

Universität Stuttgart

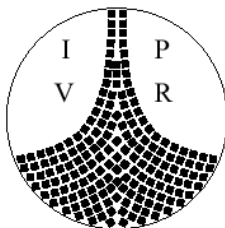
Fakultät Informatik

Prüfer: Prof. Dr. B. Mitschang
Betreuer: Dipl. Inf. Matthias Großmann
begonnen am: 1.11.2001
beendet am: 30.4.2002
CR-Klassifikation: B.4.3, C.2.4, C.2.6, D.4.4, H.2.8.

Studienarbeit Nr. 1836

VIT-Browser für PDAs

Xin Cao



Institut für Parallele und Verteilte
Höchstleistungsrechner(IPVR)
Abteilung Anwendersoftware

Breitwiesenstr. 20-22
70565 Stuttgart



Kurzfassung

Im Rahmen des Projekts NEXUS entsteht eine offene Plattform für ortsbezogene Informationen. Eine zentrale Komponente der Plattform sind Spatial Model Server, die zur Speicherung von Daten zu statischen Objekten dienen. Ein Beispiel für solche Objekte sind Virtuelle Litfasssäulen(VITs), die einen Ortsbezug für Informationen in Form von HTML-Dokumenten herstellen. Diese Seiten können mit einem portablen Gerät, das über eine drahtlose Kommunikationsschnittstelle und einen Positionssensor verfügt, abgerufen werden.

Ziel der Arbeit ist die Entwicklung eines VIT-Browsers für ein Endgerät, das aus einem PDA und einem Mobiltelefon besteht. Zur Positionierung sollen Daten über benachbarte Funkzellen aus dem Mobiltelefon ausgelesen werden. Die Virtuellen Litfasssäulen sind auf einem Spatial Model Server gespeichert. Der zu entwickelnde Browser soll Anfragen an die Schnittstelle dieser Server stellen, um entsprechende VITs zu erhalten, und die ortsgezogenen Informationen, die durch aktive Poster hingewiesen werden, aus dem Internet nehmen und auf dem PDA-Bildschirm anzeigen.

Inhaltverzeichnis

1. Einleitung.....	11
2. Das Positionierungssystem.....	15
2.1 Global System für Mobile Telekommunikation.....	16
2.1.1 Der Aufbau von GSM-System.....	16
2.1.2 Base Station.....	17
2.2 Zelle Datenbank.....	18
2.2.1 Beschreibung von der Zelle Datenbank.....	18
2.2.2 Verwendung von Zelle Datenbank.....	19
2.3 Berechnung von der Position eines Handy-Benutzer.....	21
2.3.1 Filtrierung der Input Daten.....	21
2.3.2 Koordinatensystem.....	23
2.3.3 Zelle basierendes Algorithmus.....	24
2.3.4 Trangulation Algorithmus.....	25
2.4 Zusammenfassung.....	29
3. Einführung in das NEXUS-System.....	31
3.1 Das Ziel des NEXUS-Systems.....	31
3.2 Die NEXUS-Architektur.....	32
3.3 Augmented Area.....	33
3.4 Augmented World Model.....	34
3.5 Standard Class Schema.....	35
3.6 Extended Class Schema.....	35
3.7 Komponenten des NEXUS-Systems.....	36
3.7.1 Spatial Model Service.....	36
3.7.2 Area Service Register(ASR).....	37
3.7.3 Die Föderationsebene.....	37
3.8 Der Datenaustausch.....	38
3.9 Zusammenfassung.....	38
4. Grundlagen für das VIT-System.....	39
4.1 Virtual Information Towsers.....	40
4.1.1 Das Konzept für VIT	40
4.1.2 VITs im NEXUS.....	40
4.2 Das Daten Format	42
4.2.1 Augmented World Query Language.....	43
4.2.2 Augmented World Modeling Language.....	44
4.3 Zusammenfassung.....	45
5. Palmprogrammierung.....	47
5.1 Grundlängen der Palmprogrammierung.....	47
5.1.1 Einsatz von Palmprogrammierung.....	47
5.1.2 Überblick von Palm OS.....	47

5.1.3 Palm OS Emulator.....	49
5.2 C für Palm OS.....	50
5.2.1 Die Entwicklungstools für C.....	50
5.2.2 GNU Pilot SDK.....	50
5.3 Java für Palm OS.....	52
5.3.1 Konfiguration und die CLDC.....	52
5.3.2 Die MID Profile(MIDP).....	53
5.3.3 Die Entwicklung mit J2ME für Palm OS.....	53
5.4 Zusammenfassung.....	55
6. Die Implementierungsaspekte.....	57
6.1 Die Grundidee.....	57
6.2 Die Vorbereitungsphase.....	58
6.2.1 Kurze Beschreibung.....	58
6.2.2 Die verwendete Tools.....	58
6.2.2.1 Mobiltelefon.....	59
6.2.2.2 Palm-Gerät.....	59
6.2.2.3 Zwischenserver.....	62
6.2.2.4 Spatial Model Server.....	64
6.2.3 Datenflüsse.....	66
6.3 Die Browsephase.....	71
6.3.1 Der Arbeitsablauf.....	72
6.3.2 HTML oder WML.....	74
6.3.2.1 Das Problem für HTML auf PDA.....	74
6.3.2.2 Was ist WML?.....	75
6.3.2.3 Einschränkung von WML.....	77
6.4 Zusammenfassung.....	77
7. Zusammenfassung und Ausblick.....	79
Literaturverzeichnis.....	81

Abbildungsverzeichnis

Abbildung 1-1: Die grobe Architektur von dem Dienst-System.....	11
Abbildung 2-1: Sechseckige Einrichtung von Sektoren.....	16
Abbildung 2-2: Umsetzung von FCN zu Zelle ID.....	20
Abbildung 2-3: Das Verlauf des Positionierungsalgorithmus	21
Abbildung 2-4: Die Form eines Sektors mit Empfang hinter der BTS	25
Abbildung 2-5: Vier Möglichkeiten für Kreuzung von Kreisen	27
Abbildung 3-1: Übersicht der NEXUS-Architektur	32
Abbildung 3-2: Augmented Area Model	33
Abbildung 3-3: Anwendungen benutzen die Klassen gemeinsam in NEXUS	35
Abbildung 3-4: Die Funktionalität vom Area Service Register.....	37
Abbildung 4-1: Die grobe Beschreibung für den Arbeitsverlauf	41
Abbildung 4-2: Datenaustausche zwischen NEXUS-Anwendungen und Plattform	42
Abbildung 5-1: POSE auf dem Desktop	49
Abbildung 5-2: Arbeitsweise von GNU Pilot SDK	51
Abbildung 5-3: Die Beziehung zwischen J2ME, J2SE und J2EE	52
Abbildung 5-4: Die MIDlet Entwicklungsprozeß	54
Abbildung 6-1: Die Beziehung zwischen Positionierungs- und Browsephasen	57
Abbildung 6-2: Der Aufbau des Systems in Vorbereitungsphase	59
Abbildung 6-3: Socket Aufruf für verbindungsorientierendes Protokoll	61
Abbildung 6-4: Der Aufbau eines DOM Baums	63
Abbildung 6-5: Programmflüsse des JDOM Dokumentes	64
Abbildung 6-6: Datenflüssen zwischen die Elemente	66
Abbildung 6-7: Die Architektur über die VIT-Informationen	72
Abbildung 6-8: VIT-Liste und Poster-Liste auf dem Palmbildschirm	72
Abbildung 6-9: Die Datenflüsse in der Browsephase	73
Abbildung 6-10: Ein einfacher Text über Marktplatz	75

Tabellenverzeichnis

Tabelle 2-1: Die Eigenschaften von einer BTS.....	19
Tabelle 2-2: Überblick für Zelle Auswahl Module.....	22
Tabelle 2-3: Format von Koordinaten.....	23
Tabelle 6-1: Das BTS-Model.....	65

1. Einleitung

Heute spielt die mobile Kommunikation immer wichtiger Rolle im unseren Leben. Mit einem Mobiltelefon kann man fast überall erreicht werden, und viele Leute verfügen sogar ein Personal Digital Assistant(PDA). Solche mobile Geräte ermöglichen die Leute nicht nur zu telefonieren, egal zu Hause oder auf der Bus-Haltstelle, sondern noch einfach auf das Internet zu zugreifen. In die Zukunft, die Kommunikationsverbindungen werden immer schnelle, und der Bildschirm auf dem mobilen Gerät wird auch immer größer, um mehr Information anzuzeigen.

Die mobile Kommunikation hat sich schon schnell und modern entwickelt, aber es ist immer noch nicht genug für unsere immer mehrere Anforderungen. Es gibt noch viele Anwendungen, die unsere Leben bequemer und einfacher machen können. Zum Beispiel, Sie spazieren in der Stadt, und plötzlich haben Sie Hunger, dann möchten Sie natürlich wissen, ob ein Restaurant in der Nähe von Ihnen gibt, und wie sieht das Menü von dem aus. Stellen wir uns vor, dass das Restaurant eine eigene Web-Seite in dem Internet hat und Sie auch sofort auf die Stelle auf das Internet zugreifen können, aber trotzdem taucht das Problem auf, d.h. Sie wissen nicht welches Restaurant in der Nähe steht, und Sie vielleicht wissen auch nicht, auf welcher Position in der Stadt Sie überhaupt stehen. Und das Restaurant weiß selbstverständlich auch nicht, dass Sie Hunger haben. Dann können Sie leider nur den anderen Leute mal abfragen, und das Menü können Sie normalerweise nur persönlich in dem Restaurant erfahren. Aber wenn Sie jetzt ein mobiles Gerät haben, das sofort Ihnen auf dem Bildschirm zeigen, auf welche Position Sie sind und die ganzen Informationen von einem Restaurant, das in der Nähe von Ihnen steht. Dann fühlen Sie sich sicher viel bequemer und leichter.

Um so eine Dienst zu realisieren, ein Positionierungssystem ist erforderlich, außerdem brauchen wir ein Informationssystem, um eine bestimmte Position und eine entsprechende Information zu verbinden, und die ortsbezogenen Informationen anzubieten. Endlich brauchen wir natürlich noch ein Endgerät, um die Information anzuzeigen. Abbildung 1-1 zeigt die grobe Architektur von dem ganzen System.

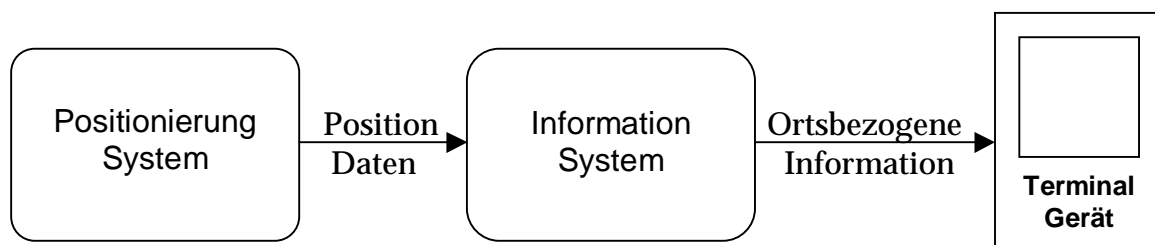


Abbildung 1-1: Die grobe Architektur von dem Dienst-System

Zuerst kucken wir das Positionierungssystem mal an. Dieses System bietet einen sogenannten Location-Based Services(LBS) an, welcher gilt nur für eine bestimmte Position, d.h. nur die Information über das Restaurant, das in der Nähe von mir ist, ist für mich sinnvoll. Es gibt schon ein paar Location-Based Services, zum Beispiel das

sehr bekannte Global Positioning System(GPS). Die Satelliten senden das Signal für Positionierung aus, und das GPS Gerät empfängt dieses Signal und berechnet damit die aktuelle Position. GPS bietet eine höhere Genauigkeit, aber es kann nur outdoor funktionieren, und der Preis dafür ist auch ziemlich teuer.

Ein Alternative ist die Technologie Global System for Mobile Telecommunications(GSM). Die ist der Standard für das heutige Mobiltelefon Netz. Ein auf GSM basierender Positionierungsmechanismus funktioniert gleich indoor und outdoor innerhalb eines gültigen mobilen Netzes. D.h. durch ein GSM Mobiltelefon können wir auch positionieren, aber ein Mobiltelefon allein kann das alles natürlich nicht schaffen. Die Aufgabe von Mobiltelefon ist nur das für Positionierung brauchbares Signal aus mobilem Netz zu empfangen, wir brauchen noch eine Software, um den Positionierungsalgorithmus auszuführen. Und durch die GSM Standard Schnittstelle ermöglicht es viele Mobiltelefon Besitzer diese Anwendung zu benutzen, es ist auch viel günstiger im Vergleich zum GPS.

Wenn wir unsere aktuelle Position ausgerechnet, was machen wir damit, wie bekommen wir die gewünschten Informationen, nämlich die ortsbezogenen Informationen? Wir können natürlich nicht alle Information und Daten auf dem mobilen Gerät speichern wegen seine sehr beschränkte Rechenleistung und Speicherkapazität. Das bedeutet, dass wir nur durch Daten Übertragung über Funknetz die ortsbezogene Information kriegen können. Das Projekt NEXUS wird für solche ortsbezogene Anwendungen entwickelt.

Im Rahmen des Projekts NEXUS entsteht eine Plattform für ortsbezogene Informationen, die ist offen für Anwendungen und Information Anbieter, ähnlich wie World Wide Web. Die Informationen werden aufgrund den Positionen in mehreren verteilten Server gespeichert, aber bei der Anwendungsseite kann man alle Daten einheitlich zugreifen, d.h. die Anwendungen haben eine integrierte Sicht über alle auf dieser Plattform gespeicherte Informationen.

Um dieses Ziel zu erreichen, ein wichtiges Konzept *Augmented World Model* wird hier verwendet. Die Basis von diesem Konzept ist, dass alle Objekte, Informationen in dieser physischen Welt werden durch virtuelle Objekte dargestellt, z.B. die Virtual Information Towers(VIT), und dann werden die alle einzelne Modelle in einer globalen *Augmented World* zusammengebaut. Jeder Anwender, der die NEXUS-Schnittstelle besitzt, kann in das NEXUS-System integriert werden, und die NEXUS-Daten benutzen, natürlich ist unser mobiles Gerät auch möglich, auf dieses System zu zugreifen.

Wenn wir die gewünschte ortsbezogene Informationen erhalten, natürlich wollen wir dies lesen. Es gibt mehrere Möglichkeiten um die Informationen anzuzeigen. Ein Notebook wird ideal, aber es ist nicht sehr bequem zu tragen und auch sehr kostbar. In meiner Arbeit wird eine PDA verwendet. PDA ist leicht und klein, sehr bequem zu tragen wie Mobiltelefon. Außerdem im Vergleich zum Mobiltelefon hat PDA ein viel größerer Bildschirm, und kann viel mehr Daten zeigen. Selbstverständlich ist die Beschränkung von PDA auch deutlich, aber trotzdem reicht PDA für viele Anwendungen, wir brauchen nicht unbedingt ein Notebook, um ein Menü vom Restaurant zu lesen.

Information auf PDA anzuzeigen, ist es gar nicht leicht wie es klingt. Wegen der geringen Rechenleistung und Speicherkapazität gibt es viele Einschränkungen für die Informationen, die auf PDA gezeigt werden. Wir dürfen nicht hoffen, dass PDA gleich wie Desktop, die hübschen HTML-Dateien auf dem Internet Browser darzustellen kann, deshalb müssen wir mal untersuchen, welches Datenformat geeignet für die mobilen Geräte anzuzeigen. Das wollen wir auch in dieser Arbeit noch diskutieren.

Hier gibt's einen Überblick für diese Arbeit. Die vorliegende Arbeit teilt sich in insgesamt 7 Teile ein. Im Kapitel 2 wird das ganze Positionierungssystem vorgestellt. Im Kapitel 3 gibt's eine kleine Einführung für das NEXUS-System. Kapitel 4 erzählt uns genau über die VITs und den Datenaustausch innerhalb dieses Systems. Im Kapitel 5 werden die Eigenschaften und Programmiersprachen für Palm OS dargestellt. Die genaue Implementierung für diese Arbeit wird im Kapitel 6 erklärt. Am Ende gibt's eine Zusammenfassung für diese Arbeit.

2 Das Positionierungssystem

In diesem Kapitel diskutieren wir das Positionierungsproblem. Bevor wir auf die ortsbezogenen Informationen zugreifen, müssen wir alle zuerst versuchen, um die aktuelle Position zu kriegen, dann können wir erst aufgrund dieser Position die gewünschten Informationen weiter suchen.

Es gibt schon ein paar Positionierungsmechanismen, und die meist häufig gebrauchte Technologie ist nämlich das GPS-System. GPS ist ganz einfach zu benutzen, im GPS-System empfängt das Endgerät die Positionierungsinformationen aus mehreren Satelliten, um die einzige aktuelle Position zu berechnen. Die Genauigkeit davon kann bis zu 15-20 Meter sein. Wenn das GPS Gerät berücksichtigt noch das Signal aus den statischen Antennen(DGPS), dann kann die Genauigkeit sich noch bis zu ein paar *cm* verbessern. Wir können die Positionsdaten direkt aus dem GPS-Gerät lesen, oder das GPS-Gerät leicht mit anderem Kommunikationsgerät verbinden, um jeweilige Anwendung zu entwickeln.

Aber die Nachteile von GPS sind auch deutlich. Zunächst funktioniert GPS nur outdoor, weil die Satelliten in der Sicht von Benutzer sein müssen, und das Wetter muss natürlich auch mitspielen. Um die GPS Funktion zu initialisieren, braucht es noch ein paar Minuten. Außerdem ist der Preis von GPS ziemlich hoch. Deshalb sollen wir ein Alternative von GPS suchen.

GSM ist eine gute Auswahl. Seit die mobile Kommunikation in unserem Leben eingesetzt geworden war, hat GSM eine sehr gute Infrastruktur mit höherer Dichte von Antennen angeboten. Diese Infrastruktur ist grade eine wichtige Bedingung für Positionierung. Aber das ursprüngliche Ziel von GSM ist natürlich nicht für Positionierung, sondern für mobile Kommunikation, ein Handy hat eigentlich nichts mit Positionierung zu tun, deswegen sollen wir noch ein paar Mechanismen entwickeln, um die Informationen aus GSM zu verwenden und die Position zu bestimmen.

Ein paar GSM basierende Techniken wurden schon entwickelt, die abhängig von der Benutzung von dem Wert sind, wie z.B. die Zeit für Signal-Übertragung, der Winkel von ankommender Signal, usw. Z.B. es gibt ein Technik heißt Enhanced observed Time Difference(E-OTD), die empfängt ein paar Signale aus den in der Nähe stehenden Base-Stationen und misst die Differenz zwischen verschiedene Zeit, die von jeweilige Signal für Übertragung gebraucht geworden sind. Diese Zeit-Differenz wird benutzt, um die aktuelle Position aufgrund den Base-Station-Positionen zu berechnen. Deshalb sind die Positionen von Base-Stationen notwendig und alle Signale aus verschiedene Base-Station müssen gleichzeitig ausgesendet werden.

Viele solche Techniken wie E-OTD werden nur bei dem mobilen Netz-Anbieter ausgeführt, und viele gebrauchte Werte ist auch nur gültig für den Netz-Anbieter. In meiner Arbeit werden nur die direkt aus GSM Handy erhaltene Werte benutzt. D.h. für mich ist nur die Feldstärke von empfangenen Signale gültig. Mit der Feldstärke können wir den Abstand zwischen den Empfangener und die umgebende Base-Station ausrechnen. Dabei gibt's ein physisches Gesetz zu folgen, d.h. wir müssen die

Abschwächung von Signal während der Übertragung und die jeweilige Topographie berücksichtigen. Außerdem sind noch die Positionen von den Base-Stationen innerhalb des Netzes notwendig für meinen Positionierungsmechanismus. Später wird das alles noch genau erklärt.

2.1 Global System für Mobile Telekommunikation

Bevor wir das GSM-System zum Positionierung verwenden, ist es notwendig, das Konzept von dem ganzen GSM-System mal genau erklären. Das ist von Bedeutung für die nachfolgenden Schritt, weil der Positionierungsmechanismus in dieser Arbeit auf dem GSM Netz basiert. Wir werden erfahren, wie wir durch Mobiltelefon telefonieren können.

2.1.1 Der Aufbau vom GSM-System

Das Mobilkommunikationsnetz mit vielen Antennen ermöglicht uns ohne Draht miteinander zu telefonieren. Wenn wir einen mobilen Anruf machen wollen, dann müssen wir in einem Bereich einer Base Transceiver Station(BTS) stehen. Und das von der Station ausgesendete Signal deckt ein bestimmtes Gebiet, das sogenannte Funkzelle ist. Innerhalb dieser Zelle können die beide BTSs und Mobiltelefon miteinander durch Signale kommunizieren.

Normale Weise hat jede Base-Station einen Sektor mit einem 120 Grade Winkel für Signal Sendung. D.h. auf einer Position kann es drei Base-Stationen geben. Die Strahlrichtung in dem Mittel von einem Sektor bietet das stärkste Signal, und bei der Richtung zwischen zwei Sektoren ist das Signal am schwächerstens. Auf Idealfall werden die Funkzellen und Sektoren in sechseckigen Formen eingerichtet, wie es in Abbildung 2-1 zeigt.

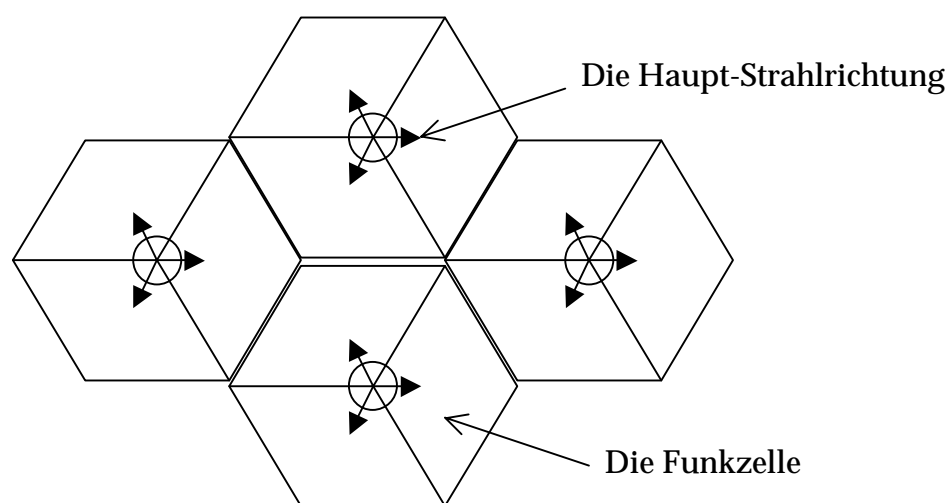


Abbildung 2-1: Sechseckige Einrichtung von Sektoren

Es kann offen passieren, dass der Handy Besitzer die aktuelle Zelle verlässt, und in die andere Zelle eintreten. Um weiter zu telefonieren ohne Unterbrechung, die Zellen sollen mit anderen benachbarten Zellen überlappende einrichten werden. Das Mobiltelefon überwacht die Signalstärke von benachbarten BTS, wenn eine benachbart BTS ein stärkeres Signal als die jetzige BTS sendet, dann wird die dienende BTS sofort gewechselt. So funktioniert unsere mobile Telekommunikation.

Wegen der speziellen sechseckigen Einrichtung von BTS soll unser Mobiltelefon sechs benachbarte BTS überwachen. Diese sechs BTS geben die dienende BTS grade um. Jede BTS hat einen Kanal heißt the Broadcast Control Channel(BCCH). Dieser Kanal sendet ein besonderes Signal für die Auswahl von dienende BTS aus alle überwachte benachbarte BTS. Und alle BTS sollen den BCCH mit gleicher Leistung ausstrahlen, um das Mobiltelefon die Feldstärke mehrere Signale zu vergleichen.

Jeder BCCH von jeweiliger BTS kann natürlich nicht gleich sein. Um diese zu unterscheiden, nutzt jeder BCCH eine eigne Frequenz. Unser Mobiltelefon kann von der bedienenden BTS noch eine Liste von alle benachbarte BTSs benutzenden Frequenzen(BCCH Allocation, BA) erhalten. Dadurch kann das Mobiltelefon alle benachbarte BTS in dieser Liste überwachen. Die BA wird von Netz-Anbieter gegeben.

Das Mobiltelefon benutzt zwei Parameter, um die Verbindungsqualität mit BTS zu bestimmen, nämlich die Feldstärke in Einheit dBm und die Signalqualität mit der Fehlerrate. Weil der Wert von Fehlerrate für unsere Positionierung keine Bedeutung hat, beobachten wir nur den Wert der Feldstärke, der in dem Bereich zwischen -110 dBm und -48 dBm liegt. Unser Mobiltelefon misst die Feldstärken von BCCH aus der bedienende BTS und den benachbarten BTS in der BA, und erzeugt eine Liste von sechs stärksten BTS. Das ist nämlich alles was wir von mobil Phone kriegen können. Die BA wird allen 30 Minuten mal dekodiert, und wenn die bedienende Funkzelle gewechselt wird, dann muss die aktuelle BA auch geändert werden.

2.1.2 Base Station

Das sogenannte Zelle-ID(CI) identifiziert eine Funkzelle und jeweilige BTS. Ein paar Zellen gruppieren zusammen zu einem Ortsgebiet, das von dem Location Area Code(LAC) identifiziert wird. Innerhalb einer Staat werden die mobilen Netze von dem Mobile Network Code(MNC) unterscheidet, und das Mobile Country Code(MCC) unterscheidet die Staaten. Dann mit oben genannten vier Code MCC, MNC, LAC und CI zusammen ist eine GSM-Station in der Welt absolut eindeutig.

