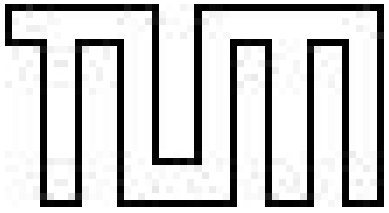

Technische Universität München

Fakultät für Informatik

Forschungs- und Lehreinheit Informatik XIII

DESIGN UND REALISIERUNG VON TERMINAL-BASIERTER POSITIONSBESTIMMUNG AUF BASIS VON SIM APPLICATION TOOLKIT FUNKTIONEN

DESIGN AND REALIZATION OF TERMINAL BASED POSITIONING BASED ON SIM APPLICATION TOOLKIT FUNCTIONS

Bachelor Thesis

Michael Dyrna

Aufgabensteller:	Prof. Dr. Uwe Baumgarten
Betreuer:	Prof. Dr. Uwe Baumgarten Dipl. Inf. Christian Hillebrand
Abgabetermin:	15. Juli 2002

Zusammenfassung / Abstract

Das Projekt COSMOS (Community Online Services and Mobile Solutions) der Technischen Universität München und der Universität Hohenheim beschäftigt sich mit der Entwicklung generischer Dienstleistungskonzepte für den Betrieb vom mobilen Communities. Eine Community ist eine Gemeinschaft von Menschen, die aufgrund gleicher Interessen miteinander interagieren und kommunizieren.

Das Teilprojekt „Client- und Endgerätetechnologien für mobile Community-Support-Systeme“ entwickelt technische Lösungen, um die Teilnahme an einer Community zu jeder Zeit und an jedem Ort mittels mobiler Endgeräte zu unterstützen.

Um einer mobilen Community auch ortsbasierte Dienste anzubieten, ist eine Positionsbestimmung des mobilen Endgeräts sowie die Verwaltung der dadurch gewonnenen Daten auf einem Hintergrundsystem notwendig.

Die Aufgabenstellung dieser Bachelor Thesis besteht zum einen darin, eine auf der SIM-Karte des Mobiltelefons laufende Software zu entwickeln, die die zur Positionsbestimmung notwendigen und im Telefon bereitgestellten Daten unter Verwendung von sogenannten *SIM Application Toolkit* Funktionen an ein Hintergrundsystem übermittelt. Zum anderen umfasst sie die Entwicklung des Hintergrundsystems, das aus den übermittelten Daten die Position berechnet und in einem Datenbanksystem verwaltet.

Inhalt

Kapitel 1

Grundlagen

Das erste Kapitel beschreibt die technischen Hintergründe, die zum detaillierten Verständnis des in den Kapiteln 2 und 3 beschriebenen Systems nötig sind.

Abschnitt 1.1 führt den Begriff des „ortsbezogenen Dienstes“ ein und zeigt anhand einiger Szenarien auf, wozu mobile Communities solche Dienste nutzen können. Abschnitt 1.2 definiert den Begriff „terminal-basierte Positionsbestimmung“ und grenzt ihn zu anderen Verfahren ab. Ein kurzer Einstieg in GSM findet sich in Abschnitt 1.3. Da die GSM-Spezifikation mehrere Zigtausend Seiten umfasst, muss sich diese Arbeit auf die für das bearbeitete Thema relevanten Aspekte beschränken. Den im Mobiltelefon verfügbaren Daten, die zur Bestimmung des Aufenthaltsorts nützlich sind, ist ein eigener Abschnitt 1.4 gewidmet. Abschnitt 1.5 gibt einen Überblick über die beiden Nachrichtenübertragungsdienste in GSM, SMS (*short message service*) und USSD (*unstructured supplementary service data*). In Abschnitt 1.7 schließlich wird die *SIM Application Toolkit* Spezifikation detailliert vorgestellt, wobei auch hier darauf geachtet wurde, den Bezug zu dem im zweiten und dritten Kapitel beschriebenen System herzustellen und nur die dafür relevanten Informationen darzustellen.

