarbeitet werden können. Weiterhin muss darauf geachtet werden, dass die Datenmengen die ausgetauscht werden klein bleiben. Beispielsweise zeigt das Programm im Fenster für die Stichwortsuche alle Stichwörter an, die das Programm kennt. Dies ist möglich, da es nur wenige Stichwörter in der Datenbank gibt. Wird ein Nexusdienst benutzt, der sämtliche möglichen Stichwörter für die ganze Welt verwaltet, so ist die Anzeige aller Stichwörter nicht mehr möglich.

Sobald die Nexusplattform ein größeres Gebiet als die Fakultät unterstützt, wird man wahrscheinlich die Navigation und die Positionierung so anpassen müssen, dass sie auch für größere Gebiete geeignet sind. Die Karten werden dann auch durch Nexusdienste bereitgestellt werden müssen, da der Palm die Karten für große Gebiete nicht speichern kann.

10.4 Zusammenfassung

Die Datenbank hat noch Probleme mit der Geschwindigkeit der Joins. Verzichtet man auf diese Verknüpfungsmöglichkeit, so ist die Datenbank für die Daten der Fakultät ausreichend schnell. Der Einsatz von Indexen beschleunigte die Datenbankabfragen. Bei größeren Datenbeständen dürfte der Einfluss noch sehr viel stärker werden. Ein großes Problem besteht in der Gefahr Daten in der Datenbank durch einen Absturz zu verlieren. Diese Probleme lassen sich darauf zurückführen, dass mobile Datenbanken ein noch sehr junges Themengebiet sind.

Die Anwendung hat eine Navigation und eine Positionierung, die für das Gebäude ausgelegt sind. Die Eingabemöglichkeiten der Stichworte ist für die bisherige Anzahl an Stichwörter (22) gut geeignet. Sinnvolle Erweiterungen des Programms ist eine Suche nach Räumen und die Möglichkeit sich mehr Informationen zu Stichworten anzeigen zu lassen. Der SQL-Parser sollte nicht im Programm implementiert sein, sondern sollte stattdessen in einem eigenen Programm realisiert sein. Durch eine Migration auf die Nexusplattform kann das Programm auch für größere Gebiete eingesetzt werden. Dies führt zu Änderungen, die Navigation und Positionierung betreffen. Auch die Eingabe der Stichwortsuche muss dann überdacht werden und die Ergebnisse der räumlichen Anfrage müssen sinnvoll begrenzt werden können.

11 Zusammenfassung und Ausblick

Durch die Verbreitung mobiler Computer werden ortsabhängige Informationssysteme immer wichtiger. Die Beispielanwendung sollte für palmkompatible Geräte entwickelt werden. Diese Geräte sind darauf ausgelegt eine Symbiose mit Desktopsystemen einzugehen. Die rechenintensiven und eingabeintensiven Anwendungen werden dabei auf den Desktopsystemen ausgeführt. Der Anwender kann die Daten durch die Palms mit sich führen und wenn nötig kleine Änderungen an diesen durchführen. Für die Entwicklung auf Palmsystemen stehen unterschiedliche Werkzeuge zur Verfügung. Eingesetzt wurde die Programmiersprache C. Als mobile Datenbank wurde DB2 Everyplace von der Firma IBM eingesetzt. Als Vorgehensmodell wurde ein Phasenmodell genommen, wobei Prototypen in der Anforderungsanalyse erstellt wurden. Die Anforderungen an die Anwendung sind unterschiedlicher Natur. Die Benutzer benötigen eine leichte Bedienung, während die Entwickler des Nexusprojektes auf eine gute Erweiterbarkeit und Wiederverwendbarkeit des Programms benötigen. Die Entwickler der Abteilung Anwendersoftware wollten eine Anwendung mit der sie mobile Datenbanken bewerten können. Das Programm ermöglicht die Orientierung im Gebäude. Die Benutzer können ihre Position auf einer Karte markieren und sind in der Lage Informationen zu Mitarbeitern und Räumen zu erhalten. Das Programm ermöglicht die Ausführung räumlicher Anfragen. Hierbei werden Mitarbeiter zu Stichworten gesucht und dann nach der Entfernung zum Benutzer sortiert. Die Architektur der Nexusplattform diente als Ausgangspunkt für die Architektur des Programms. Die Architektur des ersten Prototypen dieser Architektur wurde ebenfalls betrachtet. Es zeigte sich beim Entwurf der Architektur, dass diese nicht durch palmtypische Eigenschaften beeinflusst wird. Der Entwurf der Karte, die das Programm darstellen sollte, wurde dagegen durch Bildschirmgröße des Palms sehr stark beeinflusst. Die Realisierung der räumlichen Anfrage zeigt, dass einfache räumliche Anfragen, wie die find nearest Funktion, an die Anforderungen des Benutzers angepasst werden müssen. Es stellte sich dabei heraus, dass ein echtes find nearest bei der Ausgabe eines Mitarbeiters nicht sinnvoll ist. Eine Ausgabe mehrerer Benutzer benötigt eine sinnvolle Sortierung. Hierbei ist der Einsatz von nur positionsabhängigen Sortierungen wegen der vielen gleichweiten Personen problematisch. Eine Kombination mit anderen Sortierungen bringt Fortschritte.