Jeder Netz Anbieter wie E-Plus oder D1 hat einen eignen Frequenzbereich, z.B. von 890 MHz bis 960 MHz. Dieser Frequenzbereich verteilt sich noch in viele Kanäle für Datenübertragung, jeder Kanal bekommt eine Bandbreite von 200 KHz. Das Absolute Radio Frequency Channel Numbers(ARFCN oder FCN) identifiziert jeden Kanal.

Und jeder BTS wird eine Untermenge von den Kanälen für Kommunikation zugewiesen, welche den BCCH auch enthält. Wir haben gerade ernannt, dass jede

BTS einen BCCH hat, um sich zu identifizieren. Deswegen ist die Kanal-Nummer von BCCH die einzige wichtige Dinge, was ich brauche in meiner Arbeit. So bezieht die Kanal-Nummer sich immer auf den BCCH in folgenden Abschnitten.

2.2 Zellen-Datenbank

Es gibt viele Funkzellen innerhalb eines GSM-Systems, und jeder Zelle hat noch eine Menge Informationen, nämlich die Kanal-Nummer, die Zelle Position, das Code usw. Dann ist es wichtig, eine Zellen-Datenbank zu erstellen, um alle Informationen zu verwalten. Diese Datenbank ist ein wesentliches Teil in unserem Positionierungssystem. Weil wir nur die Informationen mit Bezug auf das Zelle-ID aus dem Mobiltelefon erhalten können, die aktuelle Position kann nur durch anderen Weg bekommt werden, d.h. wir müssen die Informationen in der Zellen-Datenbank verwenden, um die Informationen aus Handy in die aktuelle Position umzusetzen.

Alle Werte in einer Zellen-Datenbank können manuell erzeugt werden, die BTS Positionen, die Kanal-Nummer, die Zelle-ID usw. Wir können die Datenbank selbst machen, aber das ist natürlich langweilig und unnötig. Glücklicherweise haben wir noch einen anderen Weg, d.h. durch German cellular phone network provider Deutschen Telekom MobilNet GmbH(DeTeMobil) können wir einige nötige Daten direkt erhalten. Selbstverständlich haben die Netz-Anbieter die ganze Menge aktuelle Informationen der Zellen-Datenbank, und sogar in sehr hoch Genauigkeit, die wir manuell nie erreichen können, aber wegen des Geschäftsgeheimnisses ist diese Daten leider nicht offen. Wenn wir diese Arbeit zu einer praktischen Anwendung entwickeln möchten, können wir nur mit dem Netzanbieter kooperieren.

2.2.1 Beschreibung von der Zellen-Datenbank

Die Zellen-Datenbank besteht aus viele Datensätze, jeder Satz enthält mehrere Eigenschaften von einer einzelnen BTS und der entsprechenden Zelle. Die vier Werte MCC, MNC, LAC und CI zusammen spielen als den Schlüssel der Datenbank. Die Tabelle 2-1 zeigt die wichtigen Eigenschaften von BTS.

Mobile Country Code(MCC)	Eindeutige Staat Identifizierung
Mobile Network Code(MNC)	Eindeutige mobile Netz Identifizierung innerhalb einer Staat
Location Area Code(LAC)	Ortsgebiet Identifizierung
Cell ID(CI)	Eindeutige BTS oder Zelle-IDentifizierung innerhalb eines Ortsgebiets
Absolute radio Frequency Channel Number(ARFCN or FCN)	Die Kanal-Nummer stellt eine bestimmte Frequenz eines Kanals von einer BTS. In meiner Arbeit bezieht FCN sich immer auf das Broadcast Control Channel(BCCH)

Liste von der benachbarten BTS, jede BTS wird von MCC MNC LAC und CI identifiziert	Diese Liste zählt die überwachte benachbarte BTS von einer bestimmten BTS
Latitude	Latitude der Position von einer BTS mit Grad Einheit
Longtitude	Longtitude der Position von einer BTS mit Grad Einheit
Geodätische Daten	Dieser Wert stellt die Kodierung der Position, Latitude und Longtitude mit Format WGS84 dar.
Altitude	Altitude der Position von einer BTS oben die Meer Enene
Transmitter Power	Funksender Kraft der BTS mit Watt
Serving Range	Empirischer festgelegte maximal Bereich für Empfang einer bedienenden BTS. Der default Wert ist durchschnittlich 750 m
Neighbour Range	Empirischer festgelegte maximal Bereich für Empfang einer benachbarten BTS. Der default Wert ist durchschnittlich 1750 m
Beschreibung	Beschreibung von einer BTS Position
Gültigkeit seit	Seit wann ist dieser Satz gültig
Gültigkeit bis	Bis wann ist dieser Satz gültig

Tabelle 2-1: Die Eigenschaften von einer BTS

Hier sind nicht alle Werte nötig für unseren Positionierungsalgorithmus, einige Eigenschaften wie z.B. Beschreibung, Geodätische Daten sind nur für die Vollständigkeit von Zellen-Datenbank und vielleicht später für andere Anwendung brauchbar.

2.2.2 Verwendung von Zellen-Datenbank

Es gibt noch einige Aspekte zu berücksichtigen bei Verarbeitung mit Zellen-Datenbank. Bei Verlauf der Positionierung die falsche Informationen können durch die Zellen-Datenbank korrigiert werden, d.h. wir können z.B. die ungültige Messung erkennen mit Hilfe von Datenbank oder die Abbildung zwischen Kanal-Nummer und Zelle-ID erfahren.

Zunächst ist es notwendig, falsche Messung aus Handy zu erkennen, z.B. wegen des Messungsfehlers passt die Daten über den bedienenden Zelle einfach nicht zu den benachbarten Zellen, dann sind diese Daten ungültig und verzichtet werden soll. Um die ungültige Messung zu erkennen, gibt's zwei Möglichkeiten. Man kann die empfangene BA aus jetzigem bedienenden Zelle mit der in Datenbank gespeicherten BA vergleichen, oder einfach prüfen, ob die benachbarte Zellen in der eigenen BA vorkommt.

Für Positionierung ist die eindeutige Identifizierung von bedienende oder benachbarter BTS notwendig, d.h. das Zelle-ID wird gebraucht. Für bedienende BTS ist das kein Problem, weil das ID davon direkt aus Handy bekommt werden

kann[M35 2000]. Und die benachbarte BTSs können nur durch ihre Kanal-Nummer bemerkt werden. Dann kommt das Problem vor, um Kanal-Nummer zu Zelle-ID umzusetzen.

Man kann die Zellen-Datenbank mit der gegebenen Kanal-Nummer(FCN) durchsuchen, aber es kann mehrere Ergebnisse gefunden werden, weil die Kanal-Nummer nicht eindeutig in Datenbank ist. Nur eine Zelle-ID ist richtig, deshalb brauchen wir noch eine „Nachbar sein“ Relation in der Zellen-Datenbank einzusetzen, d.h. für jeden Zelle gibt's eine Liste von benachbarte BTS nach jeweiliger BA. Dann ist das Problem klar, statt Suchung nach der Kanal-Nummer in einer großen Datenbank fragen wir zuerst nach der Liste von benachbarten BTSs aus der bedienenden BTS. In dieser Liste normale Weise gibt's 6 bis 10 Zelle-IDs, ein davon muss die Kanal-Nummer passen, wenn die Datenbank vollständig ist.

Ein Beispiel: Ein Mobiltelefon hat erhalten ein Zelle-ID 02 für die bedienende BTS und zwei FCN 009 und 023 für die benachbarte BTSs. Jetzt möchten wir die Zelle-IDs für die beide benachbarte BTS ermitteln. Schritt 1, BTS 02 bekommt die Liste von benachbarte BTS aus der Datenbank, die BTS 05, 03, 07 sind. Aufgrund dieses Zelle-ID fragen wir wieder mal nach der jeweiligen Kanal-Nummer in Datenbank, und dann vergleichen die beide Kanal-Nummer aus Datenbank und aus Handy-Messung(Schritt 2, 3). Wenn die zueinander passen, dann ist das gewünschte Zelle-ID gefunden. Hier BTS 05 besitzt FCN 023 und BTS 03 besitzt FCN 009. Für n benachbarte BTSs sollen wir maximal $n+1$ Anfragen der Datenbank geben.

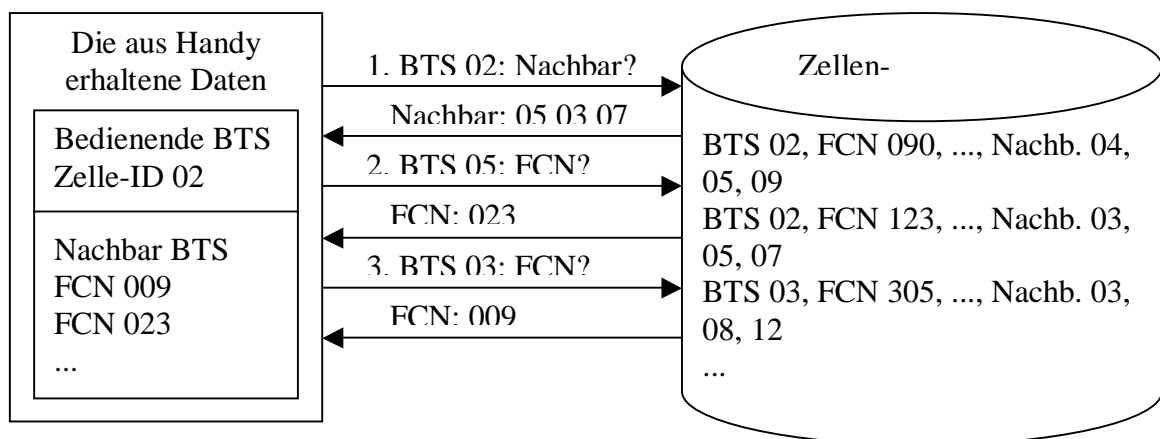


Abbildung 2-2: Umsetzung von FCN zu Zelle-ID

Die Kanal-Nummer werden regelmäßig wieder neu verteilt, d.h. jede BTS regelmäßig bekommt eine neue Kanal-Nummer. Das ist natürlich nicht angenehm für die Datenbank-Verwaltung, weil die Beziehung zwischen Zelle-ID und Kanal-Nummer muss oft erneuert werden. Aber das Zelle-ID bleibt normalerweise unveränderlich, die Relation „Nachbar sein“ ist auch so. Weil das Handy nur die Liste von Kanal-Nummer von der benachbarten BTS lesen kann, nach der Signalmessung sollen wir die Kanal-Nummer sofort in Zelle-ID umsetzen, wenn die Kanal-Nummer neu verteilt werden, dann ist die vorherige Messung auch ungültig.

Das ganze Positionierungssystem basiert auf einem gültigen GSM-Mobilnetz und seinem Anbieter. Solche Netz Anbieter in Deutschland sind D1, D2, E-plus und Viag Interkom(O₂). Mit der Kooperation mit DeTeMobil hat D1 eine gute Deckung und vollständige Daten für Zellen-Datenbank, so in meiner Arbeit wird D1 ausgewählt.

2.3 Berechnung von der Position eines Handy-Benutzers

Mit der Daten aus dem Mobiltelefon und der Zellen-Datenbank können wir endlich die aktuelle Position berechnen. Es gibt mehre Algorithmus für Positionierung. Wegen der eingeschränkten Menge von gültigen Werten aus Handy wird es angenommen, dass die berechnete Position nicht so präzise wie die wirkliche Position ist, d.h. für jede ausgerechnete Position gibt's auch einen Error-Bereich, die wirkliche Position soll sich in diesem Bereich befinden. Das einfachste Beispiel ist ein Kreis um die ausgerechnete Position, und der Radius dieses Kreises soll lange genug sein, um die wirkliche Position zu decken.

Bevor wir den Positionierungsalgorithmus entwickeln, gibt's noch ein paar notwendige Aspekte zu erklären, nämlich die Filtrierung für Daten, das Koordinationssystem und die Berechnung des Abstandes.

2.3.1 Filtrierung der Input-Daten

Der Positionierungsalgorithmus erhält die aktuellen gültigen Messungsdaten unveränderlich direkt aus Handy. Aber manchmal ist es auch sinnvoll, vor der Berechnung eine Aufbereitung für die Daten zu machen. Der Algorithmus selbst kann natürlich nicht unterscheiden, ob die Daten aufbereitet wurden oder nicht. Diese Aufbereitung ist nämlich Filtrierung von empfangenen BTSs. Die muss am Anfang des Algorithmus ausgeführt werden. Abbildung 2-3 zeigt der Verlauf des ganzen Algorithmus.

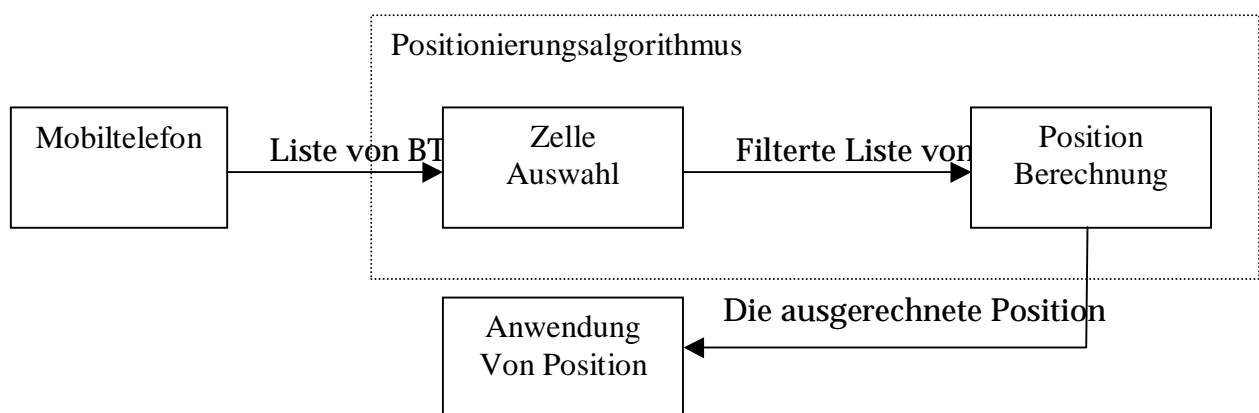


Abbildung 2-3: Der Verlauf des Positionierungsalgorithmus

Die gemessenen Daten aus Handy besteht aus Zelle-ID und die Kanal-Nummer von der bedienende BTS und den benachbarten BTS. Die Umsetzung von Kanal-Nummer

zu Zelle-ID wird vor dem Positionierungsalgorithmus ausgeführt. Außerdem werden noch ein paar zusätzliche erforderliche Daten aus Zellen-Datenbank gesucht. Wir nehmen an, dass die Menge von Input Daten für den Algorithmus gültig sind, d.h. ungültige Daten wegen Zelle-Wechselung oder falsche Messung schon ausgeschmissen sind.

Wie es in Abbildung 2-3 gezeigt, das Algorithmus bekommt eine Liste von BTS aus Mobiltelefon, nämlich das Zelle-ID von bedienender BTS, die Kanal-Nummer von benachbarten BTS, und jeweilige Feldstärke. In dieser Liste gibt's normale Weise sieben Einträge, aber tatsächlich sind nicht alle nützlich, d.h. die gemessene Feldstärke muss in einem bestimmten Bereich beleiben.

Für diesen Zweck gibt's einige Zelle-Auswahl-Module, die eine gegebene Menge von Zellen unter bestimmten Bedienungen zu einer Untermenge filtern. Diese Module können einfach miteinander verbinden, d.h. der Output eines Moduls ist ein Input eines anderen Moduls. In jedem Modul sind einige Bedienungen zu unterscheiden. Hier ist ein Überblick für die Module.

Haupt Sektor Auswahl	Wird die Daten aus verschiedenen sich auf der gleichen Position befindenden Sektoren empfangen, wird nur die BTS mit stärkster Feldstärke so angenommen, dass seiner Sektor gegenüber den Handy Besitzer steht. Es wird angenommen, dass zwei BTS sich auf der gleichen Position befinden, wenn der Abstand dazwischen ihre Positionen aus Zellen-Datenbank kleiner als 90 Meter ist. Wenn zwei BTS sich mit gleichen Feldstärken auf gleicher Position befinden, dann wählen wir die mit niedrigem Zelle-ID.
Bereich Auswahl	BTS mit Feldstärke höher als $1.57e-8$ Watt(ungefähr -48,04 dBm)wird gestrichen. Solche BTS hat starke Empfangsmöglichkeit, trotzdem ist es nötig, um zu streichen, weil dieser Wert die Berechnung verfälschen kann.
Beste zwei Auswahl	Die zwei BTS mit stärksten Feldstärken werden ausgewählt. Die andere BTS werden gestrichen. Es ist nützlich, um die Schwankung zu vermeiden, weil starke BTS selten abweicht.
Beste drei Auswahl	Die drei BTS mit stärksten Feldstärken werden ausgewählt. Die andere BTS werden gestrichen.

Tabelle 2-2: Überblick für Zelle Auswahl Module

Wie oben genannt, durch die Benutzungen von Filtern hintereinander können wir eine Filterkombination machen. Z.B. wir möchten die drei stärksten BTSs ausfiltern, deren Feldstärke kleiner als -48 dBm sind, und sich auf unterschiedlichen Positionen befinden, dann sollen wir die drei Module Beste drei Auswahl, Haupt Sektor Auswahl und Bereich Auswahl verwenden. Die Reihenfolge von Filtern ist auch wichtig, weil sie nicht austauschbar sind:

- Erstens, Bereich Auswahl stricht alle BTS mit -48 dBm Feldstärke.
- Zweitens, Haupt Sektor Auswahl stricht die schwache Sektoren auf der gleichen Position
- Drittens, die drei stärkste BTS werden durch Beste Drei Auswahl ausgewählt.

2.3.2 Koordinatensystem

Die Koordinaten werden fast überall innerhalb dieser Berechnung benutzt, die kommen beim Input aus Handy und Out aus Berechnung vor. Normale Weise bestehen die Koordinaten aus drei Werte: Longitude, Latitude und Altitude. Für Zwei-Dimension Berechnung ist der Wert von Altitude gleich Null. Longitude und Latitude bilden das horizontale Teil von Position und Altitude bildet das senkrechte Teil von Position. Die Input- und Output-Werte werden übereinstimmend durch das bekannte Koordinatensystem WGS84 dargestellt. D.h. die Einheit von Longitude und Latitude ist Grad, von Altitude ist Meter. Es gibt's noch einpaar unterschiedlichen Formaten. In Tabelle 2-3 stellen mehr Format die gleiche Position dar. Der Unterschied ist die Darstellung von dem Teil nach dem Punkt. Vor dem Punkt sind die Werte immer Grad. Wir benutzen das letzte Format, weil das entsprechend für die Berechnung ist.

Minuten und Sekunden in Grad Format	48°46'3''	9°14'58''
Minuten in Grad, Sekunden in dezimales Format	48.46051	9.14981
GPS Daten Format	4846.051,N	00914.981,E
Minuten und Sekunden in dezimales Format	48.76752	9.24968

Tabelle 2-3: Format von Koordinaten

Für die Berechnung selbst ist ein Koordinatensystem in Grad unpraktisch, weil der Abstand ist nicht einfach mit Grad-Einheit zu berechnen. Deshalb sollen wir zunächst die Position in ein Flach-Koordinatensystem umsetzen. Z.B. benutzen wir den Punkt in oberer Tabelle(longitude_m, latitude_m, altitude_m) = (9.24968, 48.76752, 0.0) als die Zentrale von dem Test-Gebiet Stuttgart-Wangen, können wir die Werte von andere Positionen verkleinern, d.h. im Vergleich zum diesen Punkt können die Positionen durch x und y Werte dargestellt werden, x Wert zeigt den Abstand von diesem Punkt zu einer Ost-liegenden Position, y Wert zeigt den Abstand zur West-liegenden Position. Die zwei andere Richtungen werden durch negative Werte dargestellt. Altitude ist z Wert. Jetzt sind alle Werte schon mit Einheit Meter.

Für die Koordinatenumsetzung muss es berücksichtigt werden, dass die Länge einer Einheit von dem Meridian abhängig von Latitzde-Wert ist, d.h. die Länge von dem Meridian ist immer kleiner als Latitude immer größer ist. Für Latitude ist die Länge einer Einheit immer gleich wie die Länge von 1/360 des Erdumfangs. Der Radius der Erde ist ungefähr 6370 km.

$$longitude_m = [(2\pi r_{Erd}) / 360] \cdot longitude_{Grad} \cdot \cos(latitude_{Grad}) - longitude_{Zentrale, m}$$

$$latitude_m = [(2\pi r_{Erd}) / 360] \cdot latitude_{Grad} - latitude_{Zentrale, m}$$

Die Umsetzung von Kugel ins Flach funktioniert nur in einem ziemlich kleinen Gebiet, d.h. das Error darf nicht zu groß, aber für ein paar Kilometer deckendes Gebiet bleibt das Error schon klein genug, so kann der Abstand einfach wie obige Formel berechnet werden. Bestens soll diese Abstand-Berechnung auch in ein steckbar Modul einzusetzen. Dann ist es einfach mit anderer Abstand-Berechnung zu wechseln, wenn es nötig ist.

2.3.3 Zelle-basierendes Algorithmus

Während der Entwicklung vom Berechnungsalgorithmus ist es immer wichtig die Genauigkeit des Algorithmus zu berücksichtigen, weil das Hauptziel von dem ganzen Positionierungssystem ist um eine richtige Position auszurechnen. Außerdem gibt's noch einige Anforderungen für den Algorithmus zu beachten, nämlich:

- Klein Aufwand für Berechnung
- Wenige zusätzliche Daten aus Zellen-Datenbank

Aufgrund der existierenden Struktur von GSM und das physische Verhalten von Signalen aus GSM gibt's schon einpaar Berechnungsalgorithmus für Positionierung. Auf Zellen basierender Algorithmus ist nämlich ein Algorithmus auf dem Standard GSM Zellen-Struktur. Weil ein GSM-Handy immer eine Verbindung zu einem GSM Zelle mit stärkster Feldstärke hat, können wir annehmen, dass das Zelle das nächste Zelle ist, und das ganze Zelle oder mit allen benachbarten Zellen für das Error Gebiet halten. Das Ergebnis davon ist nicht eine Position, sondern nur ein Gebiet. Aber wir können trotzdem versuchen, um das Error Gebiet zu verkleinern.

Das Zentrale eines Zelle-Kreises und die Richtung eines Sektors sind die Eigenschaften von BTS und einfach festzustellen, aber der Radius davon ist ein empirischer Wert. Z.B. der maximale Radius eines Zells ist der Abstand von BTS zu der Position, worauf die Feldstärke noch empfangen werden kann. Aber normale Weise ist der aktuelle Radius viel kleiner, weil wenn die Feldstärke aus anderer BTS stärker als die bedienende BTS ist, dann wird die Feldstärke von der aktuellen BTS nicht mehr empfangen. Wenn die Feldstärke als benachbartes BTS-Signal empfangen werden soll, dann wird der Radius ziemlich groß wegen der Zellen-Struktur von GSM.

Es gibt zwei auf Zellen basierende Implementierungen. Eine heißt Zelle Algorithmus, das den einfachsten Fall realisiert. D.h. das Error Gebiet ist ein Kreis rund um die Position der Bedienenden BTS. Die Stelle und Radius von dem Kreis werden aus Zellen-Datenbank geliefert. Wird kein Wert davon angeboten, dann wird der durchschnittliche Wert von 750 m benutzt. Die Stelle von BTS ist nämlich die ausgerechnete Position.

Das andere heißt Zelle-Sektor Algorithmus, d.h. neben dem Zelle Radius sind die Ecke des Sektors und die Orientierung davon auch notwendig. Die Orientierung ist nämlich die Haupt Strahl Richtung der BTS, die den Sektor halbiert. Bei der Ecke des Sektors gibt's normale Weise drei Möglichkeiten aufgrund DeTeMobil: 65°, 90° und 360°. Für 360° Sektor ist es offensichtlich ein Zelle-Algorithmus. Bei anderen zweien Sektorecken scheint es sich vier bis sechs BTSS auf einer Position aufzufinden. Aber es gibt niemals auf einer Position mehr als drei BTSS, weil die wirkliche Deckung größer ist. Deshalb etwa 1/3 Kreis-Ecke ist eine gute Annäherung. Außerdem gibt's noch einen Effekt zu beobachten. Die Feldstärke einer BTS ist auch ein bisschen hinter der BTS empfangbar, d.h. außer der Sektorecke in der umgekehrten Richtung von der Haupt Strahl Richtung. Dann müssen wir die Form des Sektors wie in Abbildung 2-4 verändern, d.h. ein Kreis mit kurzem Radius ergänzt den existierenden Sektor auf den übrigen Graden.

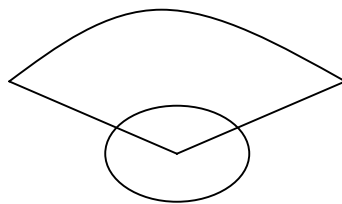


Abbildung 2-4: Die Form eines Sektors mit Empfang hinter der BTS

2.3.4 Triangulation Algorithmus

Triangulation ist ein Algorithmus, das von GPS verwendet wird, um den mobilen Benutzer zu positionieren. Der Durchschnitt von mindestens drei Kreise ergibt sich die Position. Die Radien den Kreisen werden durch die gemessene Signal-Feldstärke berechnet. Die Feldstärke können wir aus Mobiltelefon direkt erhalten, dann setzen wir die Feldstärke in Abstand(nämlich der Radius) um. Es gibt mehr Möglichkeiten um die Umsetzung zu realisieren. Das ist nicht so einfach und braucht viele Erfahrungen.