1.1 Ortsbezogene Dienste für mobile Communities

Eine *virtuelle Community* ist eine Gruppe von Gleichgesinnten, die sich mit Hilfe der Kommunikationsdienste des Internets zusammenschließen und Informationen austauschen. Eine *mobile Community* ist eine Weiterentwicklung, bei der mobile Datendienste und Endgeräte zum Einsatz kommen (vgl. [cosmos]). Die Mobilität wird in der Regel durch drahtlose Netze ermöglicht, beispielsweise ein GSM-basiertes Mobilfunknetz.

Eines der Hauptziele bei der Entwicklung eines Systems zur Unterstützung von mobilen Communities ist die Ortstransparenz. Das bedeutet, dass jedes Mitglied einer solchen Community mit jedem anderen Mitglied in (virtuellen) Kontakt treten kann, unabhängig von seinem eigenen Aufenthaltsort und ohne den Aufenthaltsort des anderen zu kennen.

Es gibt jedoch auch Szenarien, bei denen die (geographische) Position der Mitglieder eine wichtige Rolle spielt, und damit verbunden wünschenswerte Dienste, die gerade auf deren Kenntnis basieren und somit überhaupt nicht ortstransparent sind.

Ein System zur Unterstützung von mobilen Communities, das den Aufenthaltsort seiner Mitglieder bestimmen kann, wäre zum einen in der Lage, ihnen klassische ortsabhängige Informations-Dienste (*location based information services*) anzubieten. Das Mitglied könnte als *Pull*-Dienst Informationen über die nächstgelegene Tankstelle oder Arztpraxis, das nächste Einkaufszentrum oder die nächste „Party Location“ anfordern. Bei *Push*-Diensten würden ihm ortsabhängige Informationen ohne Aufforderung zugesandt, wie beispielsweise die Sonderangebote eines Ladens, am dem es gerade vorbeigeht, oder das aktuelle Veranstaltungsprogramm eines Theaters oder Kinos, vor dem es steht.

Zum anderen könnte das System community-spezifische Dienste anbieten, die auf der Lokalisierung der Mitglieder gegenseitig basieren. Das wäre beispielsweise die Auflistung aller Mitglieder, die sich in einer bestimmten Entfernung von einem selbst aufhalten, oder das Senden einer Nachricht an alle Mitglieder, die sich an einem bestimmten Ort aufhalten, über den man zeitnah Informationen erhalten möchte.

Wenn Dienste angeboten werden, bei denen Mitglieder gegenseitig ihren Aufenthaltsort abfragen können, stellen sich besondere Anforderungen an die Sicherheit, insbesondere die Wahrung der Privatsphäre. So sollte es jedem Mitglied möglich sein, zu jeder Zeit Einfluss darauf zu nehmen, ob andere dessen Position abfragen dürfen und wer sie abfragen darf. Ein Fehlen dieser Möglichkeit würde die Teilnehmer in der Benutzung hemmen und (zu Recht) misstrauisch machen.

1.2 Terminal-basierte Positionsbestimmung

Die meisten mobilen Netze sind nicht dafür vorgesehen, Endgeräte zu lokalisieren. Auch GSM bietet eine solche Möglichkeit nicht explizit. Dennoch haben sowohl Endgerät (Terminal) als auch das Netz betriebsbedingt Zugriff auf bestimmte Informationen, die mit etwas zusätzlichem Wissen auf die ungefähre Position des Endgeräts schließen lassen.

Das Netz muss zum Beispiel zu jedem Zeitpunkt wissen, in welchem ungefähren Gebiet sich ein Endgerät befindet, wenn es bei einem eingehenden Anruf nicht einen *broadcast* über das gesamte Netz senden möchte. Schon deshalb muss es den ungefähren Aufenthaltsort jedes Teilnehmers zentral speichern. Werden diese Informationen zur Lokalisierung durch eine Anwendung benutzt, so spricht man von **netz-basierter Positionsbestimmung**.

Mit zusätzlichem Wissen kann auch das Endgerät auf seine Position schließen. Ihm ist zum Beispiel zu jedem Zeitpunkt der Mobilfunkmast bekannt, bei dem es gerade eingebucht ist. Da das Endgerät auch mitverantwortlich dafür ist, den richtigen Zeitpunkt für ein Umbuchen zu einem stärkeren Masten zu finden, hat es sogar eingeschränkte Informationen darüber, welche weiteren Mobilfunkmasten gerade in empfangbarer Entfernung sind. Wenn eine Anwendung die im Endgerät vorhandenen Daten zur Lokalisierung benutzt, spricht man von **terminal-basierter Positionsbestimmung**, die Gegenstand dieser Arbeit ist.