Bei der Evaluation der Datenbank stellten sich Probleme mit der Geschwindigkeit und der Datensicherheit heraus. Diese Probleme lassen sich darauf zurückführen, dass mobile Datenbanken ein sehr junges Themengebiet sind.

Die Navigation auf den Karten und die Positionierung mit Hilfe des Programms ist für ein kleines Gebiet, wie die Fakultät ausgelegt. Bei einer Migration auf die Nexusplattform ist es wahrscheinlich, dass dieses Programm auch für größere Gebiete eingesetzt werden soll. Dies führt zu Änderungen in der Bedienung und der Architektur des Programms. Die Navigation und die Positionierung müssten dann geändert werden. Das Programm benötigt nach einer Migration auch die Module für die Datenspeicherung und die geographischen Funktionen nicht mehr. Da nur wenige Daten in einem Palm gespeichert werden können, gibt das Programm nur wenig Informationen aus. Mit Hilfe der Nexusplattform wäre das

Programm in der Lage auch auf große Informationsquellen zuzugreifen und könnte so auch Dienste anbieten, die sehr viel mehr Information darstellen und verwalten müssen. Beispielsweise könnte man das Programm so erweitern, dass der Benutzer zu den einzelnen Stichworten auch Informationen anzeigen lassen kann. Dies ermöglicht die Entwicklung komplett neuer Anwendungen. Mit Hilfe einer automatischen Positionierung ist es möglich einem Besucher der Fakultät eine Tour durch das Gebäude anzubieten. Hierbei könnte dieser angeben, welche Informationen er bei dieser Tour sehen möchte z.B. Projekte an denen die Mitarbeiter in den Räumen arbeiten. Läuft der Benutzer an einem Raum vorbei, so werden auf dem Palm die Namen der Projekte der Mitarbeiter aufgezeigt. Der Benutzer kann sich dann Informationen zu diesen Projekten anzeigen lassen.

Eine weitere Möglichkeit wäre ein Programm mit dessen Hilfe der Anwender zu einem Raum oder zu einer Person geführt werden kann. Ein Bedarf hierfür besteht, da viele Besucher Hörsäle oder bestimmte Personen suchen.

Der Einsatz mobiler Datenbanken dürfte sich in den nächsten Jahren verstärken, da sie die Entwicklungskosten für Palmprogramme reduzieren und Probleme wie Integritätsprobleme aufdecken können. Da sich die Rechenleistung der mobilen Computer laufend steigert, ist damit zu rechnen, dass das Hauptaugenmerk bei der Entwicklung mobiler Datenbanken nicht mehr die Entwicklung noch effizienterer Anfragemethoden sein wird, sondern die Probleme, die durch die Verteilung der Daten auf viele Palmgeräte entstehen, in den Griff zu bekommen. Die Bewertung von mobilen Datenbanken muss deshalb auch die Methoden betrachten mit denen sie versuchen diese Probleme einzudämmen.

A Literaturverzeichnis

[DAVIS 1999]

Davies, Judit R.