Der Abstand Aufbereitung ist Vorbedingung für den Triangulation-Algorithmus, die direkt die Genauigkeit der Positionierung beeinflusst. Die Feldstärke verändert sich mit dem Abstand, nämlich die Feldstärke vermindert sich von BTS bis Signalempfänger. Die Rate der Verminderung ist abhängig von einigen Bedingungen. D.h. die Verhaltung von der Übertragung der Feldstärke wird von einigen Faktoren beeinflusst.

Auf idealen Fall, die Feldstärke unter Free Space Bedingung vermindert sich mit Quadrat von Abstand r^2 , wenn die mit dem Abstand r von BTS nach außen überträgt wird. Die ausstrahlende Energie verteilt sich über eine Kreis-Oberfläche, die mit Quadrat von dem Radius wächst. Das ist sogenannte Free Space Abschwächung.

Aber solcher ideale Fall passiert fast nie. D.h. die Signale aus BTS werden besonders im bewohnten Gebiet durch Menschen und die natürliche Struktur beeinflusst. Z.B. große Objekte blockieren die Signal-Übertragung; Signal wird wiedergespiegelt,

wenn die Größe vom Objekt viel größer als die Wellenlänge des Signals (etwa 33 cm) ist, und das Signal spaltet sich in verschiedenen Richtungen, wenn die Größe vom Objekt gleich oder kleiner als die Wellenlänge des Signals ist; das Signal wird auch abgelenkt durch die Schwankung von der Luftdichte, usw.

Solche Effekte machen die Umsetzungen von Feldstärken von unterschiedlichen Orten in Abstand nicht gleich und schwer. Zuerst die Topographie verursacht immer die oben genannten Effekte, besonders sind die Größe und die Anzahl von Gebäuden wichtige Aspekte, z.B. hinter einem Berg ist das Signal immer schwach. Deshalb gibt's natürlich einen Unterschied zwischen der Stadt und dem Stadtrand.

Außerdem beeinflusst die Höhe von der Antenne der BTS den Übertragungsbereich, und es muss auch beobachtet werden, ob die Antenne über oder unter dem Rooftop liegt. Die empfangene Feldstärke ist auch abhängig von der Höhe des Benutzers, einerseits verbessert die Erkennung der Höhe des Benutzers die Berechnung vom Abstand zur BTS, andererseits soll die Höhe des Benutzers auch berechnet werden. So müssen wir entweder einen gemischten Abstand mit der Höhe benutzen, oder die Höhe einfach ignorieren. Schließlich gibt's noch einen großen Unterschied dazwischen, ob man indoor oder outdoor steht, weil das Signal ziemlich schwach wegen der starken Abschwächung durch die Wände und Fenster ist.

Wegen obigen Ursachen verwende ich in meiner Arbeit das Path Loss Model [Seybold 2001], um das Verhalten von Signal-Übertragung gut darzustellen. Dieses Übertragungsmodell basiert auf der folgenden Formel. Die Variable a stellt den Betrag von der BTS Energie dar; die Variable n stellt die Abschwächung dar.

$$\begin{aligned} \text{Feldstärke} &= a / (\text{Abstand})^n \\ \text{Log}(\text{Feldstärke}) &= \text{log}(a) - n \text{log}(\text{Abstand}) \end{aligned}$$

Während der Positionierungsphase sind diese Formeln für Abstand-Berechnung mit Feldstärke zuständig. Hier sind die Werte von a und n mit Erfahrung von Signal-Übertragung in unterschiedlichen Radio-Umwelt auszuwählen, d.h. die Topographie und Stadtstruktur entscheiden diese Werte. Dazu braucht es empirische Anpassung für das Modell in einem bestimmten Gebiet, wir müssen viel Mühe geben, um die beide Werte durch gemessene Position und Feldstärke herzustellen. Weil in meiner Arbeit es nur in der normalen Stadt getestet wird, dann habe ich das Ergebnis aus [Seybold 2001] einfach genommen:

$$\begin{aligned} a &= -6.63162 \\ n &= 1.36955 \end{aligned}$$

Die beide Werte gelten nur in der Stadt wie Stuttgart/Vaihingen. Durch die obige Formel mit diesen beiden Werten können alle Feldstärken in Abstand zur BTS umgesetzt werden.

2D Triangulation

Nach der Bestimmung von den Abständen zu der BTS, können wir den Triangulation-Algorithmus endlich verwenden. Wir untersuchen zuerst den 2D

Triangulation-Algorithmus, d.h. die Höhe-Werte werden ignoriert. Der Abstand bildet einen Kreis rund um die jeweilige empfangene BTS. Auf den idealen Fall liegt die Position irgendwo auf dem Kreis.

Am Anfang von dem Algorithmus soll die Anzahl von empfangenen BTS durch Zelle Auswahl Modulen gefiltert werden, das haben wir vorher schon gesagt. Dann mit dem obigen Übertragungsmodell Path Loss Propagation Model berechnen wir die Abstände zu allen BTSs, die wir mit Vorbedingungen ausgewählt haben.

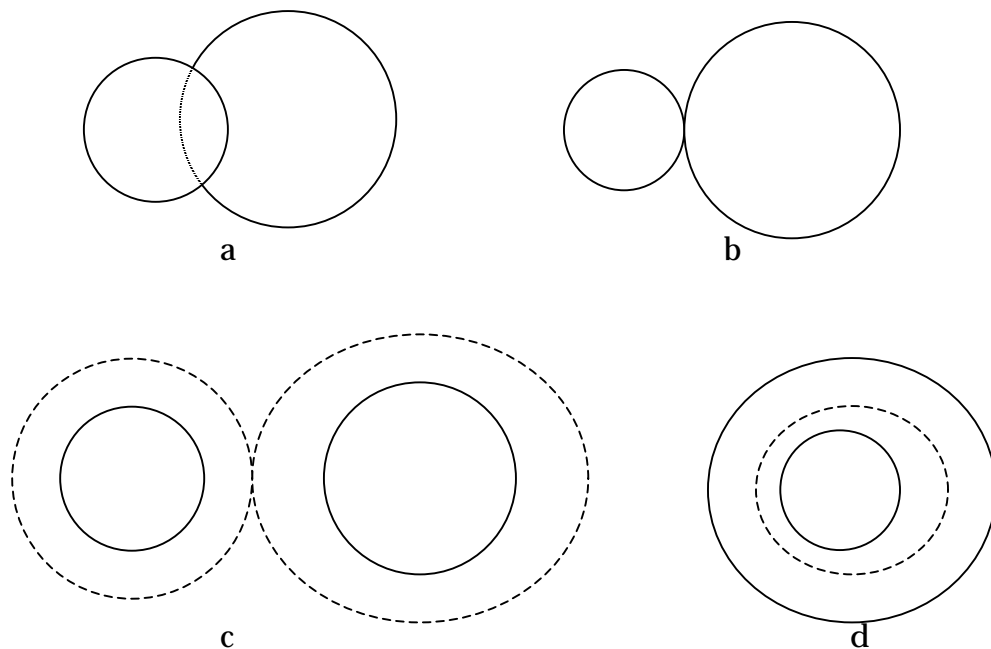


Abbildung 2-5: Vier Möglichkeiten für Kreuzung von Kreisen

Die folgende Verarbeitung verteilt sich in zweien Schritten. Zunächst, die Kreuzungen von den Kreisen erzeugen mehre Punkte, dann wählen wir die wichtigen Punkte aus, worauf unsere Berechnung basiert. Auf den idealen Fall, alle Kreise sollen sich in einem Punkt kreuzen, der nämlich die Position ist. Aber das passiert fast nie, d.h. jedes Paar von Kreisen kreuzen sich auf keinen, einen oder zwei Punkte wie in Abbildung 2-5.

Wenn es keinen Punkt von Kreuzung gibt, d.h. der Abstand zwischen zwei BTSs ist länger als die Summe von beiden Radien(die ausgerechnete Abstände zu jeweiligen BTS) rund um die jeweilige BTS(Abbildung 2-5 c). Das Ergebnis aus Abstand-Berechnungen ist offensichtlich nicht entsprechend für Positionierung. Deshalb sollen wir die beide Radien korrigieren, nämlich die beide Radien vergrößern bis sie sich miteinander treffen. Wir nehmen diesen Treffpunkt als den Kreuzungspunkt. Gleichfalls, die beide Radien können auch so korrigiert werden, dass ein sich vergrößert und der andere sich verkleinert(Abbildung 2-5 d), auch bis sie sich miteinander treffen(Abbildung 2-5 d). Bei weiteren Kreuzungen werden wieder die ursprünglichen Radien benutzt.

Jetzt haben wir für jedes Paar von Kreisen einen oder zwei Kreuzungspunkte. Für den Fall von zweien Kreuzungspunkten gibt's nur ein Punkt, der sich zu der Position nähert, und den anderen Punkt können wir streichen. D.h. wir wählen nur die sinnvollen Punkte aus. Aber das Problem ist, woher wissen wir, welcher Punkt richtig ist, die aktuelle Position ist unbekannt.

Haben wir schon einen Mittelwert aus ein paar frühe ausgewählten Punkten, dann ist es einfach zu entscheiden, welchen Punkt kann ausgewählt werden, d.h. der Punkt mit kürzerem Abstand zu dem vorherigen Mittelwert wird ausgewählt und verwendet, um den Mittelwert weiter zu verfeinern.

Jetzt ist das Problem, wie erhalten wir den Mittelwert am Anfang. Wenn bei der ersten oder zweiten Kreuzung gibt's schon einzelnen Kreuzungspunkt, dann dieser Punkt wird als den Mittelwert genommen. Wenn bei ersten und zweiten Kreuzung insgesamt vier Kreuzungspunkte erzeugt werden, dann nehmen wir die zwei Punkte mit kürzestem Abstand aus, um den Mittelwert zu bestimmen.

Nach allen Kreuzungen von allen Kreispaaren ist der letzte ausgerechnet Mittelwert nämlich das Ergebnis von Algorithmus. Die Kreuzung von Kreisen und die auf dem Mittelwert basierende Auswahl für sinnvolle Punkte werden fortgesetzt ausgeführt, die Komplexität davon ist abhängig von der Anzahl von Kreuzungen. Es gibt noch etwas zu erwähnen: wenn es nur einen Kreis für Positionierung gibt, dann ist die Stelle jeweiliger BTS das Ergebnis; wenn es gerade zwei Kreis mit zweien Kreuzungspunkten gibt, dann ist der Mittelwert dazwischen das Ergebnis. Die genaue Beschreibung steht auch in [Seybold 2001].

3D Triangulation

Das Drei Dimension Algorithmus funktioniert ähnlich wie 2D Algorithmus, aber noch kompliziert. Zuerst die durch den Abstand gebildete Form ist nicht Kreis mehr, sondern Kugel. Die Position wird durch den Durchschnitt von Kugeln erzeugt. Und natürlich muss der Wert von Höhe auch berücksichtigt werden.

Der Durchschnitt von zweien Kugeln erzeugt ein Kreis, der auf einer Ebene zwischen beide BTS liegt. Schneiden zwei Kugeln sich nicht miteinander, wird die Regel(Korrigierung für Radien) in 2D erwähnt verwendet, um einen Treffpunkt zu erhalten. Auf idealen Fall scheiden alle Kugeln sich miteinander nur in einem Punkt.

Und jetzt müssen die Kreise aus Kugel Durchschnitt sich wieder miteinander schneiden, um Punkte zu erzeugen. Aber solche Kreise schneiden sich manchmal nicht, weil die auf unterschiedlichen Ebenen in dem Raum liegen. Dann nehmen wir hier eine Methode, d.h. wir schneiden die Ebenen, worauf die Kreise liegen, statt den Kreisen sich miteinander in Linien. So schneidet eine Linie entweder den Kreis mit zwei Punkte, d.h. der Abstand von der Zentrale des Kreises zu der Linie kürzer als den Radius des Kreises, oder der Punkt auf dem Kreis mit kürzestem Abstand zu der Linie wird genommen. Das wird für beide Kreise getan. Dann können wir wieder für jeden Kreis einen oder zwei Punkte haben. Aus solchen Punkten berechnen wir

vorgesetzt den Mittelwert gleich wie in 2D, und endlich ist der letzte Mittelwert das Ergebnis von 3D Algorithmus.

Besonderes wenn es nur eine Kugel gibt für Positionierung, dann ist die Stelle der BTS das Ergebnis, wenn es gerade nur zwei Kugeln mit einem Durchschnittskreis gibt, dann ist die Zentrale dieses Kreises das Ergebnis.

2.4 Zusammenfassung

Dieses Positionierungssystem ist die Vorbedingung für den Zugriff auf ortsbezogene Informationen. Dieses System ist abhängig von der GSM, d.h. wir müssen durch ein Mobiltelefon die Zelle Informationen aus dem Funknetzwerk empfangen, dann können wir erst die Position berechnen. Dazu gibt's noch eine Zellen-Datenbank, um die zusätzliche Informationen wie z.B. die genaue Stellen von den BTSs anzubieten. Die Berechnung verwendet den 2D Triangulationsalgorithmus, und das Ergebnis wird durch das Koordinatensystem WGS84 dargestellt.

3 Einführung in das NEXUS-System

Das Hauptziel meiner Arbeit ist, die gewünschte Informationen von der Umgebung des Mobiltelefons auf dem Bildschirm vom PDA anzuzeigen. Dazu braucht man zunächst ein Positionierungssystem, um die aktuelle Position des Benutzers festzustellen. Das haben wir in letztem Kapitel gerade diskutiert. Und aufgrund der Positionsinformation dann können wir die Ortsbezogene Informationen irgendwo suchen und abholen. Aber wie macht man das? Zur Zeit gibt's schon ein paar Informationssysteme, welche die räumliche Daten des Benutzers und des Objekts oder die Organisation von Informationen durch räumliche Methoden berücksichtigen, z.B. das Auto-Navigationssystem, das Geographical Informationssystem und Tourist-Guide usw., aber solche Informationssysteme werden nur in isolierten bestimmten Bereichen verwendet, und haben starke Einschränkungen, durch das Auto-Navigationssystem erfahren wir nur den Name einer Straße, worauf wir stehen, aber sonst noch weitere Informationen über diese Straße wie z.B. die Bushaltstelle auf der Straße können wir nicht erhalten. Was wir brauchen ist ein System, das die Ortsbewusste Anwendungen unterstützt und ganz offen für fast alle Informationen zu Verfügung stellt.

Das Untersuchungsteam von dem Projekt NEXUS entwickelt eine offene Plattform für ortsbewusste Anwendungen für mobile Benutzer. Alle isolierte Anwendungen sollen in diese Plattform integriert werden, so können verschiedene Anwendungen gemeinsame oder spezifische Daten benutzen, und sogar miteinander kommunizieren oder kooperieren. Die NEXUS-Plattform wird als eine Middleware vorgestellt, um die Anbieter und die Benutzer der ortsbewussten Daten zusammenzubringen. Jetzt wollen wir das ganze System mal vorstellen.

3.1 Das Ziel des NEXUS-Systems

Das Hauptziel des NEXUS-Systems ist, um eine Infrastruktur für ortsbewusste Anwendungen anzubieten. Dazu brauchen wir ein World Model, um alle Objekte darzustellen, die in der physischen Welt existieren und relevant für die ortsbewusste Anwendungen sind. Das Model soll noch mit weiteren nötigen Informationen verbunden werden, d.h. solches Model wird durch zusätzliche virtuelle Objekte, welche die verbundenen Informationen darstellt, vergrößert. Solche Model nennen wir Augmented World Model, das die Erweiterbarkeit und Flexibilität für NEXUS mitbringt und das Interface zur Anwendung bildet. Durch das sogenannte Augmented World Model kann die NEXUS-Plattform von vielen unterschiedlichen auf NEXUS basierenden Anwendungen verwendet werden, manche Anwendungen können sogar in der Zukunft vorkommen.

Die NEXUS-Plattform ist nicht nur eine offene Plattform, sondern auch eine globale Plattform. Alle Leute könnten Informationen und Dienste auf dieser Plattform anbieten, ähnlich wie die sehr erfolgreiche WWW-Plattform. Jeder Dienstanbieter mit einem einheitlichen NEXUS-Interface kann das einfach machen, und durch die sogenannte Föderation wird das gleichzeitig vorkommende heterogene Problem

gelöst. Natürlich soll die NEXUS-Plattform in der Lage sein, eine große Menge Benutzer zu bedienen, und eine große Anzahl von Informationen zu speichern.

Außerdem weil NEXUS sich mit mobilen Benutzern beschäftigen muss, dann sind die absichtlichen Anwendungsumgebungen für NEXUS die kleine mobile Geräte. Deshalb muss NEXUS noch die mobile Kommunikation und die kleine mobile Geräte wie z.B. PDA unterstützen.

3.2 Die NEXUS-Architektur

Das NEXUS-System ist ein verteiltes System, d.h. verteilte Dienste werden benutzt, um die Informationen zu speichern. Die Architektur davon besteht aus drei Ebenen wie in Abbildung 3-1 gezeigt, nämlich die NEXUS Anwendungsebene, die Föderationsebene und die NEXUS Dienstebene.

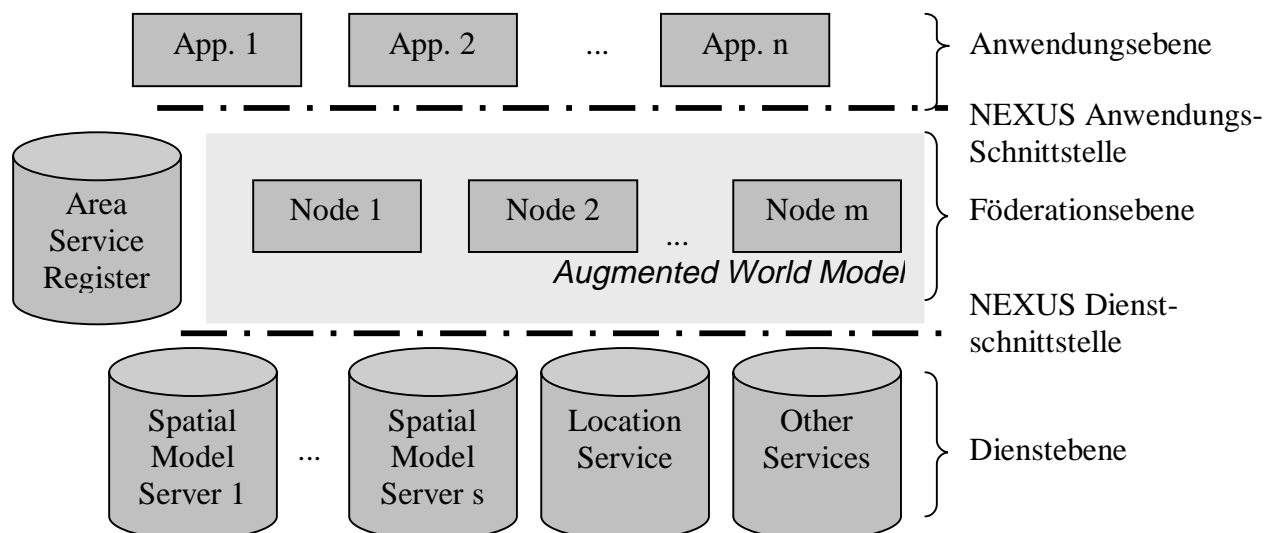


Abbildung 3-1: Übersicht der NEXUS-Architektur

Es ist ähnlich wie die WWW-Architektur, nur eine weitere Föderationsebene wird in die Architektur eingesetzt. In der Anwendungsebene befinden sich die sämtlichen unterschiedlichen ortsbezogenen Anwendungen, die durch die globale Föderation mit anderen Komponenten des NEXUS-Systems kommunizieren. Die Anwendungen können entweder auf einem NEXUS-Server oder einem NEXUS-Client ausgeführt werden. Ein NEXUS-Client, der statisch oder tragbar ist, nutzt die Anwendung, um die NEXUS-Dienste zu erreichen. Weil die Hauptabsicht von NEXUS gerade die Unterstützung für die mobilen Benutzer ist, ist es besonders bedeutend, dass alle wichtige Anwendungen auf den tragbaren Client (Notebooks, PDA usw.) ausführbar sind.

Die Föderationsebene liegt auf den unterliegenden Daten und Informationen Anbietern und besteht aus einem Area service Register und mehreren NEXUS-Knoten. Die Föderation bietet den Anwendungen eine einheitliche Methode, um auf

die Informationen zu zugreifen. Die Anwendungen brauchen nur mit dem nächsten Knoten Kontakt zu machen, und der Föderationsknoten analysiert die ankommende Anfragen und verteilt die Anfragen zu den unterschiedlichen relevanten Diensteanbieter weiter. Die Lösungen aus den NEXUS-Diensten werden von den Knoten integriert und in eine für die Anwendungen lesbare Form umgewandelt. Jeder Knoten realisiert die NEXUS-Föderation, und die Anwendungen brauchen keine Kenntnis über die Verteilung und Aufbau den Diensteanbietern.

Auf der unteren Ebene der Architektur liegen verschiedene Diensteanbieter. Die wichtigsten NEXUS-Dienste sind der Spatial Model Service, der Location Service und der Event Service. Der Spatial Model Service wird aus allen Spatial Model Server zusammgebaut, indem die statischen Objekte wie Gebäude, Straßen oder Virtual Information Towers(VIT) gespeichert werden können. Die Spatial Model Server werden in dem Area Service Register registriert, das die geographischen Gebiete mit den Spatial Model Server zusammenzuhängen lässt.

3.3 Augmented Area

Die NEXUS-Plattform bietet eine Infrastruktur für die ortsbezogenen Anwendungen. Diese Infrastruktur basiert auf den Computer Darstellungen von bestimmten Regionen in der physischen Welt. Solche Regionen, die sich miteinander überlappen und sogar enthalten dürfen, werden von den virtuellen Objekten angereichert und als Augemented Area bezeichnet. Eine Augemented Area ist räumlich begrenzt und repräsentiert eine geographische Region wie z.B. eine Stadt, eine Straße. Es besteht aus virtuelle und physische Objekte, dass die virtuellen Objekte nur durch die NEXUS-Plattform erhalten werden können und die physischen Objekte in der physischen Welt existieren. Das Augmented Area Model ist nämlich die Computer Darstellung mit einheitlichem Format für die Augmented Area(sehe Abbildung 3-2).

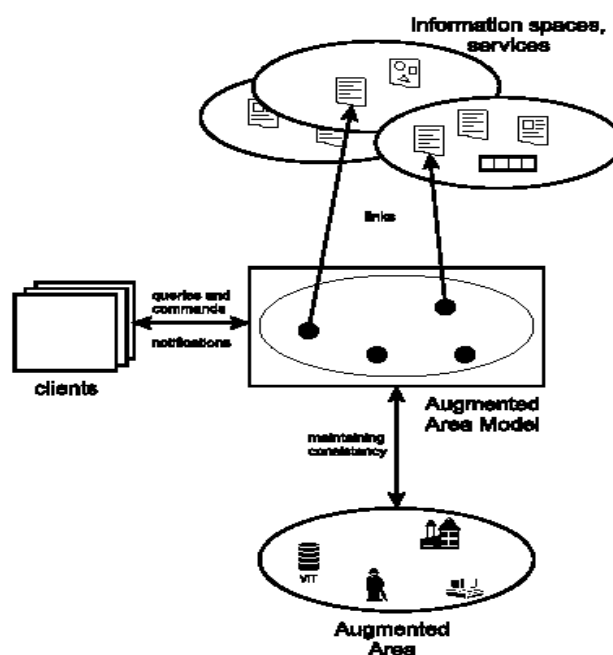


Abbildung 3-2: Augmented Area Model

Eine Augmented Area enthält Objekte, die auch auf bestimmte Klassen begrenzt sein können. Die Objekte der physischen Welt dürfen innerhalb einer Augmented Area nur eine Repräsentation haben. Weil sie in verschiedenen Augmented Area liegen können, kann es mehrere Repräsentationen für ein Objekt geben. Außerdem alle Objekte, die zu einer Augmented Area gehören, werden in einem selben Spatial Model Server gespeichert, d.h. für jede Augmented Area gibt's nur ein Spatial Model Server, wobei ein Spatial Model Server mehrere Augmented Areas verwalten kann.

Noch eine wichtige Eigenschaft von Augmented Area ist, die Konsistenz zu behalten. Die Änderung des Zustands einer Augmented Area wird automatisch zu dem jeweiligen Augmented Area Model verbreitet, und umgekehrt löst die Änderung in dem Augmented Area Model auch einige Ereignisse aus, die den Zustand der realen Welt auch verändern.

Übrigens halten wir die Augmented Areas nicht für eingeschlossene Welt, sondern integrieren sie in eine globale Augmented Welt durch die Kombination aller Augmented Areas.

3.4 Augmented World Model

Um diese Integration von mehreren einzelnen Augmented Areas einfach zu machen, wird das Augmented World Model(AWM) in NEXUS eingesetzt. Die Augmented World Model sind als verteilte Computer Repräsentationen für die Augmented Areas zu verstehen, und wird in einheitlicher Weise beschrieben und durch das selbe NEXUS-Anwendungsinterface zugänglich gemacht.

Dieses Modell ist ein Objekt-orientiertes Informationsmodell und wird von virtuellen Objekten angereicht. Die modellierten Objekte existieren in der physischen Welt und können in unterschiedlichen Detail-Ebenen definiert werden, wie Räume, Gebäude, Straße oder Gebiete. Besonders sind die mobile Objekte auch zu modellieren, z.B. der mobile Benutzer selbst, der laufende Bus usw.. Die virtuellen Objekte sind die Metaphern für externe Informationen. Ein Beispiel davon ist der Virtual Information Tower(VIT), der als die Verbindung zu externen Informationen gebraucht wird, z.B. die HTML-Datei in WWW.