1.3 GSM

1.3.1 Überblick

Bereits 1982 begannen die Planungen für GSM, ein europaweit einheitliches System zur drahtlosen digitalen Übertragung von Sprache und Daten, das einer großen Teilnehmerzahl zur Verfügung steht. Inzwischen wurden auch in den USA und weiten Teilen Asiens GSM-Netze aufgebaut.

Einige wichtige Leistungsmerkmale sind die flächendeckende Nutzbarkeit in Europa, die Kompatibilität zu ISDN, eine Standardisierung europäischer Datendienste, eine Zugangsberechtigung durch eine Chip-Karte (SIM, vgl. Abschnitt 1.6) und Geheimzahl (PIN), die

Verschleierung des Aufenthaltsortes vor Unbefugten und ein ausgeglichener Anbieter-Markt ohne Monopolisierung.

Die fünf Anbieter in Deutschland, T-Mobile und D2 Vodafone auf dem Frequenzband um 900 MHz (GSM900), E-Plus, O_2 (ehemals VIAG Interkom) und Quam auf dem Frequenzband um 1,8 GHz (GSM1800) bedienen derzeit rund 54 Millionen Kunden mit ihren Träger-, Standard-, Zusatz- und Mehrwertdiensten.

Trägerdienste (Übermittlungsdienste) umfassen die leitungs- und paketvermittelten, synchronen und asynchronen Datendienste. Standarddienste (Teledienste) umfassen die Dienste der Sprachübertragung, *Short Message Service* (SMS, vgl. Abschnitt 1.5.1), E-Mail- und Fax-Übertragung. Mit Zusatzdiensten ist zum Beispiel Rufnummernübermittlung, Anrufweiterschaltung und Gesprächsmanagement gemeint. Mehrwertdienste sind Dienste wie Hotline oder Nachrichten-Dienste, die auf den genannten Träger- und Standarddienste aufsetzen.

1.3.2 Zellularität

GSM gehört zur Klasse der zellularen Mobilfunknetze. Aufgrund der sehr begrenzten Frequenzbänder steht nur eine geringe Zahl von Gesprächskanälen zur Verfügung. Um trotzdem sehr viele Teilnehmer bedienen zu können, müssen die Frequenzen räumlich mehrfach genutzt werden. Dies führte zur Entwicklung der Zellulartechnik, mit der die Frequenzen bestmöglich ausgenutzt werden sollen.

Das abzudeckende Gebiet wird dabei in Zellen aufgeteilt. Eine Zelle ist eine geographische Fläche, innerhalb der Geräte drahtlos miteinander kommunizieren können, also eine Art „Funkzone“. Ein Endgerät ist in GSM immer in genau einer Zelle eingebucht, die als *serving cell* bezeichnet wird.

Die Zellen werden zur einfacheren Handhabung als Sechsecke modelliert. Jede Zelle erhält eine Untermenge von Frequenzen aus der verfügbaren Gesamtmenge zugewiesen. Zwei benachbarte Zellen dürfen nicht dieselben Frequenzen verwenden, da es sonst zu Interferenzen kommen kann. Beim Übergang von einer Zelle zur nächsten erfolgt bei laufendem Gespräch ein Frequenzwechsel (*Handover*), so dass eine Verbindung auch über Zellgrenzen hinweg aufrecht erhalten wird.

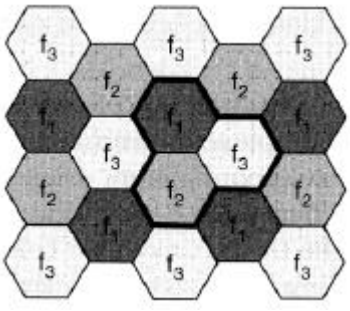

Abbildung 1: Aufteilung einer Fläche in Zellen:
Benachbarte Zellen nutzen unterschiedliche Frequenzen.