A Tiny DB2 for a Pervasive Computing World

Sommer 1999, Letzter Besuch Dezember 2000

http://www.db2mag.com/db_area/archives/1999/q2/99sp_davis.shtml

[DB2E DEVELOPMENT GUIDE 2000]

Firma IBM

Application Development Guide for Palm OS Devices

Online Dokumentation zu DB2 Everyplace Version 7

[FAKULTÄTS GEBÄUDEPLAN 2000]

Fakultät Informatik

Gebäudeplan Breitwiesenstr. 20/22

<http://www.informatik.uni-stuttgart.de/plaene/gebaeude.html>

[GRZAN 2000]

Grzan, Stjepan

Spezifikation zur Studienarbeit Konzeption und Entwicklung einer mobilen Datenbankanwendung

internes Dokument IPVR Version 1.2, August 2000

[HANDSPRING 2000]

Firma Handspring

HandSpring: Visor Deluxe

Letzter Besuch Dezember 2000

http://handspring.modusmedia.nl/handspring_duits/products/visor-deluxe/compare.asp

[HATLER FAQ]

Hatler, Wade

Pilot Programming FAQ

Laut History: Stand 1998, Letzter Besuch Dezember 2000

<http://www.wademan.com/Pilot/Program/FAQ.htm>

[HELAL ET AL 1999]

Helal, Abdelsalam (Sumi); Haskell, Bert; Carter, Jeffrey; Brice, Richard; Woelk, Darrel; Rusinkiewicz, Marek

Any Time, Anywhere Computing: Mobile Computing Concepts and Technology

Kluwer Academic Publishers, Norwell, Massachusetts, 1999, 1. Auflage

[HERCEG 1994]

Herceg, Michael

Software Ergonomie: Grundlagen der Mensch-Computer-Kommunikation

Addison-Wesley, Bonn; Paris; Reading, Mass. , 1994, 1.Auflage

[KEMPER & EICKLER 1999]

Kemper, Alfons; Eickler, Andre

Datenbanksysteme: eine Einführung

Oldenbourg Verlag, München, 1999, 3. korr. Aufl.

[LEONHARDI ET AL 1999]

Leonhardi, Alexander; Kubach, Uwe; Rothermel, Kurt; Fritz, Andreas

Virtual Information Towers - A Metaphor for Intuitive, Location-Aware Information Access in a Mobile Environment

Proceeding des dritten International Symposium on Wearable Computers (ISWC'99), San Francisco, Kalifornien, IEEE Press, 1999

[MEßMER 2001]

Meßmer, Jens

Modellierung der Augmented World in Nexus

Diplomarbeit Nr. 1870, Fakultät Informatik, Universität Stuttgart, 2001

[NICKLAS 2000]

Nicklas, Daniela (Editor)

Final Report of Design Workshop (10.-15.7.2000)

internes Dokument Nexus Projekt Version 1.1,

[PALM DEVELOPMENT 2000]

Firma Palm Inc.

Palm Development

Letzter Besuch Dezember 2000

<http://www.palmos.com/dev/>

[PALM PROGRAMMER'S COMPANION 2000]

Firma Palm, Inc.

Palm OS Programmer's Companion

Palm SDK Dokument, Version 3.5, 2000

[PALM SDK REFERENCE 2000]

Firma Palm, Inc.

Palm OS SDK Reference

Palm SDK Dokument, Version 3.5, 2000

[PRCTOOLS 2000]

PRC-Tools Projekt

GCC for Palm OS

Stand April 2000, Letzter Besuch Dezember 2000

<http://sourceforge.net/projects/prc-tools/>

[PUMATECH 2000]

Firma Pumatech, Inc
Satellite Forms Software
Letzter Besuch Dezember 2000
http://www.pumatech.com/satforms_ee.html

[RHODES & MCKEEHAN 1999]

Rhodes, Neil; McKeehan, Julie
Palm Programming: The Developer's Guide
O'Reilly, Sebastopol, California, 1999, 1.Aufl.

[VA MICRO EDITION 2000]

Firma IBM
Visual Age Micro Edition
Letzter Besuch Dezember 2000
<http://www.embedded.oti.com/>

[VILIS SYSTEM ENTWURF 2000]

ViLiS Studienprojektgruppe
System Entwurf zum Studienprojekt Virtuelle Litfasssäule
Internes Dokument Abteilung Verteilte Systeme, IPVR Universität Stuttgart, Version 2.0, Juli 2000

Erklärung

Ich versichere, dass ich die Arbeit selbstständig verfasst und nur die angegebenen Hilfsmittel verwendet habe.

(Stjepan Grzan)