Das Augmented World Model hat offensichtlich viele Vorteile. Weil alle Modelle nur durch das einheitliche NEXUS-Anwendungsinterface zu erreichen sind, kann eine Anwendung andere Modelle sehr einfach wechseln, z.B. ein Tourist Guide System wechselt das globale Stadt Modell einfach zu dem lokalen Museum Modell, wenn der Benutzer ins Museum eintritt.

Ein anderer wichtige Vorteil von Augmented World Model ist, dass das einheitliche Format für die Modelle die Erweiterung von der Augmented World ermöglicht, d.h. neue Modelle können jeder Zeit in der globale Augmented World integriert werden, und nach der Integration können die NEXUS-Anwendungen sofort die neue Modelle benutzen. Diese Integration fungiert ähnlich wie die Integration von einem WWW-

Server in die Web, und damit kann die NEXUS-Plattform sehr schnell und weitverbreitet wachsen.

3.5 Standard Class Schema

Das Augmented World Model ist ein objekt-orientiertes Modell, so benutzen wir entsprechende Klassen, um die Objekte zu beschreiben. Das Augmented World Standard Class Schema besteht aus einer Menge Objektklassen und jeweilige Attributen. Weil es unmöglich ist, eine komplette Menge von Klassen für alle mögliche NEXUS-Anwendungen zu definieren, kann das Standard Class Schema nur die Objektklassen enthalten, die wesentlich für meiste Anwendungen sind. Das exakte Aussehen wird in [Meßmer 2001] festgelegt und besteht aus über 120 Klassen.

Alle NEXUS-Dienste sollen verträglich für die Klassendefinition sein, und jede NEXUS-Anwendung kann jeden NEXUS-Knoten nach den vordefinierten Objektklassen und Attributen anfragen. Z.B. durch einen NEXUS-Knoten erhalten eine Anwendung das Attribut „Menu“ vom Objekt „Restaurant“, das schon in dem Standard Class Schema existiert.

3.6 Extended Class Schema

Weil im Extended Class Schema die Objektklassen nicht komplette für alle mögliche Anwendungen sind, dann manchmal müssen wir die Klassen im Standard Class Schema noch erweitern, um den Bedarf für die Anwendungen zu erreichen. Dazu brauchen wir das Extended Class Schema, das die noch nicht definierte Objektklassen enthält. Dieses Schema bietet eine Möglichkeit, die existierende Modelle weiter zu entwickeln.

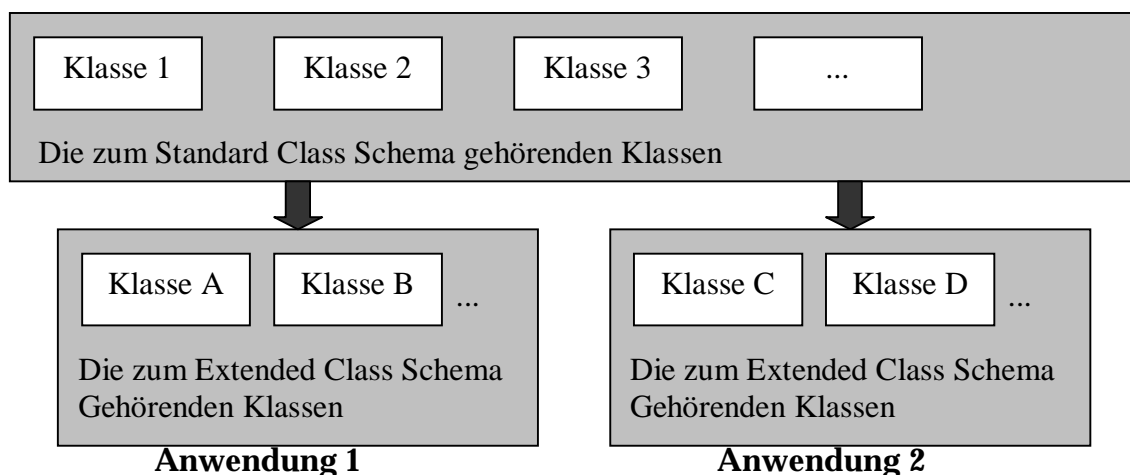


Abbildung 3-3: Anwendungen benutzen die Klassen gemeinsam in NEXUS

Ein einfaches Beispiel ist, dass ein neuer NEXUS-Dienst die Informationen für italienisches Restaurant anbieten möchte, d.h. dieser Dienst muss zuerst eine Klasse

für italienisches Restaurant haben. Aber wenn es im Standard Class Schema jedoch keine Klasse dafür gibt, dann kann der Dienst nur eine neue Klasse selbst definieren, die von einer sehr ähnlichen Klasse (z.B. die Klasse „Restaurant“) im Standard Class Schema abgeleitet wird, und fügt die neue zusätzliche Attributen gegen die Basisklasse ein. Solche Objektklassen wie „Italienisches Restaurant“ bilden das Extended Class Schema(siehe Abbildung 3-3) [Meßmer 2001].

Von einem Standard Class Schema können mehrere Extended Class Schemas abgeleitet werden, welche benutzen gemeinsam die Objektklasse im Standard Class Schema. Durch das Extended Class Schema können zukünftige Anwendungen auch die gewünschten Objektklassen selbst definieren.

3.7 Komponenten des NEXUS-Systems

Jetzt wollen wir die Komponenten des NEXUS-Systems mal vorstellen. Die Hauptkomponenten sind nämlich der Spatial Model Service, das Area Service Register und die Föderationskomponente. Die sind zuständig für die Verwaltung vom Augmented World Model.

3.7.1 Spatial Model Service

Der Spatial Model Service entsteht durch die Zusammenführung aller Spatial Model Server, und bietet das statische Teil vom Augmented World Model an. Jeder Spatial Model Server ist zuständig für eine Augmented Area, d.h. alle in einer Augmented Area liegende statische und virtuelle Objekte werden in das Spatial Model Server gespeichert.

Ähnlich wie die Aufstellung eines Webservers wird es jedem Dienstanbieter erlaubt, seinen eigenen Spatial Model Server mit den zu veröffentlichenden Informationen in den gesamten Spatial Model Service zu integrieren. Außerdem gleich wie Webserver muss der Spatial Model Server mit dem Internet in Verbindung bleiben, und muss noch das NEXUS-Interface benutzen.

Der Spatial Model Server ist nur in der Lage, statische Objekte zu speichern. Für die Speicherung den mobilen Objekten ist der Location Service zuständig. Weil die Positionen bei mobilen Objekten sehr oft verändert werden, sind die Spatial Model Server nicht geeignet dafür. Weitere Informationen über den Location Service können unter [Becker 2001] gefunden werden.

3.7.2 Area Service Register(ASR)

Der Area Service Register ist ein Repository, das die Informationen über Augmented Area enthält, nämlich die Klassen den zu einer Augmented Area gehörenden Objekten, das von dem Augmented Area abgedeckte Gebiet und der Name des Spatial Model Servers, der die Informationen des Augmentd Areas gespeichert hat. Die Funktionalität ist in Abbildung 3-4 gezeigt.

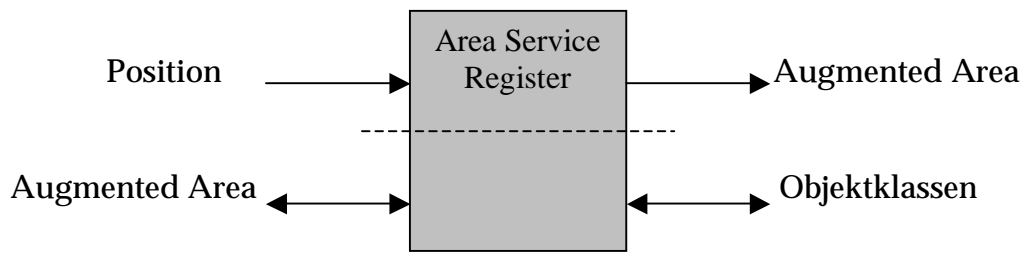


Abbildung 3-4: Die Funktionalität vom Area Service Register

Die im Area Service Register gespeicherte Informationen werden von der Föderationsebene benutzt, um einen Spatial Model Server bei einer Anfrage zu identifizieren. Deshalb muss jeder Spatial Model Server die von ihm verwaltete Augmented Area beim Area Service Register registrieren und die notwendigen Informationen abgeben. Weil der Area Service Register kein Dienstanbieter, sondern benötigt für die Anfrageverarbeitung ist, gehört er zu der Föderationsebene.

3.7.3 Die Föderationsebene

Die Föderationsebene bietet den NEXUS-Anwendungen eine einheitliche Sicht über die von den NEXUS-Diensten gelieferte Informationen. Diese Ebene liegt zwischen die Anwendungsebene und Dienstebene, und hat zwei Interface. Die NEXUS-Anwendungen senden Anfragen und empfangen Antworten durch das NEXUS-Anwendungsinterface, und alle NEXUS-Dienstanbieter verbinden sich mit der Föderationsebene durch das NEXUS-Dienstsinterface(siehe Abbildung 3-1).

Die Föderationsebene bearbeitet jede von den Anwendungen ankommende Anfrage in der folgenden Weise: Zuerst werden alle geometrische Prädikate aus der Anfrage extrahiert, um das Zielgebiet der Frage festzustellen. Aufgrund den Klassebeschreibungen für jeden Spatial Model Server werden die mit dem Zielgebiet überlappende Augmented Areas ermittelt. Da jede Augmented Area zu einem einzigen Spatial Model Server gehört, dann kann die Föderationsebene die Frage wieder aufbauen und zu den relevanten Spatial Model Servern weiterleiten. Danach versammelt die Föderationsebene die Antworten aus den sich beteiligenden Spatial Model Servern und integriert sie in einem einzelnen konsistenten Ergebnis. Schließlich wird das Ergebnis an die Anwendungen zurückgeschickt.

Die Föderationsebene wird auf jedem NEXUS-Knoten realisiert, d.h. jeder Knoten hat gleiche Fähigkeit. Die Anfrage aus den Anwendungen können grundsätzlich an alle NEXUS-Knoten geschickt werden, aber normale Weise nur an den nächsten Knoten. Außerdem kann ein NEXUS-Knoten noch sogenannte Value-added Dienste wie z.B. einen Navigationsdienst anbieten. Solche Dienste gehören nicht zu den NEXUS-Diensten, deshalb können die Anwendungen aber nur mit solchen Diensten durch seine eigene Interface Kontakt machen. Und sie selbst benutzen das NEXUS-Anwendungsinterface, um benötigte Daten aus NEXUS-Diensten zu erhalten.

3.8 Der Datenaustausch

Wie wir vorher gesagt haben, das Augmented World Model ist eine Projektion der physischen Welt zu einer virtuellen Welt. Diese Abbildung wird von den einheitlichen Beschreibungen für die physischen Objekte ausgeführt, nämlich die XML Sprache, die Augmented World Modeling Language(AWML) heißt.

Diese Sprache kann mit HTML verglichen werden. In beiden Sprachen werden die grundlegenden Semantiken durch eine „Markup“ Sprache definiert. Bei der HTML werden die Elemente wie z.B. „title“, „heading“, „image“, „table“ usw. definiert. AWML basiert auf dem Augmented World Model. Deswegen in AWML werden die Tags für die physischen Objekte, deren Attribute und deren Objektklassen definiert.

Die Anfragen aus Anwendungen werden durch die Augmented World Query Language(AWQL) dargestellt, in der das Zielgebiet und die Klassen der gesuchten Objekte angegeben werden. AWQL ist eine Query Sprache ähnlich wie SQL.

Wegen den deutlichen Vorteilen von XML wird es sich entschieden, die XML-Technologie in NEXUS für den Datenaustausch einzusetzen. XML unterstützt die optionalen Attribute, die eine wichtige Eigenschaft von NEXUS Objektklassen ist, und minimiert die Datenflüsse zwischen die NEXUS Komponente, weil durch XML die Anwendungen ihre Anfragen nur auf die beobachteten Attribute begrenzen können. Außerdem ist XML immer bekannter und in der Zukunft als Standard für den allgemeinen Datenaustausch einzusetzen. Viele Werkzeuge und Parse wurden schon entwickelt, um XML in vielen Plattformen immer einfacher zu verarbeiten.

3.9 Zusammenfassung

Das NEXUS-System ist eine offene und globale Plattform, die als Infrastruktur für ortsbewusste Anwendungen entwickelt wurde. Die Hauptabsicht davon ist, die mobile drahtlose Kommunikationen und kleine mobile Geräte wie PDA zu unterstützen. Für die mobile Geräte sind die Informationen meistens abhängig von der Positionen, nämlich die aktuelle Stellen den mobilen Benutzern. Durch NEXUS können die Anwendungen auf den mobilen Geräten die gewünschte ortsbezogene Informationen erhalten. Das ist gerade geeignet für meine Arbeit. Das Augmented World Model stellt die Objekte in der physischen Welt mit Computer Weise und ermöglicht alle Dienste in dem NEXUS-System eine einheitliche Sicht für die NEXUS-Anwendungen zu haben, dann können verschiedene isolierte Anwendungen das NEXUS-System durch das einheitliche Anwendungsinterface einfach benutzen. Für die Dienstanbieter Seite ist es auch sehr einfach, um die Dienste von unabhängigen heterogenen Dienstanbieter in der NEXUS-Plattform durch das Dienstinterface zu integrieren. Gesamtgesagt ist NEXUS für die mobilen Anwendungen eine ideale Auswahl.

4 Grundlagen für das VIT-System

Wir haben in Kapitel 2 und 3 über das Positionierungssystem und das NEXUS-System gesprochen. Beide Systeme sind für das VIT-Browser zu Verfügung zu stellen. Die Hauptaufgabe meiner Arbeit ist, die Inhalte von VIT auf dem Bildschirm eines PDAs anzuzeigen. Die Beziehungen zwischen die drei Komponente liegen in der Abbildung 4-1.

Wie es in dem Bild gezeigt hat, das Positionierungssystem empfängt die Signale aus dem mobilen Kommunikationsnetzwerk und berechnet die aktuelle Position des mobilen Benutzers mit den Feldstärken von den Signalen aus mehreren verschiedenen BTSs, und dann schickt das Ergebnis an die NEXUS-Plattform. Aufgrund der Position überprüft das NEXUS-System die Sichtbarkeiten von den VITs und liefert die VITs an den PDA, die sichtbar für den mobilen Benutzer sind. Am Ende zeigt das VIT-Browser für PDA die Inhalte von den gelieferten VITs.

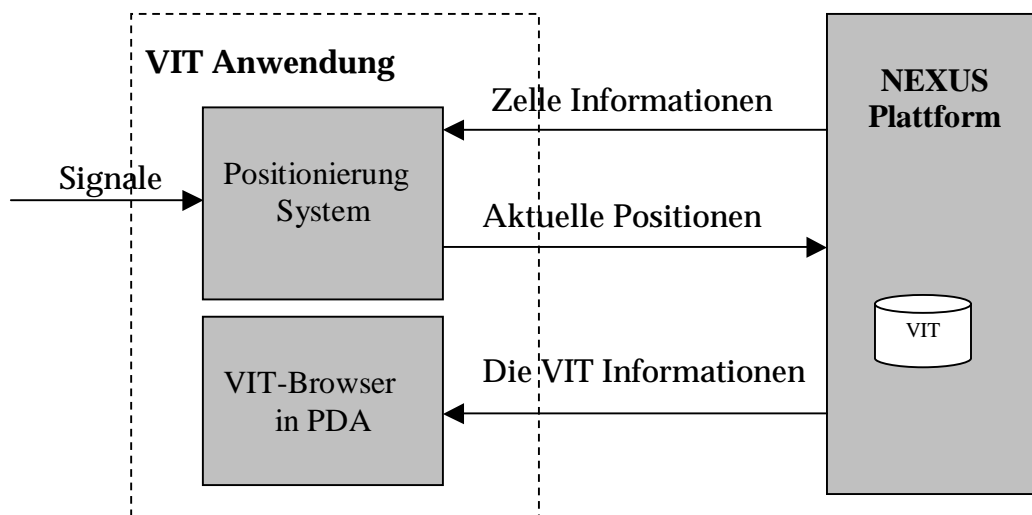


Abbildung 4-1: Die grobe Beschreibung für den Arbeitsverlauf

Es gibt noch ein Punkt zu beobachten, dass das Positionierungssystem nicht nur die ausgerechneten Positionen an die NEXUS-Plattform sendet, sondern muss noch der NEXUS-Plattform nach den Zellen-Informationen anfragen, um die notwendige Information wie die Positionen von den BTSs für die Berechnung zu bekommen. Weil die Zellen-Datenbank auch als ein Objektmodell in dem Spatial Model Server gespeichert werden muss, ist das Positionierungssystem selbst auch eine Anwendung von dem NEXUS-System.

Bis jetzt haben wir den groben Verlauf meiner Arbeit vorgestellt. Und im Kapitel 3 wurde der sogenannte VIT auch schon erwähnt, aber wir wissen nicht ganz genau, was VIT eigentlich ist. In folgenden Abschnitten wollen wir zunächst über den VIT diskutieren, und dann wird das Daten-Format für die Kommunikation zu dem NEXUS-System erklärt.

4.1 Virtual Information Towers

Das ganze NEXUS-System basiert auf den Augmented Areas, die von virtuellen Objekten angereichert werden. Ein wichtiges Beispiel für solche virtuelle Objekte ist nämlich Virtual Information Tower (Virtuelle Litfasssäulen). VIT ist als eine Versammlung von einer Menge strukturierte ortsbezogene Informationen zu verstehen. Jede VIT wird einem bestimmten Gebiet zugeordnet, und hat noch eine Zone für seine Sichtbarkeit.

4.1.1 Das Konzept für VIT

Heute in der mobilen Umgebung ist der Zugriff auf Informationen immer abhängiger von der Position des mobilen Benutzers, d.h. die gebrauchten Informationen sind ortsbezogene Informationen. Um diesen Zugriff unmittelbar und einfach für Benutzer zu realisieren, werden VITs im NEXUS entwickelt.

Wir können die VITs als die Werbungssäulen in der realen Welt vorstellen, und auf einer Säule werden mehrere Plakate „geklebt“. Jeder VIT legt sich auf einer geographischen Stelle fest, und hat eine „Höhe“. Diese „Höhe“ bestimmt ein Gebiet für die Sichtbarkeit vom VIT, d.h. ist der VIT immer höher, dann ist sein Sichtbarkeitsgebiet auch immer größer. Wenn ein Benutzer ins Sichtbarkeitsgebiet eines VITs eintritt, taucht dieser VIT auf dem Bildschirm des Benutzers auf, und der Benutzer kann auf die von VIT gelieferte Informationen einfach zugreifen. Außerdem können die auf einem VIT „geklebte“ Plakate auch ihre eigenen Sichtbarkeitsgebiete innerhalb dieses Sichtbarkeitsgebiet von dem VIT haben.

Ein VIT ist virtuell, und noch kompliziert. Der kann eine vollständige Struktur von Informationsobjekten (Plakate) anbieten, die als Active-Poster genannt werden. Solche Poster können verschiedene Typen sein, z.B. Audio, Video, Text, Image Posters usw. Sie können auch benutzt werden, um den Zugriff auf den existierenden Diensten und Informationssystemen wie WWW zu unterstützen. Es ist auch möglich, dass der Benutzer die Informationen zur Veröffentlichung auf dem VIT klebt, wenn er Recht hat.

Weil die VITs unterschiedliche Informationstypen unterstützen, können die für viele Anwendungen benutzt. Z.B. ein mobiles Tourist Guide System kann auf VITs basieren, die VITs stehen auf jeder bedeutenden Stelle, und bieten die Informationen für die Sicht an. Die VITs, die vor den Kinos und Theatern stehen, können als Ticket-Dienste benutzt werden. In meiner Arbeit bieten die VITs meistens die Homepages von wichtigen Gebäuden in jeweiligen Gebieten.

4.1.2 VITs im NEXUS

Innerhalb der NEXUS-Plattform werden die VITs im Spatial Model Server gespeichert. Normale Weise ist jeder VIT für ein bestimmtes Ziel herzustellen, z.B. für Tourist Guide, Kino usw. Und der Inhalt eines VITs kann ins ein Profil kurz

zusammengefasst. Solches Profil wird benutzt, um den entsprechenden VIT im Spatial Model Server einfach zu ermitteln.

Die VITs im NEXUS werden durch den Internet zugänglich gemacht. Das Komponent VIT-Manager verwaltet eine Menge VITs, und dadurch können die Benutzer den Inhalt von einem bestimmten VIT erhalten. In NEXUS werden alle physische und virtuelle Objekte durch die sogenannte AWML dargestellt wegen dem auffallenden Vorteil von XML. Deshalb müssen die VITs auch im AWML Format erzeugt werden. Das folgende Beispiel zeigt ein einfacher VIT durch AWML.

```
<awml>
<dataobject type="VIT" NOL="nexus://nexus://asstupro/49/13" kind="VIRTUAL">
  <NAME>Pragsattel</NAME>
  <EXTENT>
</EXTENT>
  <POS>POINT ( 9.17616100 48.81787600)</POS>
  <HEIGHT>
</HEIGHT>
  <ALTITUDE>
</ALTITUDE>
  <ADDRESS>
</ADDRESS>
  <OWNER>ifp</OWNER>
  <STARTPAGEADDRESS>http://vit.informatik.uni-
stuttgart.de/demo/vilis/city/pragsattel/pragsattel.xml</STARTPAGEADDRESS>
  <ACTIVEPLISTEADDRESS>http://vit.informatik.uni-
stuttgart.de/demo/vilis/city/pragsattel/pragsattel_active.xml</ACTIVEPLISTEADDRESS>
  <PHIERARCHYADDRESS>http://vit.informatik.uni-
stuttgart.de/demo/vilis/city/pragsattel/pragsattel_hierarchy.xml</PHIERARCHYADDRESS
>
  <EXPIRES>2005-12-31</EXPIRES>
  <VISIBILITY>MULTIPOLYGON ((( 11.17616100 48.81787600, 10.59037500
50.23209000, 9.17616100 50.81787600, 7.76194800 50.23209000, 7.17616100 48.81787600,
7.76194800 47.40366300, 9.17616100 46.81787600, 10.59037500 47.40366300,
11.17616100 48.81787600)))</VISIBILITY>
</dataobject>
</awml>
```

Über AWML wollen wir später noch genau erklären. Hier brauchen wir nur ein paar Elemente zu beobachten. In dem Element „dataobject“ wird eine VIT definiert, und für jeden VIT gibt's mehrere Attribute. Jeder VIT wird durch das Element „NAME“ identifiziert, das eindeutig im Rahmen von VIT-Manager ist. Um die geographische Position und das Sichtbarkeitsgebiet für einen VIT zu definieren, benutzen wir das einfache geographische Koordinationssystem WGS84, das eben in meinem Positionierungssystem benutzt wird. Dann kann der VIT-Manager die Positionsinformationen aus Positionierungssystem einfach verarbeiten ohne Umformung. Das Element „POS“ definiert die Position von VIT in WGS84, und das Sichtbarkeitsgebiet wird im Element „VISIBILITY“ als Polygon dargestellt, das durch die Positionen seinen Ecken spezifiziert wird. In den Elementen

„STARTPAGEADDRESS“, „ACTIVEPLISTEADDRESS“ und „PHIERARCHYADDRESS“ werden die Poster, die auf einem VIT geklebt werden, durch URL angeboten, d.h. die Informationen von VIT kann man nur durch WWW zugänglich machen.

Die VIT-Anwendungen greifen normale Weise auf die VITs mit Hilfe von Browsern zu, um die Inhalte zu lesen. Die VIT-Anwendungen sind beabsichtigt auf mobilen Geräten auszuführen, und kommunizieren mit NEXUS-System über das mobile Netzwerk und muss noch einen Positionssensor wie GPS zu Verfügung stellen. In meiner Arbeit statt dem GPS-System wird ein Mobiltelefon mit einem mobilen Kommunikationsnetzwerk zur Positionierung genutzt.

4.2 Das Datenformat

Wie es im Abbildung 4-1 gezeigt hat, als eine VIT-Anwendung fragt das VIT-Browser mit der aktuellen Position der NEXUS-Plattform nach entsprechendem VIT, und bekommt die VIT Informationen als die Antwort zurück. Die Datenflüsse dazwischen basieren auf die XML Technologie.

Im NEXUS-System wurden ein paar auf XML basierende Sprachen entwickelt, um Datenaustausch und Datenmanipulation zu unterstützen. Die sind AWML(Augmented World Modeling Language), MapPL(Map Predicate Language), AADL(Augmented Area Description Language), ChangeReport und AWQL(Augmented World Query Language). Davon sind nur AWQL und AWML sinnvoll für meine Arbeit, weil VIT Browser eine einfache NEXUS-Anwendung ist, und braucht nur die beide Sprachen zu implementieren, um Anfragen zu stellen und Antworten zu erhalten (siehe Abbildung 4-2).

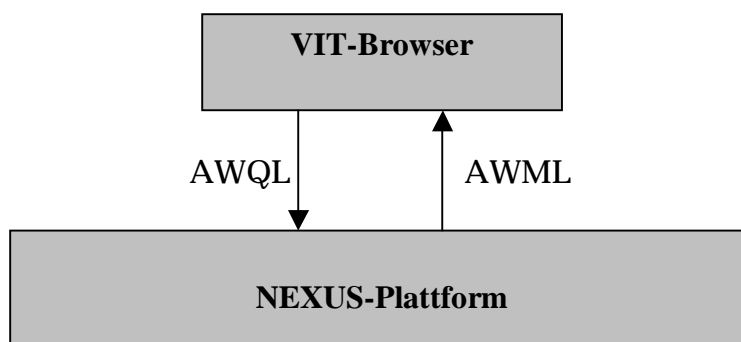


Abbildung 4-2: Datenaustausche zwischen NEXUS-Anwendungen und Plattform

4.2.1 Augmented World Query Language

Die Anfragen zur NEXUS-Plattform werden in AWQL formuliert. AWQL ist eine Anfrag-Sprache und basiert auf der XML-Datenstruktur. Die ist viel einfacher als SQL und kann die Datenflüsse zwischen die NEXUS Komponenten minimieren.