Jede Zelle hat eine ausgezeichnete Station, die Basisstation, zu der jede *mobile station*, also jedes Endgerät, eine Verbindung aufrechterhält. Zwischen den Endgeräten in einer Zelle besteht untereinander keine Verbindung. Die Basisstationen, die sich im Modell in der Mitte der Zelle

befinden, sind über eine (schnellere) Hintergrundinfrastruktur, beispielsweise hochratige Richtfunkverbindungen, miteinander und mit anderen Netzen verbunden.

In der Praxis spannt eine Basisstation (*Base Transceiver Station*, BTS) jedoch mittels gerichteter Antennen bis zu drei Zellen um sich herum auf und steht somit nicht deren Mitte, sondern an einer Ecke jeder Zelle. Das spart dem Netzbetreiber beträchtliche Kosten, weil so die Zahl der benötigten Basisstationen weit unter der Anzahl der Zellen bleibt.

1.3.3 Netzarchitektur

Abbildung 2 zeigt die Architektur von GSM, die aus einem *Switching System*, dem *Base Station System* und den *Mobile Stations* besteht.

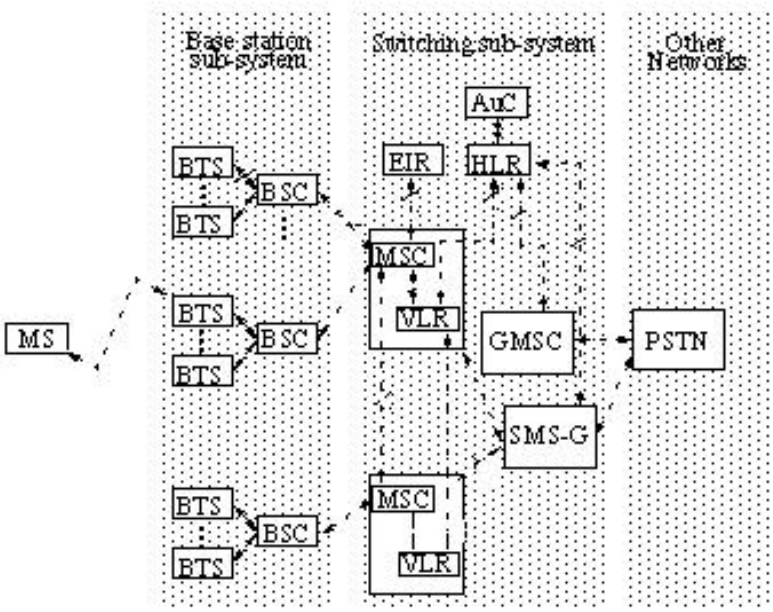

Abbildung 2: GSM-Architektur

Die funktionalen Einheiten werden im folgenden kurz beschrieben:

MS *Mobile Station*. Das Gerät, das vom Mobilfunkteilnehmer für den Dienstzugang genutzt wird. Es besteht aus zwei wesentlichen Komponenten: dem Gerät selbst (*Mobile Equipment*, ME) und dem *Subscriber Identity Module* (SIM, vgl. Abschnitt 1.6).

BTS *Base Transceiver Station*. Die BTS umfasst die Einrichtung zur Funkübertragung und ist zuständig für die Signalverarbeitung an der Luftschnittstelle.

BSC *Base Station Controller*. Der BSC verwaltet mehrere BTS. Das umfasst hauptsächlich die Zuweisung, Rücknahme und Übergabe von Funkkanälen. Hier wird außerdem das *Handover*-Protokoll abgewickelt.

MSC *Mobile Switching Centre*. Das MSC verwaltet die Verbindungen, leitet Daten weiter und regelt den Übergang zu den Festnetzen.

VLR *Visitor Location Register*. Das VLR enthält alle Teilnehmerdaten, sowohl temporäre wie permanente, die notwendig sind, um eine MS im Empfangsgebiet eines MSC zu verwalten. Das VLR ist i. d. R. Bestandteil des MSC.

AuC *Authenitification Centre*. Die AuC-Datenbank enthält die notwendigen Teilnehmer-Schlüssel und die Algorithmen zur Berechnung der Authentifikations-Parameter, die an das HLR gesendet werden.

HLR *Home Location Register*. Die HLR-Datenbank enthält permanente und halb-permanente Teilnehmerdaten. Somit weiß das HLR immer, in welcher *Location Area* sich die MS aufhält.