AWQL besteht aus folgenden Bestandteile:

- Ein „aas“ Element ist eine Liste mit Nexus Augmented Area Locator (NAAL). Die Anwendung kann durch ein „aas“ Element die betroffenen Dienstanbieter direkt bestimmen. Hierdurch kann die Föderationskomponente die sonst notwendige Suche nach den betroffenen Spatial Model Servern ersparen und den Suchraum auf Daten spezieller Anbieter beschränken.
- Mit Hilfe eines „scope“ Elements, das eine Liste von Extended Class Schemas enthält, wird angezeigt, welche Schemas die NEXUS-Anwendung versteht. In einem „scope“ Element wird ein Alias für ein Extended Class Schema definiert. Dieser Alias kann danach in der Antwort benutzt werden.
- Ein „restriction“ Element enthält die Bedingungen, die ein Objekt erfüllen muss, damit es zur Ergebnismenge gehört. In diesem Element können sieben Elemente beliebig angeordnet werden. Die sieben Elemente sind „and“, „or“, „not“, „in“, „equal“, „inside“ und „overlaps“. Mit einem „equal“ Element kann eine Anwendung bestimmen, welchen Wert ein bestimmtes Attribut haben soll. Das Element „in“ wird benutzt, wenn ein Attribut einen Wert aus einer angegebenen Wertemenge besitzen soll. Mit Hilfe der geometrischen Prädikate „inside“ und „overlaps“ lassen sich Objekte auswählen, die innerhalb einer Region sind bzw. sich mit einer Region überschneiden. Mit Hilfe von „and“, „or“ und „not“ lassen sich Anfragen mit mehreren Bedingungen verknüpfen.
- Da nicht alle Anwendungen alle Attribute eines Objektes benötigen, lassen sich diese Angaben mit Hilfe des Elementes „filter“ auf die gewünschten Attribute eines Objektes beschränken. Hierfür gibt eine Anwendung mit „includes“ die gewünschten Attribute und mit „excludes“ die unerwünschten Attribute an. Das Element „includeallother“ dient dazu alle Attribute einzutragen, die nicht durch das „excludes“ Element ausgeschlossen wurden. Durch ein „excludeallother“ Element werden alle Attribute eines Objektes entfernt, die nicht in „includes“ stehen.
- Ein „closest“ Element begrenzt die Anzahl der gesuchten Objekte. Durch den Wert des „num“ Attributs des „closest“ Elements wird die Anzahl der gesuchten Objekte angegeben.
- Das „generalization“ Element dient dazu, die geometrischen Attribute der Objekte zu generalisieren. Z.B. Details eines Attributes können entfernt werden.
- Mit Hilfe des „aggregation“ Elementes lassen sich Objekte zu einem größeren Objekt verschmelzen.

- Ein „update“ Element enthält die neuen Werte der Attribute, die geändert werden sollen.

Hier gibt's ein einfaches Beispiel für eine AWQL-Anfrage: Das Ergebnis soll die Name und Menu von den fünf am nächsten liegenden chinesischen Restaurant enthalten. Die genau Beschreibung für AWQL liegt in [Großmann 2001].

```

<awql>
  <restriction>
    <and>
      <equal>
        <attr name = "type"/>
        <nexusdata>restaurant</nexusdata>
      </equal>
      <equal>
        <attr name = "nationality"/>
        <nexusdata>chinese</nexusdata>
      </equal>
    </and>
  </restriction>

  <closest num = "5" acc="1">
    <nexusdata>
      <gml>... GML representation of an object describing my position ...
    </gml>
  </nexusdata>

  <filter>
    <includes>
      <attr name = "name"/>
      <attr name = "menu"/>
    </includes>
  </filter>
</awql>

```

4.2.2 Augmented World Modeling Language

Die Augmented World Modeling Language(AWML) stellt die Informationen über die Augmented World dar, und besteht aus eine Menge von Objekten aus der Augmented World. Für jedes Objekt werden der Typ, die Attribute und die Beziehungen zu anderen Objekten in AWML definiert. Alle Objekte von dem Augmented World Model können durch AWML beschrieben werden. Die Spezifikation von AWML liegt auch in [Großmann 2001]. Die wichtigste Funktion von AWML ist die Modellierung des Ergebnisses für die AWQL-Anfragen. Außerdem wird es beim Datenaustausch zwischen unterschiedlichen Diensten benutzt. Wir haben schon ein Beispiel im Abschnitt 4.1.2 gesehen. Dieses Beispiel ist nämlich ein AWML-Beschreibung für ein VIT im Spatial Model Server, wenn dieser

VIT von einer Anwendung verlangt wird, dann schickt Spatial Model Server einfach das AWML-Dokument zu der Anwendung. Deshalb durch AWML sind die Informationen im Spatial Model Server einfach zu verwalten, und die Datenflüsse werden auch minimiert.

4.3 Zusammenfassung

Im diesem Kapitel haben wir über die VITs und das Datenformat gesprochen. Die VITs sind virtuelle Objekte, welche die ortsbezogenen Informationen beschreiben sollten. Jede VIT hat ein Sichtbarkeitsgebiet, wenn ein Benutzer in dieses Gebiet eintritt, dann ist die jeweilige VIT für den Benutzer sichtbar. Alle auf der VIT geklebte Informationen können dann von dem Benutzer erhalten.

Weil alle VITs sich in dem Spatial Model Server befinden, dann muss die Anwendung zunächst mit dem Server kommunizieren, um nach den VITS zu abfragen. Die Datenaustausch zwischen die Anwendung und den Server werden durch die beide auf XML basierte Sprachen, nämlich AWML und AWQL dargestellt. D.h. die Anwendung muss XML-Dateien verarbeiten.

5. Palmprogrammierung

Jetzt kommt das interessante Thema, nämlich die Programmierung für Palm-Gerät. In meiner Arbeit wird das VIT-Browser so entwickelt, dass es auf dem Palm-Gerät laufen kann. Weil die ortsbewusste Anwendungen nur für mobile Geräte sinnvoll sind, wurde das NEXUS-System besonders für mobile Geräte entwickelt, um ortsbezogene Informationen anzubieten. Dann ist das Palm-Gerät ein wichtiger und sehr häufig benutzter Client vom NEXUS-System. Deswegen spielt die Palmprogrammierung auch eine wichtige Rolle in NEXUS Entwicklung.

5.1 Grundlagen der Palmprogrammierung

Wenn wir „Palm“ erwähnen, denken viele Leute an die erfolgreichste PDAs aus dem bekannten Unternehmen Palm-Computing, aber hier verstehen wir Palm lieber als eine Plattform, nämlich das Palm Operating System(Palm OS). Deshalb sind Palm-Geräte nicht unbedient aus Palm-Computing, aber muss ein Palm OS beherrschen.

Palm-Gerät ist ein sehr beliebtes mobile Gerät, und hat mächtige Fähigkeit, um viele kleine unterschiedliche Anwendungen darauf zu erlauben und fast überall zu benutzen. Alle Anwendungen für Palm-Geräte müssen selbstverständlich auf dem Palm OS entwickelt werden. Aber im Vergleich zum PC hat Palm-Gerät auch deutlich viele Einschränkungen, deshalb unterscheidet sich die Palmprogrammierung von der normalen Programmierung auf dem Desktop.

5.1.1 Einsatz von Palmprogrammierung

Die Palm-Geräte sind im Gegensatz zum Desktop kein Computer, sondern eine Ergänzung zum Desktop-System. Das Desktop-System ist für die Eingabe und Verarbeitung großer Datenmenge zuständig. Und die Palm-Geräte ermöglichen es dem Benutzer die Daten bei sich zu tragen, um sie irgendwo anzuschauen und zu ändern. Die folgenden Besonderheiten vom Palm-Gerät sollen den Einsatz von Palmprogrammierung beeinflusst[Grzan 2002]:

- Palm-Gerät besitzt keine Eingabemechanismen mit denen man Text effizient eingeben kann.
- Palm-Gerät hat eine geringe Leistungsfähigkeit, d.h. sie besitzen nur wenig Speicher und einen relativ langsamen Prozessor.
- Palm-Gerät hat einen sehr kleinen Bildschirm.
- Viele Palmprogramme müssen ihre Daten mit Desktopprogrammen austauschen.

Die offensichtlichsten Folgen für Palmprogramme ist, dass sie nur einen geringen Speicherverbrauch haben dürfen. Aber die Ausführungsgeschwindigkeit muss sehr hoch sein, da die mobilen Benutzer normale Weise nicht geduldig auf die Palmprogramme zu warten wollen. Außerdem sollte die Eingabe längerer Texte nicht auf einem Palm-Gerät gemacht werden, da dieses ziemlich mühsam ist. Ein weiteres Problem ist die Ausgabe der Daten. Der Bildschirm ermöglicht es nur relativ

wenige Daten gleichzeitig anzuzeigen. Es ist unmöglich, ein WWW-Browser im Palm-Gerät einzusetzen.

Auf dem Desktop versucht man möglichst viele Funktionen in ein Programm zu integrieren, um so viele unterschiedliche Aufgaben mit einem Programm erledigen zu können. Aber bei Palmprogrammen muss der Entwickler sich darauf konzentrieren, dass das Programm jeder Zeit nur eine Aufgabe lösen kann, da Palm OS keine parallelen Prozesse unterstützt. Sollen mehrere Aufgaben gelöst werden, so ist es oft sinnvoller mehrere kleine Programme zu entwickeln als ein großes.

5.1.2 Überblick von Palm OS

Jetzt wollen wir das Detail von Palm OS mal anschauen. Bei Palm OS gibt's ein Benutzer Interface für die Anwendungen. Dieses Interface erlaubt jeder Zeit nur eine einzige Anwendung ausgeführt zu werden. Deshalb, wenn eine Anwendung geöffnet wird, dann beherrscht die den ganzen Bildschirm.

Der Speicher

Es gibt zwei Arten von Speicher auf dem Palm-Gerät. Die erste Art ist das sogenannte Dynamic Memory, welches als Speicher für die Daten des Betriebssystems und der Palmanwendung dient. Die zweite Art ist das Storage Memory, in dem alle Daten sind, die nach dem Beenden eines Programms nicht gelöscht werden.

Der Prozessor

Palm-Geräte haben einen ziemlich langsamen Prozessor, der auf die Einsparung von Batterie optimiert ist. Die Geschwindigkeit des Prozessors beträgt im Normalfall 16 oder 20 MHz und lässt sich mit Hilfe von Programmen übertakten. Der Prozessor besitzt keine Fließkommaeinheit, weshalb wir diesen Datentyp bei Möglichkeit vermeiden sollten.

Die Ereignisse

Die Pam OS Anwendungen sind Ereignis-gesteuert. Die Programme empfangen Ereignisse, bearbeiten diese und warten danach auf das nächste Ereignis. Jeder Zeit wird nur ein Ereignis verarbeitet werden kann.

Die Benutzungsoberfläche

Die Benutzungsoberfläche hat ein paar Eigenschaften. Neben einem kleinen Display und einer niedrigen Auflösung(160x160 Pixel) gibt es auch ein paar typische Palm-Bildschirmelemente. Die wichtigsten sind wohl die Forms. Forms haben auf dem Palm in etwa die Funktion von Fenstern auf dem Desktop und zeigen die Bildschirmelemente des Programms an.

Die Dateien

Eine Anwendung auf dem Palm OS ist eine Resource Datenbank(PRC), die viele verschiedene Dateien enthalten soll. Eine Datei ist einfach eine Datenbank Rekord mit einem Typ und einem ID. Und in solchen Dateien werden die Inhalte und Schnittstellen von der Anwendung gespeichert werden, z.B. der Code, die Oberflächenelemente, Icons usw.. In Desktop haben die Resourc Datenbanken eine .PRC Extension, nämlich die PRC-Dateien, die direkt von Palm-Gerät heruntergeladen werden kann.

5.1.3 Palm OS Emulator

Der Palm OS Emulator(POSE) ist eine Software, die viele Type den Palm-Geräten nachbilden kann, und ermöglicht es die erstellen Programme zu testen ohne sie auf dem Palm-Gerät laden. Deshalb ist POSE sehr wertvoll für die Entwicklungen von Palmprogrammen.

Für POSE braucht man eine ROM-Datei. Jeder Palm Typ hat seine eigene ROM-Datei, und ohne ROM-Datei ist der POSE gleich wie ein Computer ohne Betriebssystem. Wir können uns als Palm Entwickler bei Palm-Computing registrieren, um unterschiedliche ROM-Datei zubekommen. Wenn wir nur die ROM-Datei für unsere eigene Geräte, dann laden wir die ROM-Datei einfach aus den Geräten durch ein mitgeliefertes Programm herunter.

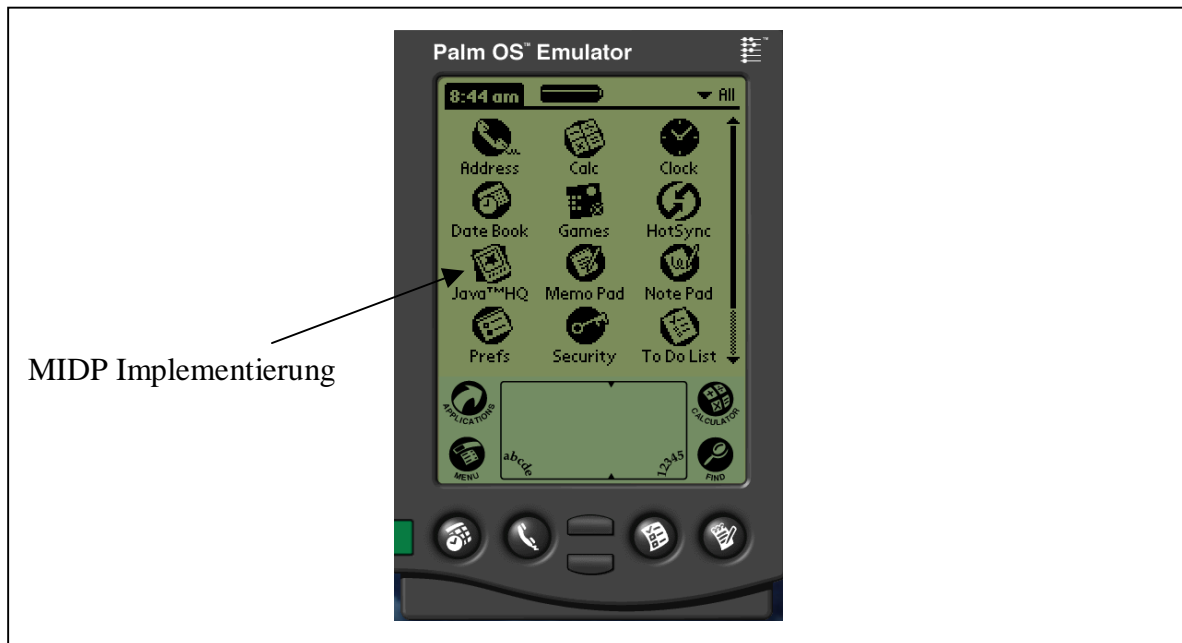


Abbildung 5-1: POSE auf dem Desktop

POSE können wir direkt von der Palm Homepage kostenlos herunterladen. Er ist nicht notwendig für die Entwicklung von Palmprogrammen, aber ohne ihm ist das Testen von Palmprogrammen kompliziert, weil man immer wieder die PRC-Datei auf Palm-Gerät herunterladen muss. Deshalb ist POSE das Entwicklungstool, das man auf jeden Fall haben sollte. Abbildung 5.1 zeigt die Erscheinung des POSEs auf dem Desktop.

5.2 C für Palm OS

Zur Zeit gibt's nur zwei Programmiersprachen für die Entwicklung auf Palm OS, nämlich C und Java. Seit lange Zeit wird C sehr häufig für Palmprogrammierung benutzt und kommt perfekt mit Hardware zurecht. Java für Palm OS ist eine relativ neue Technologie, aber dehnt sich sehr schnell im Bereich der Palmprogrammierung aus und wird ein starker Konkurrent gegen C.

5.2.1 Die Entwicklungstools für C

Für auf C basierende Palmprogrammierung sind viele verschiedene Entwicklungstools verfügbar, dadurch können wir C-Programme schreiben und kompilieren. Es ist wichtig, um ein entsprechendes Tool für unsere Entwicklungen auszuwählen, da jedes Tool seine Vorteile und Nachteile haben.

Für die Entwicklungen von den Palm-Anwendungen können wir Programme auf der Windows, Unix oder Macintosh Plattform schreiben. Die offizielle Palm-Entwicklungsumgebung ist CodeWarrior, die nur für Windows und Macintosh Plattform gültig, aber die hat ein sehr freundliches Interface für Entwickler, und ist auch einfach zu verwenden. CodeWarrior ist eine kommerzielle Entwicklungsumgebung, und kostet etwa \$300, so empfehle ich die nur für ziemliche große Brauchbarkeit. Außerdem ist Satellite Forms eine sehr gute Alternative zu CodeWarrior, wenn man die Preise leisten kann. Satellite Forms ermöglicht eine ganz schnelle Entwicklung, und der Benutzer braucht nicht unbedingt die low-level technische Einzelheit kennenzulernen.

Für die Unix und Windows Benutzer gibt's noch eine Menge Tools, die GNU C Compiler oder GCC heißt. Wenn man kein Geld bezahlen will, dann ist GCC die richtige Auswahl. GCC ist vielleicht die beste Wahl für ein normaler C-Benutzer mit seiner Flexibilität, aber manchmal auch ein bisschen schwer zu nutzen. In Folgendem Abschnitt wollen wir GCC etwa genau vorstellen.

5.2.2 GNU Pilot SDK

Am Anfang gab es eigentlich nur die Entwicklungsumgebung CodeWarrior, die auf Mac OS läuft. Viele Unix und Windows Benutzer möchten Palm-Anwendungen entwickeln aber wollten nicht bezahlen, deshalb hatten ein paar behilfliche Programmentwickler GCC mit einigen Tools so entwickelt, dass es Palm OS

Maschine Code erzeugen kann. Die Sammlung von solchen Tools ist nämlich GNU Pilot SDK.

Die Haupt Tools in dieser SDK sind GCC und PilRC[Howlett 1997]. GCC ist ein Generalcompiler, der ein von den beliebtesten C-Compilern auf Unix und Windows ist. GCC kompiliert die C-Datei und erzeugt eine einzelne Datei, die alle M68K Binary Dateien enthält. Bis jetzt unterstützt GCC C++ leider noch nicht vollständig. PilRC zerbricht eine RCP-Datei in mehrere BIN-Dateien. Als Eingabe erhält es eine Textfile(RCP-Datei), in dem die Oberfläche für Anwendungsprogramm beschrieben ist. Die erzeugten Dateien werden dann zum Programm gelinkt.

Außerdem gibt's noch zwei wichtige Tools. Das Obj-res Tool zerbricht die M68K Binary Datei aus GCC in mehrere getrennte Dateien, um später geeignet in eine PRC-Datei umzufassen. D.h. nach erfolgreicher Ausführung werden eine Menge Dateien mit dem Format „codeXXXX.filename.grc“ erzeugt, wobei XXXX das Datei-ID ist. Das Tool build-prc kombiniert die alle kleine Dateien in eine Datei-Datenbank, nämlich die PRC-Datei. Der ganze Arbeitsverlauf von diesen vier Tools wird in Abbildung 5-2 gezeigt.

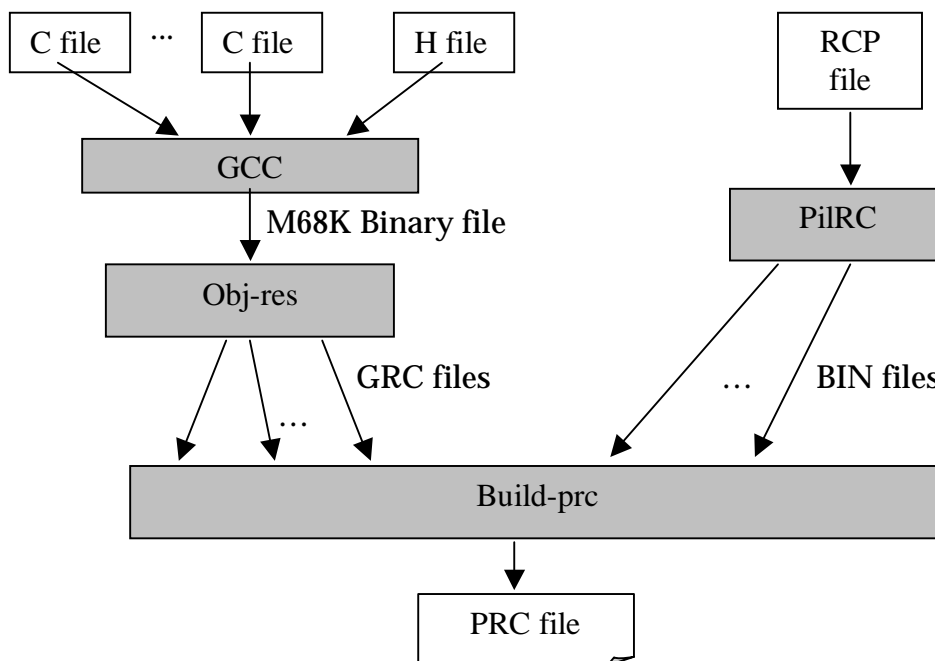


Abbildung 5-2: Arbeitsweise von GNU Pilot SDK

Um die Bibliotheken für C Palmprogrammierung zu erhalten, müssen wir noch eine Palm SDK installieren, und dem GCC noch den Pfad zu dieser SDK angeben. Die Palm SDK kann man bei Palm-Computing erhalten. Man sollte dabei darauf achten, dass man eine GCC angepasste SDK benutzt, d.h. eine richtige Version sollte ausgewählt werden, höhere Version ist nicht unbedingt geeignet. Für Anfänger wird eine ältere SDK(z.B. 3.0) empfohlen.

5.3 Java für Palm OS

Java 2 Micro Edition (J2ME) ist eine Version von der Java 2 Plattform, die besonders für kleine Geräte wie Mobiltelefon, PDAs verwendet werden. Bis heute werden solche Geräte immer noch oft in C oder C++ programmiert. Aber Sun hat J2ME entwickelt, und zeigt einen neuen Weg für Palmprogrammierung. Ich würde nicht sagen, dass Java C in der Zukunft im Palm Bereich ersetzen wird, aber es ist wirklich eine gute Alternative.

Java ist eine gute Auswahl für tragbaren Code, d.h. durch die Java Technologie ist es sehr einfach und sicher, Software in mobile Geräte zu installieren oder modifizieren. Als Plattform ist Java sehr bekannt wegen seiner Fähigkeit für die sichere Ausführung von unverlässigen Programmen. Während C nur auf Palm OS funktionieren kann, ist Java unabhängig von den Geräten, und ein Java-Programm kann überall laufen, worauf die MIDP unterstützt wird.

5.3.1 Konfiguration und die CLDC

Was ist genau J2ME? Es ist einfach eine Java 2 Plattform für kleine Geräte, hier bedeutet „kleine Geräte“ die mobile Geräte wie Mobiltelefon oder PDA. Solche mobile Geräte haben weniger Computing Power und kleineren Bildschirm, deshalb ist J2ME selbstverständlich als eine vereinfachte Vision von J2SE zu präsentieren. J2ME ist eine Untermenge von J2SE und unterstützt nur eine minimale Menge von Fähigkeiten, die für mobile Geräte verwendbar sind. In Abbildung 5-3 zeigt es die Beziehung zwischen J2ME, J2SE und J2EE.

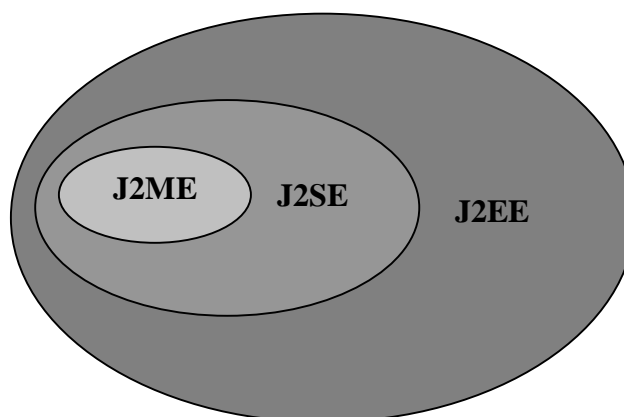


Abbildung 5-3: Die Beziehung zwischen J2ME, J2SE und J2EE

Ein wichtiger Unterschied zu J2SE ist, dass J2ME sich selbst noch aufgrund dem konkreten Typ von dem gezielten Gerät unterscheidet, d.h. die J2ME APIs sind nicht festgelegt. Ein Teil von J2ME ist ständig und allgemein gültig für alle Geräte, und der andere Teil wird für einen bestimmten Typ von Gerät wie Phone oder PDA definiert.

Weil es so viele verschiedene Typen von Geräten gibt, hat Sun J2ME in eine Menge von Konfigurationen und Profilen aufgeteilt. Eine J2ME Konfiguration ist eine minimale Menge von APIs, die nützlich für eine Klasse von Geräten ist [Morrison 2001]. D.h. die Konfiguration spezifiziert, welcher Teil von normaler Java Plattform benutzt werden soll, welche Fähigkeiten unterstützt werden sollen, usw.. Ein Standard Konfiguration für mobile Geräte heißt die Connected Limited Device Configuration (CLDC). Die CLDC spezifiziert eine Version von Java für kleine Geräte, die in der Lage sind, sich mit Netzwerk zu verbinden. Solche Geräte müssen mindestens einen 160KB gültigen Speicher und einen 16-bit Prozessor beherrschen, um sich als ein CLDC-Gerät zu qualifizieren.

Die CLDC imponiert ein paar Einschränkungen auf die Java Programm Sprache. Z.B. CLDC sagt, dass keinen Floating-Point Variable und Operation unterstützt wird, da die meisten gezielten Geräte keinen Floating-Point Unterstützung haben.

Die CLDC spezifiziert auch eine Untermenge von gültigen APIs. D.h. die auf einem CLDC Gerät laufende Anwendung ist abhängig von einer well-defined Untermenge von den java.lang, java.io und java.util Packages. Außerdem definiert CLDC noch eine neue Package javax.microedition.io, welche ein paar Interface für Netzwerkkommunikationen enthält.

5.3.2 Die MID Profile (MIDP)

Auf dem Oberteil von Konfiguration liegt eine Profile, welche eine spezifischere Menge von APIs für einen besonderen Typ von Gerät wie Mobiltelefon oder PDA ist [Morrison 2001]. Während die Konfiguration einen relativ großen Bereich von mobilen Geräten beschreibt, ist die Profile noch spezifischer und isoliert einen besonderen Typ innerhalb dieses Bereichs.

Eine auf der CLDC basierende Profile wurde schon definiert, nämlich die Mobile Information Device Profile (MIDP). Die MIDP beschreibt ein drahtloses mobiles Gerät wie Mobiltelefon. Weil die MIDP auf der CLDC gebildet wird, sind die CLDC APIs auch gültig für die MIDP-Anwendungen.

Glücklicherweise wurde eine MIDP für Palm OS gerade von Sun fertig gemacht. Diese ist noch besonders für Palmprogrammierungen und kann man sofort bei Sun frei herunterladen.

5.3.3 Die Entwicklung mit J2ME für Palm OS

Eine auf der MIDP-Spezifikation geschriebene J2ME Anwendung heißt ein MIDlet, das nur die Klassen und Interfaces nutzt, die in der CLDC- und MIDP-APIs definiert werden. Außerdem muss eine MIDlet noch durch einen sogenannten Vor-Prüfungsprozeß verarbeitet werden, d.h. wegen den Einschränkungen von mobilen Geräten müssen die MIDlets vor ihren Ausführungen nachgeprüft werden, um zu garantieren, dass die keine unzulässige Operation haben.

Die Vor-Prüfung passiert gerade nach der Kompilierung und erzeugt eine neue Klasse-Datei, die bereit für Testen und Ausführung ist. Aber vor der Ausführung muss die nachgeprüfte Klasse-Datei noch mit ein paar notwendigen Dateien in eine JAR-Datei zusammengepackt werden. Dabei wird noch in eine JAD-Datei erzeugt, welche die Informationen über die J2ME-Anwendung zeigt. Bis jetzt haben wir eine J2ME-Anwendung für MIDP, nämlich die beide JAR- und JAD-Dateien.

Aber für Palm OS sind nur PRC-Dateien gültig, und können auf Palm OS laufen. Wie erhalten wir diese PRC-Datei aus den JAR/JAD Dateien? D.h. wir müssen die J2ME-Anwendung in eine Palm-Anwendung umsetzen. Bei Sun's MIDP für Palm OS gibt's ein PRC Converter Tool für solche Umsetzung. So können wir endlich eine PRC-Datei bekommen, und entweder ins ein Simulator oder direkt ins PDA herunterladen. Die ganze MIDlet Entwicklungsprozeß steht in Abbildung 5-4.

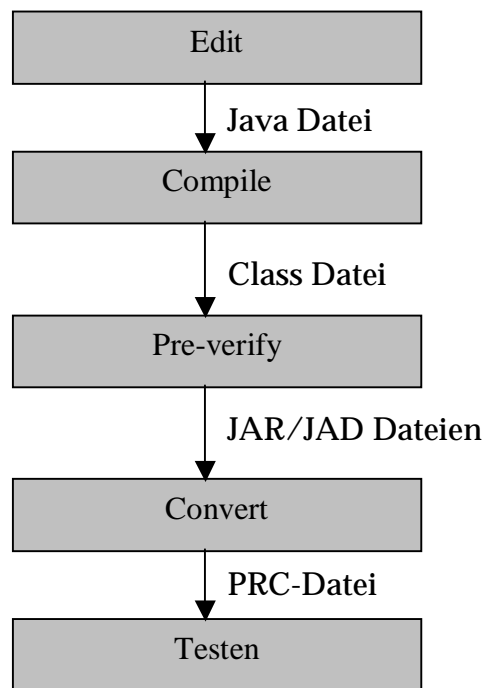


Abbildung 5-4: Die MIDlet Entwicklungsprozeß

Auf der PDA-Seite um die heruntergeladene Anwendungen auszuführen, müssen wir noch etwas erledigen, nämlich eine MIDP Implementierung für Palm-Geräte installieren. In den Dateien, die wir von Sun kostenlos heruntergeladen haben, gibt's eine *Java™ HQ* Datei, die wie eine JVM auf der Desktopplattform funktioniert (siehe Abbildung 5-1). Mit dieser MIDP-Implementierung auf Palm OS dann können wir alle MIDP-Anwendungen ausführen.

Für J2ME gibt's auch ein paar Entwicklungstools, davon sind drei Tools häufig zu benutzen: Java 2 SDK, J2ME Wireless Toolkit (J2MEWTK) und Visual Development Environment. Java 2 SDK kennen wir schon sehr gut. J2MEWTK ist eine Standard Toolkit von Sun und enthält ein Bytecode Prüfer und ein paar J2ME Simulator für Testen.

Bei Visual Development Environment gibt's zur Zeit KtoolBar, Forte for Java und CodeWarrior for Java usw.. Durch sie können wir die Command-Line Tools von Java 2 SDK und J2MEWTK verzichten und alle Aufgaben der Entwicklung automatisch durchführen lassen. So geht's bei Programmierung ziemlich schnell und bequem.

5.4 Zusammenfassung

Die Entwicklung von Palmprogrammen unterscheidet sich in einigen Bereichen sehr stark von der Entwicklung von Desktopprogramm. Es gibt allerdings auch Gemeinsamkeiten. Die Anwendungen sind ereignisgesteuert und die zur Verfügung stellende Programmierungssprachen sind C und Java. Es gibt auch mehrere Entwicklungstools für beide Sprachen, und die Auswahl für ein entsprechendes Werkzeug ist auch wichtig. Außerdem ist der Einsatz eines Palm-Simulators bei der Entwicklung ganz sinnvoll. Als Ergebnis der ganzen Entwicklung wird die PRC-Datei ins Palm-Gerät heruntergeladen und auf dem Bildschirm gezeigt.

Für ein Java Programmentwickler oder Jemand, der gelegentlich für Hobby Palm-Anwendungen entwickelt, dann würde ich J2ME empfehlen, da es mit J2ME einfach und bequem zu entwickeln ist. Für die etwa große und komplizierte Anwendungen, besonders die speziellen Anforderungen auf die Hardware haben, sollte es mit C programmieren, weil C die Hardware einfach gut unterstützt.

6 Die Implementierungsaspekte

In diesem Kapitel handelt es sich um die Implementierungsaspekte für die VIT-Browser System in meiner Arbeit.

6.1 Die Grundidee

Wir haben vorher schon über das Positionierungssystem, das NEXUS-System und das Palm OS gesprochen. Diese drei Systeme bieten die Grundunterstützungen für das ganze VIT-Browser System an. In Abbildung 4-1 habe ich schon den groben Ablauf dieses Systems gezeigt: das Positionierungssystem empfängt die Signale und berechnet die aktuelle Position des mobilen Benutzers; das NEXUS-System bietet die notwendigen Zelle-Informationen für die Positionsberechnung und die VIT-Informationen an; das Palm-Gerät zeigt die ortsbezogenen Informationen auf dem Bildschirm an. Alles klingt einfach und interessant.

Es ist schon interessant, aber gar nicht so einfach, weil es sich um viele neue Technologien über Hardware und Software handelt. Und in folgenden Abschnitten werden wir die eingesetzten Technologien mal ins Detail vorstellen und auswerten. Aber zuerst diskutieren wir den Ablauf dieses Systems.

Um diesen Ablauf einfach zu verstehen, teilen wir den in zwei Phasen ein, nämlich die Vorbereitungsphase und die Browsersphase. Jede Phase ist für bestimmte Funktionen zuständig. Die beide Phasen werden nicht gleichzeitig ausgeführt, und die Vorbereitungsphase selbstverständlich passiert vor der Browserphase. Abbildung 6-1 zeigt die Beziehung zwischen beide Phasen.

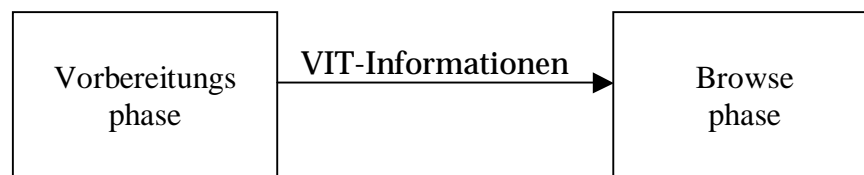


Abbildung 6-1: Die Beziehung zwischen Positionierungs- und Browsersphasen

Hier sind die VIT-Informationen immer noch nicht die Inhalte von VITs, sondern eine VIT-Struktur, d.h. eine Menge von den für Benutzer sichtbaren VITs und die zugehörige Active-Poster. Wir haben vorher vorgestellt, dass auf einem VIT mehrere Poster geklebt werden, solche Poster bieten die Hinweise für jeweilige ortsbezogene Informationen an, nämlich die Web-Adresse, und dadurch kann man erst die Inhalte bekommen, später wird es noch genau erklärt.

6.2 Die Vorbereitungsphase

Das Hauptziel meiner Arbeit, nämlich die ortsbezogenen Informationen auf PDA anzuzeigen, wird offenbar in der Browsersphase erreicht, aber trotzdem liegt die

meiste Arbeit in der Vorbereitungsphase. D.h. in der Vorbereitungsphase müssen wir die aktuelle Position des Benutzers berechnen und die sichtbare VITs ermitteln. Und wie können solche Ziele erreicht werden?

6.2.1 Kurze Beschreibung

Unsere Positionierung basiert auf dem Global System for Mobile Telecommunication(GSM), deshalb müssen wir zunächst die Signale aus dem mobilen Kommunikationsnetzwerk empfangen, und dann die Signale analysieren und der Zellen-Datenbank anfragen, um die Zellen-Informationen für die bedienende BTS und die Feldstärke von mehreren unterschiedlichen benachbarten BTSs zu finden. Danach können wir mit den gültigen Informationen die aktuelle Position berechnen.

Mit der ausgerechneten Position können wir die sichtbare VITs aus der NEXUS-Plattform ausholen. Wegen der Datendarstellung von VITs innerhalb Spatial Model Server können wir die zu einem bestimmten VIT gehörende Active-Poster noch nicht direkt bekommen, d.h. wir können nur zuerst einen Hinweis davon kriegen, also in meiner Arbeit ist dieser Hinweis eine Web-Adress, welche die Active-Poster in der Internet zeigen sollte. Und jeder Active-Poster bietet wieder einen Hinweis an, der die eigentlichen ortsbezogene Informationen zeigt. Diese Struktur über VITs und Active-Poster kann die Browsephase nutzen, um das endliche Ziel zu erreichen.

6.2.2 Die verwendete Tools

Für meine Arbeit habe ich folgende Tools benutzt, um das System in Vorbereitungsphase aufzubauen. Die sind ein Mobiltelefon(Siemens m35), ein Palmpilot(m105) und die Spatial Model Server im NEXUS. Siemens m35 wird klar als Sensor für Positionierungszweck verwendet, und außerdem funktioniert es noch als Modem, dadurch der Palmpilot mit dem Netzwerk in Verbindung bleiben kann. Der Spatial Model Server bietet die gewünschten Informationen an, er ist die Basis für das ganze System.

Den Palmpilot sollen wir genau mal beobachten. Aus Entwurfsaspekt soll dieses Gerät eigentlich als die Plattform für meine Anwendung sein, d.h. es berechnet die Position, stellt Fragen zum Spatial Model Server, verarbeitet die Datenflüssen usw., aber im Kapitel 5 haben wir schon gesehen, dass Palm-Geräte viele Einschränkungen haben, z.B. die kleine Speicherkapazität, die niedrige Leistungsfähigkeit, keine Unterstützung für Floating-Berechnung usw., und die hohe Leistungsgeschwindigkeit ist sehr wichtig für die mobile Benutzer. Deswegen ist es nicht geeignet, dass der Palmpilot aufwendige Aufgaben machen soll. Deshalb habe ich entschieden, wieder ein klein Server zwischen die Palm-Gräte und das Spatial Model Server einzusetzen, ich nenne das ein Zwischenserver, das die meiste Aufgabe von Palm übernehmen soll. So sind die Berechnung und Datenverarbeitung einfach zu realisieren und schnell auszuführen, und auf der Desktopplattform unterstützt es problemlos fast alle Technologien. Das ist eine günstige Lösung, der

einzigste Nachteil ist, dass man sich noch um ein Server kümmert. Abbildung zeigt der Zusammenbau allen beteiligten Elementen.

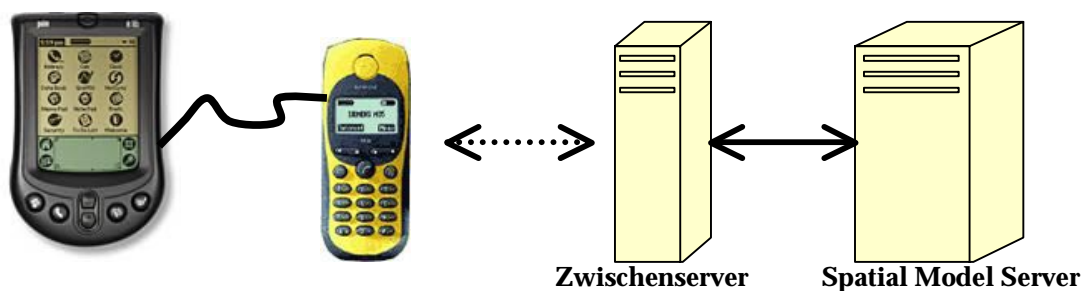


Abbildung 6-2: Der Aufbau des Systems in Vorbereitungsphase

6.2.2.1 Mobiltelefon

Das Siemens m35 wurde in meiner Arbeit benutzt, und der Netzanbieter ist D1, da D1 bessere Signalsdeckungen hat und ziemliche vollständige Zellen-Informationen anbietet. Durch das Siemens-Handy können wir die folgenden Informationen erhalten:

- Das Zelle-ID und die Feldstärke für die bedienende Zelle.
- Die Kanal-Nummer und die Feldstärken für die benachbarten Zellen.

Für die Kommunikation zwischen Mobiltelefon und Palm-Gerät werden die AT-Befehlen mit dem SIM-Toolkit Interface benutzt, das erzähle ich später in Abschnitt Datenflüssen.

Als Modem ist Mobiltelefon für die physische Verbindung zwischen das Palm-Gerät und Internet zuständig.

Um das Mobiltelefon mit dem Palm-Gerät zu verbinden, ein Kabel ist gebraucht. Ein einzelnes Kabel besonders für die Verbindung dazwischen ist schon teurer, so haben wir eine günstiger Lösung, d.h. das Kabel vom Palm-Gerät und das Kabel von Phone verbinden sich miteinander durch ein Null Modem Kabel, das auf beiden Seiten gleiche Köpfe hat. Dann mit zweimal Stecken bleiben die Mobiltelefon und Palm-Gerät zusammen miteinander in Verbindung.

6.2.2.2 Palm-Gerät

Für die Auswahl vom Palm-Gerät gibt's keine besondere Anforderung, ein m105 Gerät schon reicht. Die Arbeit für diesen Palmpilot ist auch nicht so viel in der Vorbereitungsphase, d.h. es ist nur zuständig für die Verbindungen zum Mobiltelefon und zum Zwischenserver, und die Daten aus Mobiltelefon einfach

weiter zu liefern. Deshalb muss die Anwendung für Palm OS hier zwei Mechanismen beherrschen, nämlich die Serial-Kommunikation und Internet-Kommunikation.

Serial-Kommunikation

Die Verbindung zum Mobiltelefon darf nur durch den Serial-Port realisieren. Deshalb muss das Anwendungsprogramm Serial-Kommunikation verwenden. Eigentlich hatte ich entworfen, alle Programme auf Palm OS mit J2ME zu entwickeln. Dann tauchte ein unangenehmes Problem, dass J2ME keine Serial-Kommunikation unterstützt [Hatler FAQ], die hat einfach keine Klasse für Serial-Verbindung. D.h. nur wegen diesem Problem kann ich J2ME leider nicht einsetzen, sondern C. Auf Palm OS hat C die Hardware viel besser unterstützt, weil C die Bibliothek von Palm SDK direkt benutzt.

Die Palm OS Kommunikationssoftware unterstützen die Serial-Kommunikationsfähigkeit, und haben eine Menge Funktionen, die sogenannte Serial-Manager heißt. Der Serial-Manager ist zuständig für Byte-Code Serial I/O, und damit kann man eine Serial-Verbindung einfach aufbauen.

Um einen Serial-Port zu öffnen, wir sollen die Methode *SerOpen* mit notwendigem Parameter wie Portsnummer, gewünschte Geschwindigkeit usw. nutzen. Wenn dieser Port schon geöffnet ist, dann wird eine Warnung zurückgeliefert und die Öffnungszahl von Port erhöht sich. D.h. ein Port kann für mehrere Aufgaben geöffnet werden. Durch *SerClose* kann man einfach den Port schließen. Es ist wichtig, immer den Port rechtzeitig schließen nach *SerOpen*, da ein geöffneter Port mehr Energie verbraucht, deshalb sollen wir den Port nicht lange öffnen lassen.

Alle Daten in der Serial-Kommunikation sind nach dem Word oder DWord Format. Um die Daten zu senden rufen wir die Methode *SerSend*, diese Funktion blockiert, bis alle Daten schon ausgesendet wurden. Bei *SerReceive* müssen wir ein Buffer mit den Größe- und Timeout-Werten spezifizieren, *SerReceive* blockiert auch bis alle gewünschte Daten da sind. Dann durch *SerReceiveCheck* können wir die Daten in Buffer lesen. In [Palm 3.0] gibt's vollständige Referenz für die Methode.

Internet-Kommunikation

Außer der Verbindung zum Mobiltelefon soll das Palm-Gerät noch in der Lage sein, durch das Standard TCP/IP Protokoll eine Verbindung mit anderen Rechnern in dem Internet (z.B. unser Zwischenserver) aufzubauen, und Daten auszutauschen. Deshalb müssen wir Sockets in dem Anwendungsprogramm für Palm OS einsetzen.

Sockets sind ein Kommunikationsmechanismus. Das ursprüngliche Sockets Interface tauchte in die Berkeley Software Distribution 4.2 für Unix in Jahr 1983 auf. Das Berkeley Sockets Interface ist das Standard-API für Netzwerkanwendungen. In dem Sockets Modus, die Daten werden auf einer Maschine ins ein Socket gesendet, und kommen aus einem Socket von einer anderen entfernten Maschine. Mit einem verbindungsorientierenden Socket-Interface können wir solche Kommunikationen zwischen zwei Maschine aufbauen, um Daten auszutauschen (siehe Abbildung 6-3).

Sockets erlaubt auch ein verbindungslos Modus für Datagrams-Übertragung. Die genaue Beschreibung ist in [Winton 2001].

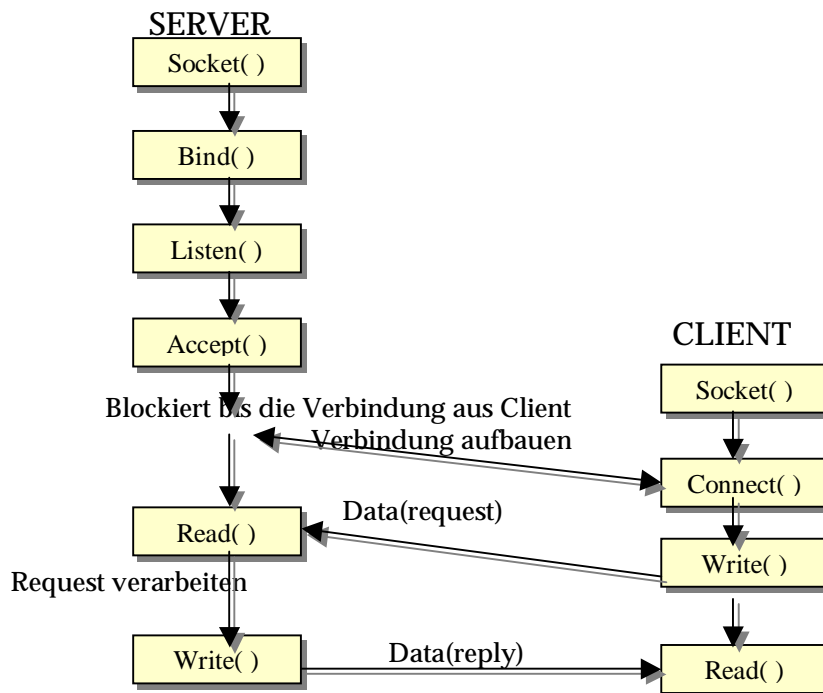


Abbildung 6-3: Socket Aufruf für verbindungsorientiertes Protokoll

Eigentlich möchten die Palm OS Designer auch ein solche Berkeley Sockets Interface in Palm SDK entwickeln. Aber wegen vielen Einschränkungen konnten sie nur eine Net-Bibliothek erzeugen, welche die Berkeley Sockets Arbeitsweise folgt, und die Einschränkungen anpasst. Mit dieser Bibliothek kann eine Palm-Anwendung Verbindungen mit allen anderen Rechnern im Netzwerk durch z.B. TCP/IP machen. Und das API in dieser Bibliothek ist ein socket-ähnliches Interface. Der Unterschied zwischen Net-Bibliothek und Berkeley Sockets Interface ist, dass die Net-Bibliothek noch drei zusätzliche Parameter mehr akzeptiert. Deshalb wurde ein „Klebstoff“ entwickelt, um das Berkeley Sockets Interface auf dem Palm OS Net-Bibliothek zu setzen. Dieser Klebstoff enthält Macros, Funktionen und Definitionen, die das Palm OS Netzwerk Interface ins das Berkeley Sockets API einpackt, dann könne die Entwickler das traditionale Berkeley Sockets Interface statt des Palm OS Interfaces aufrufen. Z.B. die Funktion für eine Erzeugung eines Sockets in Net-Bibliothek ist *NetLibSocketOpen()*, aber bei Berkeley Sockets ist einfach *socket()*, dann der Klebstoff definiert *socket()* als ein Macro:

```
#define socket(domain, type, protocol) \
    NetLibSocketOpen(AppNetRefnum, domain, type, protocol, \
        AppNetTimeout, &error)
```

Die zusätzliche Parameter werden in eine Head Datei definiert, nämlich *Unix\Sys_Socket.h*. Jetzt können wir einfach die Funktion *socket()*, und das Klebstoff setzt die in Palm OS Interface um.

6.2.2.3 Zwischenserver

Der Zwischenserver spielt die entscheidende Rolle in der Vorbereitungsphase. Der wird auf einer mit Internet verbindenden PC-Plattform realisiert, und absolut mit Java Programmiersprache entwickelt.

Der Zwischenserver muss viele Aufgaben schaffen, z.B. die originale Daten aus Mobiltelefon analysieren, die Position berechnen, mit Spatial Model Server kooperieren, usw.. Wie es in Abbildung 6-2 zeigt, hat er zwei Schnittstelle nach Außen, eine ist für die Verbindung mit Palmpilot, die andere ist für den Spatial Model Server. Die Schnittstelle zum Palm-Gerät wird natürlich auch durch Sockets realisiert. Sockts Technologie wird von Java sehr gut untergestützt, und ist sehr einfach zu nutzen[Java Dokumentation]. Aber die andere Schnittstelle zum Spatial Model Server muss eine andere Technologie verwenden, nämlich das Simple Object Access Protocol(SOAP), da das Interface vom Spatial Model Server auf SOAP basiert.

SOAP

SOAP ist eine neue Technologie, die von mehreren bekannten Unternehmen wie z.B. IBM, Microsoft, DevelopMentor usw. entwickelt. SOAP ist ein einfacher, erweiterbarer und Plattformübergreifender Datentransportsmechanismus, der in der Lage ist, die Probleme von verteilten Anwendungen zu lösen. Alle SOAP Nachrichten sind durch XML Technologie geschrieben, und werden über http übertragen.

SOAP stellt einen einfachen Mechanismus zum Austausch von strukturierter und typisierter Information zwischen miteinander kommunizierenden Rechnern in einer verteilten Umgebung zur Verfügung. Außerdem bietet SOAP noch ein einfacher Mechanismus an, um die entfernten Funktionsaufrufe(RPC) zu benutzen. Z.B. der Zwischenserver spielt als Client gegen Spatial Model Server, und als Server bietet der Spatial Model Server ein paar Methoden auf seinem Interface an. Um diese Dienste zu benutzen, muss die Anwendung einen RPC verwenden. Die RPCs werden absichtlich in dem XML-Datenstrom eingepackt und übertragen, um die RPCs einheitlich darzustellen, so können wir auf existierenden Standards wie http und XML noch ein einheitliches und erweiterbares Protokoll entwickeln.

Um eine entfernte Methode aufzurufen, werden im Allgemeinen die folgenden Informationen veröffentlicht[Apache SOAP 2001]:

- Der URI des Servers
- Der Methodename
- Vorhandene aktuelle Parameter der Methode

SOAP ist eine Programmiersprachen-unabhängige, Betriebssystem-unabhängige, und Plattform-unabhängige Technologie und wird ganz breite unterstützt.