EIR *Equipment Identity Register*. Die EIR-Datenbank enthält Informationen über die MS und deren Fähigkeiten.

GMSC *Gateway Mobile Switching Centre*. Zum GMSC werden alle Anrufe an MS zuerst geleitet. Das GMSC erfragt den Aufenthaltsort vom HLR und leitet den Anruf an das zuständige MSC weiter.

SMS-G *Short Messages Service Gateway*. Das SMS-G (auch: *Short Message Service Centre*, SMSC) leitet Kurznachrichten von und an MS weiter (vgl. Abschnitt 1.5.1)

Räumlich ist ein GSM-Netz hierarchisch gegliedert: Es besteht aus mindestens einer Verwaltungsregion, die einem MSC untersteht. Jede dieser Verwaltungsregionen besteht aus mindestens einer *Location Area* (LA). Eine LA setzt sich aus mehreren Gruppen von Zellen zusammen. Je LA ist also mindestens ein BSC vorhanden, der eine gewisse Anzahl von BTS steuert. Die Gliederung ist in der folgenden Abbildung 3 dargestellt.

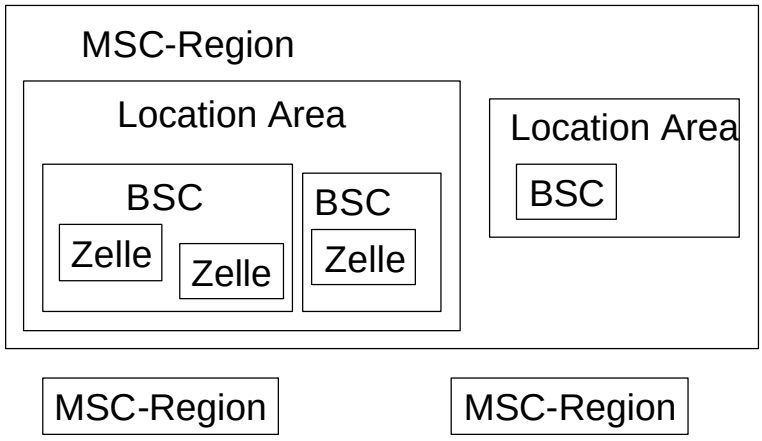

Abbildung 3: Hierarchische Struktur eines GSM-Netzes

1.4 Zur Lokalisierung nützliche Daten in GSM

Wie in Abschnitt 1.2 bereits erwähnt, sind im Mobiltelefon Daten gespeichert, aus denen man auf seine (geographische) Position schließen kann. Welche das genau sind, wird in diesem Abschnitt erläutert.

Jedes GSM-Netz wird anhand eines dreistelligen **Ländercodes** (*Mobile Country Code*, MCC) und eines zweistelligen (bei GSM1900 dreistelligen) **Netzcodes** (*Mobile Network Code*, MNC) identifiziert. Der *Country Code* von Deutschland ist beispielsweise 262. Die Netzcodes der fünf GSM-Netze in Deutschland sind in Tabelle 1 dargestellt.

Netz	Network Code
T-Mobile (D1)	01
D2 Vodafone	02
E-Plus und Quam	03
O_2 (ehemals VIAG Interkom)	07

Tabelle 1: Mobile Network Codes der deutschen Mobilfunknetze

Folgende Attribute kennzeichnen eine Zelle:

- **Location Area Code** (LAC)
  Mehrere Zellen werden in GSM zu einer *Location Area* zusammengefasst (vgl. Abschnitt 1.3.2). Der *Location Area Code* (LAC) ist 5 Dezimalstellen lang und netzweit eindeutig. Zusammen mit dem *Country Code* und dem *Network Code* ergibt er die international eindeutige *Location Area Identification* (LAI).

- **Cell Identifier** (CID)
  Eine Zelle wird anhand ihres *Cell Identifiers* (CID) referenziert. Der *Cell Identifier* ist maximal 16 Bit lang und innerhalb der *Location Area* eindeutig. Zusammen mit der *Location Area Identification* ergibt sich die international eindeutige *Global Cell Identity*.