JDOM gegen DOM

Außer der Kommunikation zum Spatial Model Server hat der Zwischenserver noch eine ganz wichtige Aufgabe, nämlich die XML-Dokumente zu verarbeiten. Da im Spatial Model Server alle Objekte durch sogenannte AWML dargestellt werden, und die Datenflüsse zwischen Spatial Model Server und andere Rechner meistens auch auf AWML und AWQL basieren, und die beide Sprachen basieren wieder auf der XML-Technologie. D.h. wenn man etwas mit Spatial Model Server zu tun möchte, dann trifft er sich unvermeidlich mit den XML-Dokumente.

Um die XML-Dateien zu verarbeiten, nämlich die XML-Dokumente zu parsen, brauchen wir einen XML-Parser. Zur Zeit gibt's schon mehrere XML-Parser auf dem Markt. Sie unterscheiden sich in der Programmiersprache, in der Standardunterstützung, oder in der Geschwindigkeit usw.. Deshalb ist es auch wichtig, einen entsprechenden Parser für unsere Anwendung auszuwählen.

Die Grundaufgabe vom XML-Parser ist, XML-Dokumente zu empfangen und die Inhalte(Elemente, Attribute usw.) davon der Anwendung zur Verfügung zu stellen. So werden die APIs für Parser noch benötigt, um mit unterschiedlichen Programmiersprachen zusammenzuarbeiten. DOM und JDOM sind beide APIs, welche die Dienste für die Verarbeitung von XML-Dokumenten anbieten. Also, ich habe die beide beobachtet und endlich die Entscheidung für JDOM getroffen.

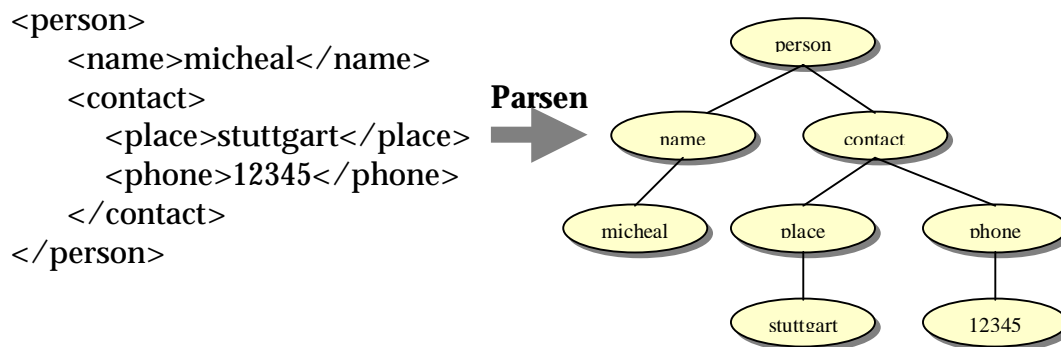


Abbildung 6-4: Der Aufbau eines DOM Baums

Das Document Object Model(DOM) ist eine API, die von W3C für die Erzeugung und Manipulierung für die Struktur und Inhalt vom XML Dokument entwickelt. Nach der Parsung von einem XML-Dokument wird jedes Bauelement wie Element, Text oder Bemerkung ins ein Objekt(auch sogenannt Knoten) abgebildet[Akif 2001]. Alle diese Knoten bauen einen Baum im Speicher der virtuellen Maschine auf. Das Root-Element des XML Dokuments fungiert als Wurzel für diesen Baum. Abbildung 6-4 zeigt ein einfaches Beispiel dafür.

Mit diesem Baum dann können wir durch DOM-API die Struktur eines XML-Dokuments manipulieren. Weil dieser Baum der ganzen Zeit während der Anwendung in dem Speicher steht, ist es offensichtlich nicht schön, wenn das XML Dokument zu groß ist.

Java Document Object Model(JDOM) ist eine ähnliche API wie DOM, aber die hat nichts mit DOM zu tun. JDOM wurde besonders für Java Entwickler entworfen, d.h. JDOM ist Java-API, deshalb statt der Definierung für Interfaces für Manipulation wie DOM nutzt JDOM einfach die Java Collections APIs, so ist JDOM intuitiv und effektiv für Java-Anwendungen[Java & JDOM 2001]. Außerdem geht JDOM relativ sparsam mit Speicher um.

JDOM benutzt einen XML-Parser(z.B. Xerces) und unterstützt SAX und DOM, d.h. JDOM kann DOM Dokumente und SAX-Events verarbeiten und entsprechenden Output erzeugen. Abbildung 6-5 demonstriert die verschiedenen Möglichkeiten, wie JDOM mit SAX und DOM zusammenarbeitet.

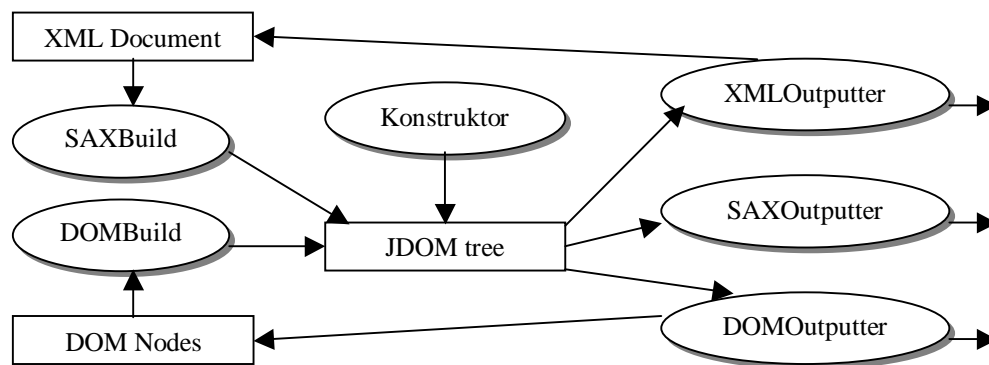


Abbildung 6-5: Programmflüsse des JDOM Dokumentes

Durch die Pakete *org.jdom.input* und *org.jdom.output* in JDOM-APIs wird einen JDOM-Baum erzeugt oder ausgesendet. Für Erzeugung des JDOM-Baums ist die Klasse *SAXBuild* durch eine SAX-Implementierung besonders häufig genutzt. Da der DOM selbst noch SAX benutzt, um XML-Dokument zu parsen, so ist es nicht sinnvoll, die Klasse *DOMBuild* zum Parsen verwenden. Für Ausgabe bietet die Klasse *XMLOutputter* die Standardmöglichkeit, ein JDOM-Dokument als XML in einen beliebigen Ausgabestrom zu senden. Mehrere Informationen über JDOM kann man in der Apache Homepage finden.

6.2.2.4 Spatial Model Server

Der Zwischenserver ist eigentlich eine Anwendung für NEXUS-System, und statt der Verbindung mit Spatial Model Server durch Föderationsknoten soll der Zwischenserver direkt mit einem bestimmten Spatial Model Server kommunizieren, und alle notwendige Informationen dabei abholen. Hier spielt der Spatial Model Server als ein Informationsanbieter für meine Anwendung.

Wir haben schon im Kapitel 2 über Spatial Model Server gesprochen. Ein Spatial Model Server ist eine Kopie von der physischen Welt mit statischen Objekten wie Gebäude, Bäume, Straßen usw.. Die Hauptaufgabe davon ist, die räumliche Objekte zu beschreiben, und die Beschreibungsdaten anzubieten.

Für meine Arbeit handelt sich um zwei Objekt-Modelle in dem Spatial Model Server, nämlich das BTS(Base Transceiver Station)-Model und das VIT-Model. Über das VIT-Model wissen wir schon, dass VIT ein virtuelles Objekt für ein bestimmtes Gebiet ist, und darauf kleben sich mehrere Poster, welche die Informationen über dieses Gebiet anbieten. Das VIT-System innerhalb des Spatial Model Server wurde schon realisiert und kann man jeder Zeit darauf zugreifen.

Aber es gab noch kein BTS-Model im Spatial Model Server in NEXUS vor meiner Arbeit. D.h. ich musste selbst ein Model für BTS definieren und eine Menge BTS-Objekte in eine sogenannte Zellen-Datenbank speichern.

Base Transceiver Station
*mcc : Mobile Country Code >> 1583
*mnc : Mobile Network Code >> 624
*lac : Location Area Code >> 50905
*ci : Cell ID >> 9044
*id : mcc + mnc + lac >> 158362450505
*fcn : Channel Number >> 650
*id-Liste(je mcc, mnc, lac): Eine ID Liste für alle benachbarte BTS >> 1583, 624, 9029, 1583, 624, 29508, ...
*lati : Latitude >> 48.78402784
*longi : Longitude >> 9.23837881
*alti : Altitude >> 300.0
geodetic : Geodätische Daten >> 100
power : Transmitter Power >> 0.0
range ser: Serving Range >> 0.0
range nei : Neighbour Range >> 0.0
description : Beschreibung >> Wangen Stadtmitte A
op-from : Gültigkeit seit >> 13.3.01
op-to : Gültigkeit bis >> null

Tabelle 6-1 : Das BTS-Model

Die Zellen-Datenbank ist ein wichtiger Teil von dem Positionierungssystem. Weil durch Mobiltelefon erhalten wir nur die Kanalnummer von den benachbarten BTSs und das Zelle-ID von der bedienenden BTS. Diese reichen noch nicht für unsere Positionsberechnung. Wir müssen mehr Informationen wie die genaue Position der jeweiligen BTS haben. Deshalb müssen wir eine Datenbank über BTS herstellen, um die gebrauchten Informationen anzubieten.

Das Model für BTS ist ganz einfach, wir haben es schon im Kapitel 1 beschrieben. Für die Berechnung sind die Attribute nicht alle wichtig, d.h. nicht alle Attribute werden für die Berechnung benutzt, aber eine vollständige Datenbank über alle Daten von Funkzellen innerhalb eines Netzwerks zu speichern ist immer sinnvoll, vielleicht in der Zukunft werden einige Informationen für andere Anwendungen wichtig sein. In Tabelle 6-1 ist ein Überblick für das BTS-Model. Die Attribute mit Stern Zeichen sind die Daten, was wir für die Berechnung brauchen. Für jedes Attribut gibt's ein Beispiel. Diese Datenbank wurde durch DB2 im Spatial Model Server realisiert.

6.2.3 Datenflüsse

Wir haben vier die gebrauchten Elemente für die Vorbereitungsphase vorgestellt, nämlich das Mobiltelefon, das Palm-Gerät, der Zwischenserver und der Spatial Model Server. Abbildung 6-6 zeigt die Datenflüsse zwischen diese Elemente.

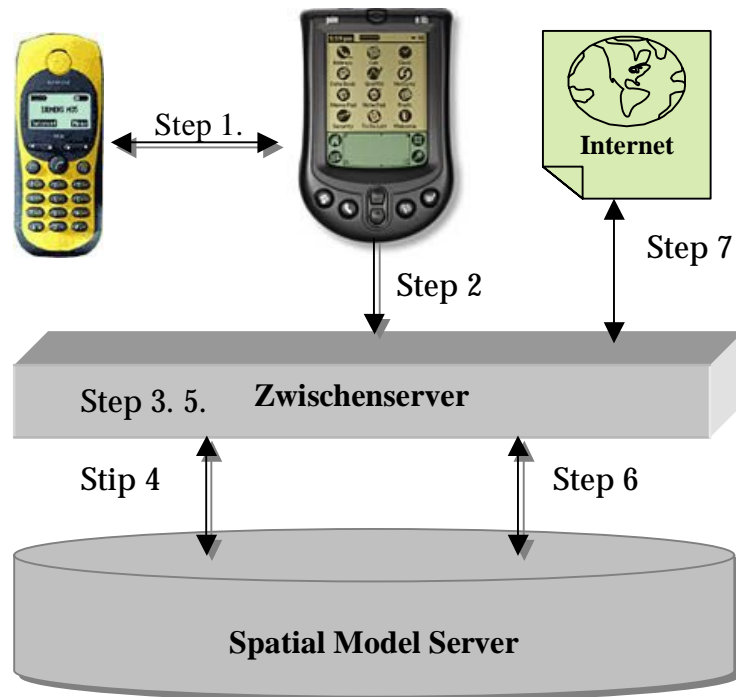


Abbildung 6-6: Datenflüssen zwischen die Elemente

Step 1.

In dem ersten Schritt soll das Palm-Gerät die AT-Befehle durch Serial-Port nach Siemens m35 senden und das Messungsergebnis zurückbekommen.

Um die Feldstärken und die Zelle-Information aus dem Siemens m35 zu erhalten, müssen die SIM-Toolkit Funktionen aufgrund [ETSI1114 1999] verwendet werden. Diese Funktionen werden durch den Befehl `AT^SSTK=a,b` aufgerufen, wobei a die Länge von der folgenden PDU-Nachricht ist, und b entscheidet, ob eine oder mehrere Funktionen benutzt werden sollen. Die PDU-Nachricht ist eine hexe Zeichenkette für die von Palm angeforderte Funktionen, und es gibt zwei Funktionen, die wir brauchen, eine heißt „Local Information“, welche die MCC, MNC, LAC und CI von der bedienenden BTS anbietet; die andere heißt „Network Measurement Result“, welche die Kanal-Nummer und die Feldstärken von den benachbarten BTS anbietet. Hier liegt ein Beispiel für eine „Network Measurement Result“ Anfrage.

```
AT^SSTK=21,0  
> D009810301260282028182
```


alle Anfragen durch AWQL dargestellt, so müssen wir noch alle Anfragen nach der AWQL formalisieren.

Im Abschnitt 2.2.2 wurden die Datenflüsse zwischen Anwendung und Zellen-Datenbank schon beschrieben. Das erste Kontakt zum Spatial Model Server ist, mit dem Zelle-ID von der bedienenden BTS der Datenbank nach den Zelle-ID von allen benachbarten BTS anzufragen. Die Anfrage sieht wie Folgende aus:

```
<awql>
<restriction>
  <and>
    <equal>
      <attr name="type"/>
      <nexusdata><Table>BaseTranceiverStation</Table></nexusdata>
    </equal>
    <equal>
      <attr name="id"/>
      <nexusdata>158362450505</nexusdata>
    </equal>
  </and>
</restriction>
<filter>
  <include>
    <attr name="id-Liste"/>
    <attr name="fcn"/>
  </include>
  <excludeallother/>
</filter>
</awql>
```

Die Funktion *query* auf der Schnittstelle vom Spatial Model Server dann verarbeitet diese Frage aufgrund der Zellen-Datenbank, um die *id-Liste* und *fcn* von der bedienenden BTS zu ermitteln. Die Antwort ist ein ermitteltes BTS-Objekt mit drei Attributen, und wird in AWML geschrieben und zu dem Zwischenserver zurückgesendet. Es sieht auch ganz einfach aus:

```
<awml>
<dataobject type="BaseTranceiverStation">
  <id>158632480505</id>
  <id-Liste>156685156, ... </id-Liste>
  <fcn>635</fcn>
</dataobject>
</awml>
```

Die folgenden Anfragen und Antworten werden gleich wie die obigen beide Beispiele formalisiert. Dann nach ein paar Male Fragen haben wir endlich alle Daten, was wir brauchen zur Berechnung.

Step 5.

Bei dem Schritt 5 wird der Positionierungsalgorithmus durchgeführt. Wir haben das im Kapitel 1 schon ganz genau erzählt. Das Ergebnis ist ein Punkt, der durch Latitude und Longitude dargestellt wird.

Step 6.

Mit dem Ergebnis aus Schritt 5 könne wir jetzt nach den ortsbezogenen VITs wieder bei dem Spatial Model Server fragen. Die beide Latitude und Longitude Werte sollen natürlich in den Fragen eingestellt werden. Wir formalisieren die Anfrage wie Folgende:

```
<awql>
<restriction>
  <and>
    <equal>
      <attr name="type"/>
      <nexusdata><Table>VIT</Table></nexusdata>
    </equal>
    <overlaps>
      <attr name="visibility"/>
      <nexusdata><WKT>'POINT (9.17499900 48.78255900)'<</WKT></nexusdata>
    </overlaps>
  </and>
</restriction>
</awql>
```

In dieser Frage gibt's ein Element „overlaps“, dadurch wird der Spatial Model Server versuchen, um alle gespeicherte VIT, deren Attribut „visibility“ sich mit der im Element „WKT“ angegebenen Position überlappt, zu finden, und schreibt die in einer AWMML-Datei. D.h. wir könnten eine Liste von VIT erhalten, wie Folgende zeigt:

```
<awml>
<dataobject type="VIT" NOL="nexus://nexus://asstupro/49/13" kind="VIRTUAL">
  <NAME>Pragsattel</NAME>
  <EXTENT>
</EXTENT>
  <POS>POINT ( 9.17616100 48.81787600)</POS>
  <HEIGHT>
</HEIGHT>
  <ALTITUDE>
</ALTITUDE>
  <ADDRESS>
</ADDRESS>
  <OWNER>ifp</OWNER>
  <STARTPAGEADDRESS>http://vit.informatik.uni-
stuttgart.de/demo/vilis/city/pragsattel/pragsattel.xml</STARTPAGEADDRESS>
```

```

    <ACTIVEPLISTEADDRESS>http://vit.informatik.uni-
stuttgart.de/demo/vilis/city/pragsattel/pragsattel_active.xml</ACTIVEPLISTEADDRESS>
    <PHIERARCHYADDRESS>http://vit.informatik.uni-
stuttgart.de/demo/vilis/city/pragsattel/pragsattel_hierarchy.xml</PHIERARCHYADDRESS
>
    <EXPIRES>2005-12-31</EXPIRES>
    <VISIBILITY>MULTIPOLYGON ((( 11.17616100 48.81787600, 10.59037500
50.23209000, 9.17616100 50.81787600, 7.76194800 50.23209000, 7.17616100 48.81787600,
7.76194800 47.40366300, 9.17616100 46.81787600, 10.59037500 47.40366300,
11.17616100 48.81787600)))</VISIBILITY>
    </dataobject>

... weitere VIT-Objekte ...

</awml>

```

In dieser AWML-Datei befindet sich eine Liste für mehrere VITs, für jede VIT gibt es ein paar Attribute. Das Attribut „VISIBILITY“ ist für die Sichtbarkeit der jeweiligen VIT zuständig, und der Spatial Model Server vergleicht dieses Attribut mit der aktuellen Position, um zu entscheiden, ob eine VIT sichtbar für den Benutzer oder nicht.

Step 7.

Vielleicht haben wir bemerkt, dass in obiger AWML-Datei vorhandene VIT-Informationen interessieren uns überhaupt nicht, z.B. die Sichtbarkeit, die URL Adresse usw.. Es ist unnötig, solche Informationen auf der PDA zu zeigen. Aber die URL-Adresse im Attribut „ACTIVEPLISTEADDRESS“ ist wichtig für unsere Anwendung, da diese Adresse die eigentliche VIT-Beschreibung zeigt. D.h. wir sollten durch diese Adresse wieder auf eine XML-Dokument zugreifen, um weitere Informationen zu erfahren.

Mit dieser URL-Adresse dann brauchen wir nicht mehr noch mal mit dem Spatial Model Server zu kommunizieren, sondern direkt auf Internet zu suchen. Als Ergebnis sollten wir ein XML Dokument wie Folgende bekommen:

```

<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE ACTIVELISTE PUBLIC "-//VIT//DTD VIT 1.0 Strict //EN"
"http://vit.informatik.uni-stuttgart.de/xml/dtd/vit.dtd">
<ACTIVELISTE>
    <ACTIVE-POSTER>
        <NAME>Stuttgart 21 </NAME>
        <ADDRESS>http://www.stuttgart21.de</ADDRESS>
        <AUTHOR>www.stuttgart21.de</AUTHOR>
        <VISIBILITY>
            <PRISM>
                <CIRCLE>
                    <POSITION TYPE="WGS84">48.783283N,
9.182254E</POSITION>

```

```

        <RADIUS>
            <DISTANCE
UNIT="METER">200</DISTANCE>
            </RADIUS>
        </CIRCLE>
    </PRISM>
    </VISIBILITY>
</ACTIVE-POSTER>

... weitere ACTIVE-POSTER ...

</ACTIVELISTE>

```

Dieses XML-Dokument kann man als eine originale Beschreibung für jeweilige VIT verstehen. Wir beobachten, dass es in diesem Dokument wieder eine Liste über mehrere Active-Poster gibt. Solche Active-Posters sind nämlich die Posters, die auf der VIT geklebt wurden. Für jeden Poster gibt's das Attribut „ADDRESS“ wieder mit einer URL-Adresse, die endlich den Inhalt von jeweiligem Poster hinweisen soll.

Bis jetzt soll die Vorbereitungsphase endlich fertig sein. Wenn alles klappt, dann sollten wir eine Liste von VITs bekommen, und für jede VIT haben wir noch eine Liste von Active-Posters, und jeder Active-Poster hat einen Hinweis zu seinem Inhalt, hier normale Weise ist der Inhalt eine HTML-Datei. Wir stellen jetzt der Browsephase alle solche Informationen zur Verfügung.

6.3 Die Browsephase

Die Browsephase ist offensichtlich nicht so kompliziert wie die Vorbereitungsphase, weil es nicht so viele Arbeiten gibt. Die Grundaufgabe in der Browsephase ist, die ortsbezogene Informationen aus Internet nehmen, und auf dem Palm-Gerät anzuzeigen.

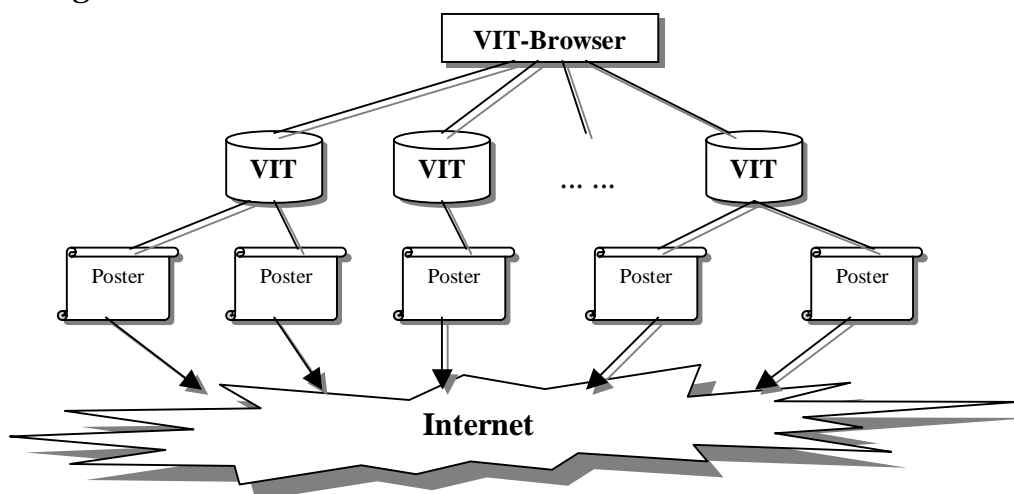


Abbildung 6-7: Die Architektur über die VIT-Informationen

Am Ende der Vorbereitungsphase haben wir die ganze Struktur von den VIT-Informationen, nämlich die sichtbare VITs und ihre Active-Poster. Hier sind die Poster die Hinweise, welche uns die Adressen zu den im Internet bleibenden Daten-Quellen zeigen. Diese Struktur(siehe Abbildung 6-7) bleibt am Anfang der Browsephase in dem Speicher vom Zwischenserver, dann wird von der Anwendung auf Palm OS abgeholt, und zeigt sich auf dem Bildschirm vom Palm-Gerät. Durch diese Struktur kann der Benutzer sehr intuitiv einen schönen Eindruck über die Umgebung von der aktuellen Position haben. Der Benutzer braucht nur der interessante VIT und Poster(siehe Abbildung 6-8) auszuwählen, dann tauchen die gewünschten ortsbezogenen Informationen sofort auf dem Bildschirm auf. Der Benutzer kann natürlich die Posters wechseln und sogar die VIT wechseln, um andere Informationen zu kucken.

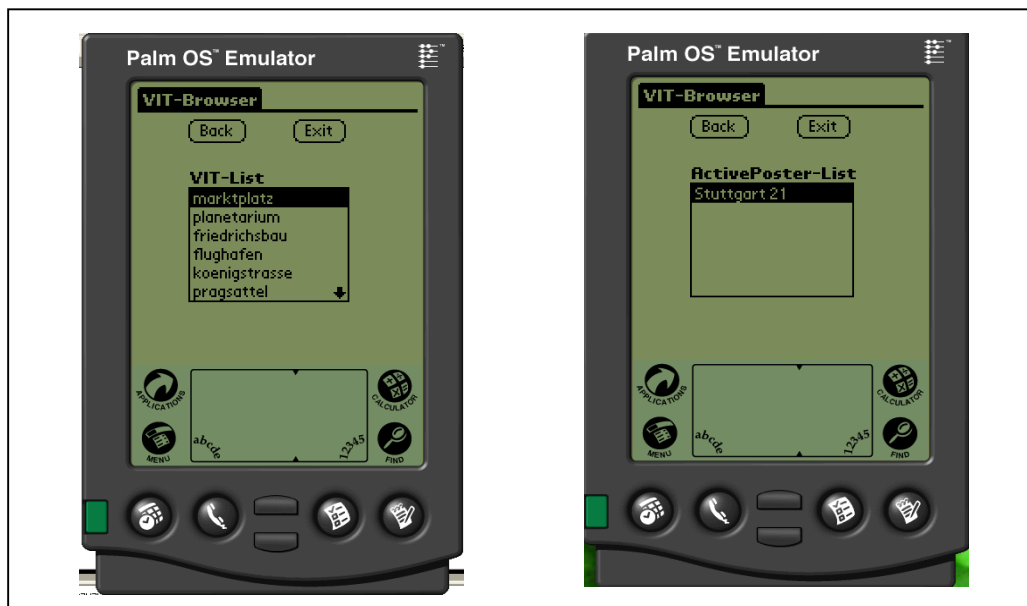


Abbildung 6-8: VIT-Liste und Poster-Liste auf dem Palmbildschirm

6.3.1 Der Arbeitsablauf

Nach der Vorbereitungsphase stellt der Zwischenserver diese VIT-Struktur für PDA zur Verfügung. Auf der PDA-Seite wird ein Anwendungsprogramm entwickelt, die zeigt diese Struktur und endliche gewünschte HTML-Dateien auf dem PDA-Bildschirm an. Abbildung 6-9 zeigt die Datenflüsse in dieser Phase. Um solche Aufgaben zu realisieren, muss das PDA-Gerät ständig mit dem Zwischenserver kommunizieren, da der Benutzer immer neue VIT und Poster ganz frei auswählen können. Deswegen ist es nötig, ein einfaches Protokoll zwischen PDA und Zwischenserver entwerfen, um die Nachrichten auszutauschen.

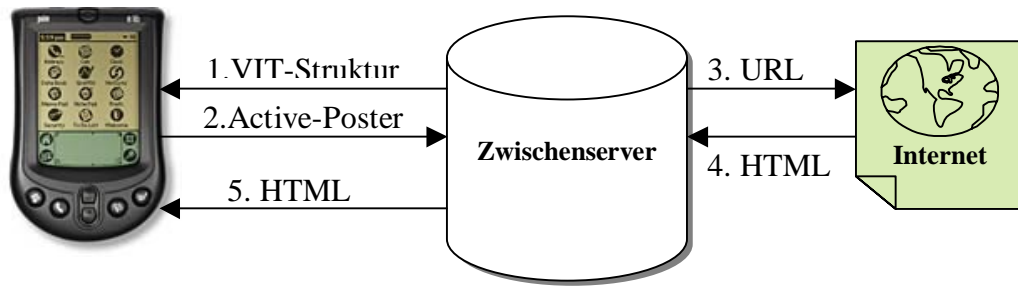


Abbildung 6-9: Die Datenflüsse in der Browsephase

In der Browsephase gibt's keine neue Technologie einzusetzen, das Mobiltelefon wird nur als Zugang zum Netzwerk gebraucht, und die Verbindung wird auch durch Sockets-Technologie wie vorher realisiert, aber der Datenaustausch basiert nicht mehr auf XML, sondern auf ein selbstdefiniertes Protokoll. Das Spatial Model Server brauchen wir nicht mehr, da der Zwischenserver direkt auf Internet zugreifen wird, um die HTML-Dateien zu bekommen.

Dieses selbst definierte Protokoll ist einfach zu machen. Zuerst wird eine Liste von sichtbaren VITs auf dem PDA-Bildschirm angezeigt, der Benutzer wählt einen VIT aus und der PDA sendet diesen VIT an den Zwischenserver. Aufgrund diesem VIT wird wieder eine Liste von zugehörigen Active-Postern von dem Zwischenserver zurückgeschickt und auch auf dem Bildschirm für den Benutzer zur Verfügung gestellt. Wenn der Benutzer kein interessantes Active-poster ausfindet, dann kann er zurück zu der VIT-Liste gehen um einen neuen VIT zu wählen; oder wählt der Benutzer ein interessantes Active-Poster aus, und sendet es wieder an den Zwischenserver. Als Antwort bekommt der PDA eine HTML-Datei, und der PDA zeigt endlich die wichtige Elemente von der HTML-Datei an, nämlich ein einfaches Browser. Nach dem Lesen kann der Benutzer diese Anwendung direkt verlassen oder wieder zu der VIT-Liste gehen.

Auf der Zwischenserver-Seite wartet es immer auf den verschiedenen Anfragen aus PDA und sendet jeweilige nötige Informationen zurück. Wenn eine Anfrage nach den VITs kommt, dann liefert Zwischenserver eine Menge sichtbare VITs zurück; Wenn eine ausgewählte VIT kommt, dann werden eine Menge zugehörige Active-Poster zurückgeschickt; Wenn ein ausgewählter Active-Poster kommt, im Zwischenserver verfügt jedes Active-Poster eine URL-Adress, dadurch können wir die gewünschte ortsbezogene Information, nämlich die HTML-Datei aus dem Internet abholen, und sofort an das PDA zurück schicken.

Wenn der Benutzer die ortsbezogenen Informationen von einem anderen Ort lesen, d.h. der Benutzer möchte die aktuelle Position ändern, dann kann er die Browsephase verlassen und zurück zu dem Anfang der Vorbereitungsphase gehen. Dazwischen wird die Verbindung zum Netzwerk automatisch vorläufig beendet, um die Batterie vom PDA und die Kosten(Mobile Anrufstarif) zu sparen. Der Zwischenserver verlässt die Browsephase gleichzeitig und vorbereitet für eine neue Positionierungsberechnung. Wenn der Benutzer die Vorbereitungsphase noch mal initialisiert, dann messt der PDA die neuen Signale aus Mobilnetz und baut wieder eine Verbindung zum Zwischenserver.

6.3.2 HTML oder WML?

Die von Active-Poster gezeigte Dateien, die wir über Internet erhalten sollten, sind normale Weise beliebige HTML-Dateien. Und meine alle letzte Aufgabe ist, solche ankommende HTML-Dateien auf dem Bildschirm vom Palm anzuzeigen, nämlich ein kleines Browser zu realisieren. Das klingt vielleicht einfach, aber ist tatsächlich ganz schwer.

6.3.2.1 Das Problem für HTML auf PDA

HTML(Hypertext Markup Language) ist die Mutter aller Web-Seiten. HTML hat den Vorteil, dass sich die in den Web-Seiten enthaltenen Informationen durch das Prinzip des logischen Markup automatisch an die Fenstergröße des Client-Rechners anpassen, egal ob es sich um einen Notebook oder PC mit großem oder kleinem, niedrig oder hoch auflösendem Bildschirm handelt. Aber diese automatische Anpassung gilt nur innerhalb eines Bereiches von wenigstens einigermaßen vergleichbaren Fenstergrößen, d.h. der Bildschirm darf nicht zu klein sein. Und um die Web-Seite immer schöner und zauberhafter zu machen, sind die HTML-Dateien auch immer komplizierter, dass manchmal die Quelltexte wie eine Katastrophe aussehen.

Für Erzeugung von HTML-Datei vielleicht ist nicht so schrecklich, in den meisten Fällen wird man die HTML-Dateien nicht händisch erstellen, sondern automatisch und dynamisch aus den in einer Datenbank oder einem Workflow-System gespeicherten Informationen generieren. Und normale Weise berücksichtigen die Autoren überhaupt nicht, ob die Dateien geeignet für klein Gerät ist. Wenn wir eine beliebige vorhandene HTML auf einem PDA-Bildschirm zu zeigen, dann ist es ganz anders wie auf dem PC-Bildschirm.

Das Haupt Problem ist nämlich das Parsen. Die HTML-Datei hat keine strenge Syntax, z.B. die Groß- und Klein-Schreibung ist egal, die Argumente müssen nicht unbedingt in Anführungszeichen eingeschlossen, den End-Tag kann man weglassen usw.. Deshalb sind die HTML-Dateien gar nicht so einfach wie XML-Dateien zu parsen, und bringt immer riesige Probleme, wenn wir sie parsen. Außerdem ist die Verarbeitung für HTML-Parsen auch sehr aufwendig, und kann die Geschwindigkeit von der ganzen Anwendung verringern. Ein anderes Problem ist, dass im Moment die HTML-Dateien viele Informationselemente(Tags) enthalten, die nicht geeignet auf PDA anzuzeigen sind, z.B. Grafik, Animation oder sogar Video usw. Solche Element zeigen manchmal wichtige Informationen was wir brauchen, aber leider können wir die nicht auf PDA-Bildschirm zeigen. Deswegen sind HTML-Dateien im Moment noch nicht geeignet für PDA-Gerät.

Ich habe auch versucht, einen schon fertiggestellten HTML-Browser in meiner Arbeit einzusetzen, d.h. ich übernehme einfach den Quellecode aus dem HTML-Browser und stecke den in meine Anwendung. Aber leider hat es auch nicht geklappt, weil solche Browser normale Weise geschäftliche Anwendungen sind, d.h. man kann

vielleicht die Anwendung einfach frei herunterladen, aber auf den Quellcode darf man nicht zugreifen wegen geschäftlichen Geheimnissen.

Ein anderes Problem ist, dass im Moment die HTML-Dateien viele Informationselemente(Tags) enthalten, die nicht geeignet auf PDA anzuzeigen sind, z.B. Grafik, Animation oder sogar Video usw. Solche Elementen enthalten manchmal wichtige Informationen was wir brauchen, aber leider können wir die nicht auf PDA-Bildschirm zeigen. Deswegen sind HTML-Dateien als Informationsquelle im Moment noch nicht sehr geeignet für PDA-Gerät.

Deshalb konnte ich nur einige einfache Elemente in einer HTML-Datei auf dem PDA-Bildschirm zeigen(siehe Abbildung 6-10), und die meiste Elemente musste ich verzichten, weil es zu schwer ist, um die richtige Argumente herauszunehmen.

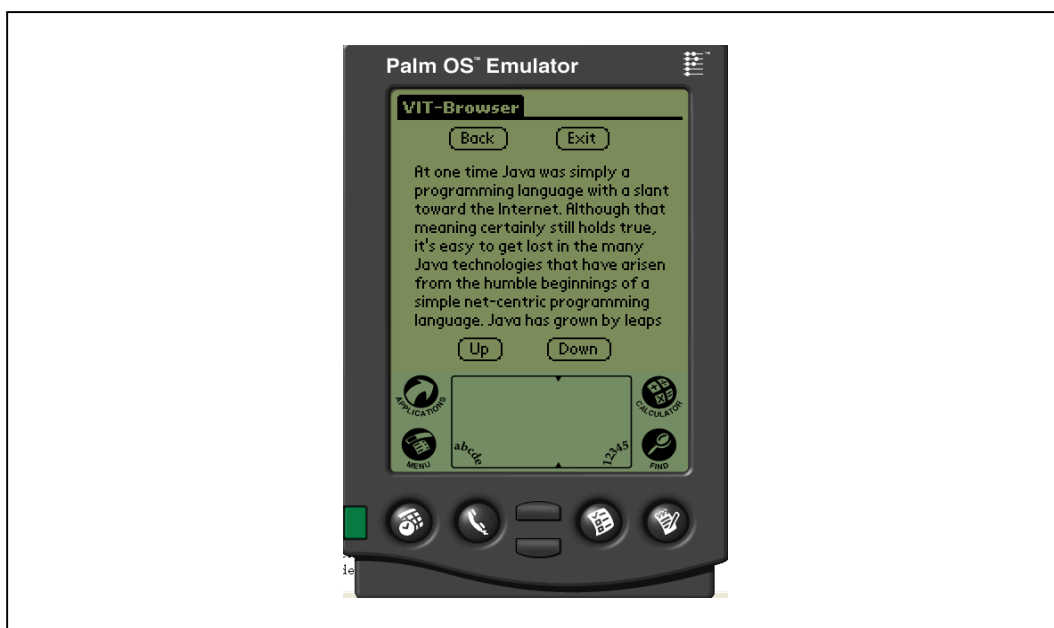


Abbildung 6-10: Ein einfacher Text über Marktplatz

6.3.2.2 Was ist WML?

Weil es zu kompliziert ist, HTML als eine Sprache zu benutzen, um die Inhalte auf dem PDA-Bildschirm anzuzeigen. Dann vielleicht können wir eine Alternativ Sprache finden, die kleinen Geräten gerade passt.

Um die Informationen auf kleinem Bildschirm zu zeigen, muss man die Informationen in allgemeinen extra aufbereiten, in einem stark gekürzten und wesentlich kompakteren Format. Da solche Geräte meistens auch nur über geringere Rechenleitungen verfügen, müssen außerdem ein Format und ein Protokoll verwendet werden, die möglichst effizient verarbeitet werden können.

WML scheint wie eine Lösung zu sein. WML(Wireless Markup Languag) ist das Format, damit Hypertext-Informationen für Geräte mit kleinen Bildschirmen definiert werden können, z.B. für Mobiltelefon, PDA usw.. WML erfüllt eine ähnliche

Funktion wie HTML, und kann man als eine stark vereinfachte Version von HTML verstehen[DevGuru 2002].

WML basiert nicht direkt auf HTML sondern auf XML, deshalb hat WML auch strengere Regeln als HTML, z.B. in WML muss man WML-Tags und Attribute immer mit Kleinbuchstaben schreiben, der End-Tag darf man nicht weglassen, jedes Argument muss sich innerhalb eines Elements befinden, die WML-Tags müssen immer richtig geschachtelt, usw.. So erlaubt WML dadurch eine effizientere Verarbeitung.

Ein einfaches WML-Dokument sieht wie Folgende aus:

```
<wml>
  <card>
    <p>
      ... Text ...
    </p>
  </card>
</wml>
```

Jedes WML Dokument muss zwischen den Tag <wml> und </wml> eingeschlossen werden, und kann eine oder mehrere „card“ enthalten, welche die Einheit sind, die nach einander am Bildschirm angezeigt werden. Jede card entspricht also einer Bildschirm-Seite.

Jede card kann wieder ein oder mehrere Text-Absätze enthalten, die zwischen den Tag<p> und </p> eingeschlossen werden. Die Texte können auch Hyperlinks und Bilder enthalten.

Ein Absatz <p> kann folgende Elemente wie HTML enthalten, aber muss strenge Regel wie XML folgen.

```
<p>
  ... Text ...
  <em> ... </em> <b> ... </b> <i> ... </i>
  <br />
  <table> ... </table>
  <a> ... </a>
  <img ... />
  ... ..
</p>
```

Außerdem müssen die WML-Dateien so geschrieben werden, dass sie von der Client Hardware und Software-version unabhängig sind, deshalb verwendet WML dazu wie HTML das Prinzip des logischen Markups, und wir sollten nur die Elemente verwenden, die in der WML-Form festgelegt sind und die von allen Clients richtig dargestellt werden.

Jetzt können wir feststellen, dass WML viel besser als HTML ist, auf PDA-Gerät anzuzeigen, weil WML Datei einfach zu verarbeiten ist, und die Informationen in WML sind auch für kleine Geräte entworfen und passen dem Palmbildschirm sehr gut.

Wenn wir die WML als unsere Datenquelle nehmen, dann können wir noch einfach ein WAP-Browser direkt benutzen, um die WML-Dateien zu zeigen. Der WAP-Browser befindet sich in den mobilen Geräten und ist für die Darstellung der WML Dateien zuständig. Zur Zeit kann es noch nur eine reine textuelle Darstellung der Datei erfüllen.

WAP-Browser funktioniert nur mit dem WAP. WAP(Wireless Application Protocol) ist ein besonders effiziente Protokoll, und ermöglicht die Informationen aus dem Internet auf mobilen Geräten darzustellen[WAP 2002]. Im allgemeinen wird WAP zwischen dem Handy und einem WAP-Gateway verwendet, und die Übertragung zwischen dem Web-Server und dem WAP-Gateway erfolgt über das Internet mit dem Http-Protokoll, d.h. dieses Gateway ist die Vermittlungsstelle zwischen dem PDA und dem Internet, und unser Zwischenserver kann gerade diese Rolle übernehmen.

6.3.2.3 Einschränkung von WML

Aus technischen Aspekten ist WML perfekt als die Datenquelle in dem Internet für unseres VIT-Browser. Aber das Problem ist, das VIT-Browser ist eine ortsbezogene Anwendung, und alle Web-Seiten können als Datenquelle für das Browser sein, d.h. der Web-Server muss alle Informationen in zwei Versionen schreiben, eine komplette Version für PC-Rechnern als HTML-Datei und eine gekürzte Version für kleine Geräte. Das ist im Moment offenbar unmöglich. Deshalb ist die Anwendung von WML noch nicht praktisch, und kann nur im kleinen Bereich realisieren, z.B. im Mobiltelefon Bereich.

6.4 Zusammenfassung

Im diesen Kapital haben wir die ganze Realisierung von der VIT-Browser Anwendung vorgestellt. Die Ausführung für diese Anwendung wird in zwei Teile eingeteilt, eine ist die Vorbereitungsphase, und die andere ist die Browsephase. In der Vorbereitungsphase wird die Position des Benutzers festgestellt, und aufgrund der Position werden die sichtbare VITs für den Benutzer ausgewählt. Dazwischen spielt der Zwischenserver eine wichtige Rolle, da er die Position berechnen und mit dem Spatial Model Server kommunizieren muss. Außerdem ist die Messung für Zellen-Signale auch ein Schwerpunkt.

Jede sichtbare VIT hat mehrere Active-Poster, um die Datenquelle im Internet hinzuweisen. In der Browsephase holen wir die Datenquelle aus dem Internet aus und versuchen, die auf PDA-Bildschirm anzuzeigen. Aber leider kann kleines Gerät die beliebige HTML-Dateien nicht gut unterstützen wegen der Komplexität von HTML-Dateien. Als Alternative haben wir WML und WAP untergesucht, und aus

technischem Aspekt ist WML viel besser als HTML, in unserer Anwendung zu verwenden, aber leider wird WML zur Zeit noch nicht breit unterstützt.

7 Zusammenfassung und Ausblick

Zur Zeit werden die kleine und tragbare Geräte wie z.B. Mobiltelefon, PDA immer häufiger benutzt. Wegen der Tragbarkeit können wir fast überall jeder Zeit solche kleine Geräte benutzen, damit werden die ortsbezogenen Anwendungen langsam von uns berücksichtigt. D.h. Wir sind nicht mehr zufrieden mit den originale Funktionen von kleinen Geräten(z.B. Telefonieren, Datenspeicherung usw.), und möchten aufgrund den vorhandenen Gräten und Technologien mehrere modernere Anwendungen entwickeln.

Um die ortsbezogene Anwendungen gut zu unterstützen, wurde das NEXUS-Projekt von dem Institut für Parallele und Verteilte Höchstleistungsrechner(IPVR) der Universität Stuttgart entwickelt. Das sogenannte NEXUS-System ist nämlich eine offene Plattform, die sich besonders für ortsbezogene Anwendungen zur Verfügung stellt. Die Grundidee ist, dass alle physische oder virtuelle Objekte in der physischen Welt in einer Augmented World modelliert werden, um eine einheitliche Schnittstelle für alle NEXUS-Anwendungen anzubieten, damit werden alle unterschiedliche Anwendungen zusammen in diesem System integriert werden. Durch die NEXUS Anwendungsinterface können die Client alle Informationen aufgrund der aktuellen Position erhalten.

In meiner Arbeit wurde auch eine NEXUS-Anwendung entwickelt, nämlich das VIT-Browser für PDA. Das ist eine absolute ortsbezogene Anwendung, da der Benutzer von dieser Anwendung durch ein Mobiltelefon und ein PDA jeder Zeit und überall(mindestens in Stuttgart-Vaihingen) die Informationen über die aktuelle Umgebung erfahren sollte, z.B. das Krankenhaus oder der Kaufhof in der ganzen Näh von dem Benutzer.

Um so ein Ziel zu erreichen, sollten wir zunächst die aktuelle Position des Benutzers feststellen, und dann aufgrund dieser Position die entsprechenden Informationen aus Internet ausholen. Diese Anwendung wurde durch ein paar Bauelemente realisiert, nämlich ein Siemens m35 Handy, ein Palm m105 Gerät und der Spatial Model Server innerhalb der NEXUS-Plattform. Weil die kleinen Geräte wie PDA geringe Speicherung und niedrige Leistungsgeschwindigkeit haben, dann habe ich einen sogenannte Zwischenserver zwischen den Spatial Model Server und Palm-Gerät eingesetzt, um die meiste Datenaustausch und die Positionsberechnung zu übernehmen.

Die Positionierungsverfahren basiert auf GSM(Global System for Mobile Telecommunication), deshalb sollten wir das Handy benutzen, um die Signale aus dem mobilen Funknetzwerk zu erhalten. Außerdem ist das Handy noch für die Verbindung zwischen Palm-Gerät und Internet zuständig. Das Palm-Gerät verbindet sich direkt mit dem Handy und kommuniziert dadurch mit dem Zwischenserver.

Der Zwischenserver ist grundsätzlich für den Datenaustausch zu dem Spatial Model Server und aufwendige Berechnung zuständig. Die Verbindung zum Spatial Model Server wird durch SOAP realisiert. Da die Daten-Format innerhalb des NEXUS-System durch die beide auf XML basierende AWQL und AWML dargestellt werden,

muss der Zwischenserver noch die JDOM-Technologie benutzen, um die XML-Dateien zu parsen.

Die Hauptaufgabe für Palm-Gerät ist, die HTML-Dateien aus dem Internet zu erhalten und auf dem Palmbildschirm zu zeigen. Wegen der höheren Komplexität von HTML-Dateien können die Palm-Geräte leider nur ein paar einfache Elemente anzeigen. Dafür ist WML aus technischem Aspekt eine gute Alternative. Es gibt zwei Programmierungssprache für Palm OS, nämlich C und J2ME, in meiner Anwendung wurde C genutzt, da J2ME die Serial-Kommuniktion nicht unterstützt.

Das ganze System von meiner Anwendung wegen den Einschränkungen von den mobilen Geräten sieht ziemlich kompliziert aus, und es ist schon ein bisschen komisch, wenn man eine Hand mit dem Handy und die andere Hand mit PDA steht, auf die Information zu warten. Deshalb wäre es ideal, wenn die Funktionen vom Handy und Palm-Gerät auf einem einzigen Gerät zusammengebaut werden, dann kann man sehr leichter benutzen, und das wurde tatsächlich schon realisiert bei Sony und Nokia, aber der Preis ist ziemlich hoch. Aber in der Zukunft wird es sowieso immer billiger.

WML ist eine gute Alternative gegen HTML für meine Anwendung, aber leider wird die noch nicht breit unterstützt, d.h. bei meisten Web-Seiten wird keine WML Datei besonders für kleine Geräte erzeugt. Ich glaube diese Situation wird immer so bleiben, weil die Hardware sich ganz schnell entwickelt, wir können uns vorstellen, in der Zukunft können die Palm-Geräte HTML-Dateien auch problemlos auf dem Bildschirm mit starker Leistung anzeigen.

Literaturverzeichnis

[Akif 2001]

Akif, Mohammad
Java XML Programmer's Reference
Wrox press Ltd, July 2001
ISBN: 1861005202

[Apache SOAP 2001]

Dokumentation über Apache SOAP Version 2.2
<http://xml.apache.org/soap/>
Apache XML Projekt, 2001

[DevGuru 2002]

DevGuru WML Introduction
http://www.devguru.com/Technologies/wml/quickref/wml_intro.html

[ETSI1114 1999]

Technical Specification Digital cellular telecommunication system;
Specification of the SIM Application Toolkit
GSM 11.14 version 8.3.0 Release 1999,
ETSI TS 101267 V8.3.0

[ETSI0408 2000]

Technical Specification Digital cellular telecommunication system;
Mobile radio interface layer 3 specification
GSM 04.04 version 7.8.0 Release 1998,
ETSI TS 100940 V7.8.0

[Großmann 2001]

Schwarz, Thomas; Nicklas, Daniela; Großmann, Matthias; Volz, Steffen
Information Management and Exchange in Nexus
Technical Report, Research Group Nexus, Universität Stuttgart, 2001

[Grzan 2002]

Stjepan, Grzan
Enabling technology for an indoor location aware information system
Diplomarbeit Nr. 1958, Fakultät Informatik, Universität Stuttgart, 2002

[Hatler FAQ]

Halter, Wade
Pilot Programming FAQ
<http://www.wademan.com/Pilot/Program/FAQ.htm>

[Howlett 1997]

Howlett, Andrew
GNU PILOT SDK TUTORIAL
Free Software Foundation, Inc., 1997

[Java & JDOM 2001]

Jason Hunter
JDOM: How It Works, and How It Opened the Java Process
O'Reilly Open Source Convention 2001

[JDOM 2001]

JDOM Spezifikation und API
<http://www.jdom.org>

[Knudsen 2001]

Knudsen, Jonathan
The Big Small Platform
http://java.oreilly.com/news/j2me_0201.html

[Morrison 2001]

Morrison, Michael
Wireless Java with J2ME
Sams, 2001
ISBN: 0-672-32142-4

[McKeehan 1999]

Rhodes, Neil & McKeehan, Julie
Palm Programming
O'Reilly & Associates, Inc., 1999
ISBN: 1565925254

[Meßmer 2001]

Meßmer, Jens
Modellierung der Augmented World in Nexus
Diplomarbeit Nr.1870, Fakultät Informatik, Universität Stuttgart, 2001

[M35 2000]

**Siemens AG : Manual Reference, AT Command Set
for the SIEMENS Mobiltelefons s35, m35, c35 2000**
GSM 07.07, GSM07.05, Siemens specific commands

[Nicklas 2001]

Nicklas, Daniela; Großmann, Matthias; Schwarz, Thomas; Volz, Steffen;
Mitschang Bernhard
**A Model-Based, Open Architecture for Mobile, Spatially Aware
Applications**

[Palm 3.0]

**Developing Palm OS
3.0 Applications Part III**
<http://www.palm.com>

[Prctools 2000]

**PRC-Tools Projekt
GCC for Palm OS**
<http://sourceforge.net/projects/prc-tools>

[Seybold 2001]

Seybold, Christian
**Positioning of Mobile Devices using Network Information from GSM
Devices**
Diplomarbeit Nr.1905, Fakultät Informatik, Universität Stuttgart, 2001

[WAP 2002]

WAP Tutorial
<http://www.w3schools.com/wap/default.asp>

[Winton 2001]

Winton, Greg
Berkeley on the Palm
http://palmos.oreilly.com/news/palmosnetwork_0801.html