- **Kanal**
  Die beiden Frequenzbänder um 900 MHz und um 1800 MHz unterteilt GSM jeweils in 124 bzw. 372 Teilfrequenzbänder (physische Kanäle), von denen jeder durch Zeitmultiplex wiederum in acht logische Kanäle unterteilt wird.
  Ein physischer Kanal wird anhand einer maximal drei Dezimalstellen langen Nummer bezeichnet. Die Verteilung der Kanalnummern auf die Mobilfunkbetreiber in Deutschland ist in der folgenden Tabelle angegeben.

Betreiber	GSM900	GSM1800	Kanäle
T-Mobile (D1)	13-49, 81-102, 122-124	587-611	62+25
D2 Vodafone	1-12, 50-80, 103-121	725-751	62+25
E-Plus und Quam		752-863	112
O_2		612-723	112

Tabelle 2: Kanalzuteilung in Deutschland

  Ein Kanal kann zwar nicht von zwei benachbarten Zellen gleichzeitig benutzt werden, weil es sonst zu Interferenzen käme; weiter voneinander entfernte Zellen verwenden aber durchaus dieselben Kanäle, so dass eine Zelle nicht allein durch die Kanalnummer ihres *Broadcast Control Channel* (BCCH) identifiziert werden kann.

Das Zeitmultiplex-Verfahren bei GSM arbeitet mit so kurzen Zeitschlitzen, dass ein Mobiltelefon seine Datenpakete derart zeitversetzt absenden muss, dass sie im jeweils richtigen *time slot* bei der BTS ankommen. Die Anzahl der Bitzeiten, um die das Signal zeitlich verschoben werden muss, heißt **Timing Advance** und ist ein direktes Maß für die Entfernung zwischen Mobiltelefon und BTS. Ein Unterschied von einer Bitzeit (48/13 µs) entspricht einem Entfernungsunterschied von etwa 550 Metern.

Der *Timing Advance* ist als 6-Bit-Wert gespeichert.

Das Mobiltelefon misst regelmäßig die **Empfangsstärke** (*Reception Level*, RXLEV) und **Bitfehlerrate** (*Reception Quality*, RXQUAL) des BCCH von bis zu sechs benachbarten Zellen.

Diese Werte werden für den Entscheidungsalgorithmus des *Handover* benutzt. (*Handover* bezeichnet das Umbuchen von einer Zelle in eine andere.)

Die Empfangsstärke gibt die Leistung des empfangenen Signals an. Da – zumindest im Vakuum – die Leistung proportional zur der Entfernung zwischen Sender und Empfänger abnimmt, ist sie auch unter dem Einfluss von Störungen (Luft, Witterung, Bebauung) zumindest ein ungefähres Maß für die Entfernung zwischen Endgerät und benachbarter Basisstation und somit nützlich für die Positionsbestimmung. Die Bitfehlerrate lässt dagegen keine verwertbaren Rückschlüsse auf die Distanz zu.

Der Empfangspegel wird im Mobiltelefon in 64 Stufen diskretisiert und steht für die Berechnung somit als 6-Bit-Wert (RXLEV) zur Verfügung.

Als einziger Mobilfunkbetreiber in Deutschland überträgt O_2 (ehemals VIAG Interkom) die **geographische Position** der aktuellen Zelle als Gauß-Krüger-Koordinaten. Diese Information überträgt jede BTS mittels *Cell Broadcast* (vgl. Abschnitt 1.7.6) an alle bei ihr eingebuchten Teilnehmer.

Eine Gauß-Krüger-Koordinate ist eine 14-stellige Zahl, die aus einem siebenstelligen Rechtswert und einem siebenstelligen Hochwert besteht. Der Rechtswert teilt sich wie folgt auf: Die erste Ziffer ist ein Vielfaches vom dritten Grad östlicher Länge; die restlichen sechs Ziffern bezeichnen auf den Meter genau die Entfernung östlich von dem in der ersten Ziffer bezeichneten Längengrad. Der Hochwert bestimmt auf den Meter genau den Abstand vom Äquator.

Im O_2-Netz werden Gauß-Krüger-Werte übermittelt, die 12 Stellen lang sind und somit die Position nur mit einer Genauigkeit von zehn Metern in jeder Dimension angibt.

1.5 Nachrichtenübertragungsdienste in GSM

Zur Übertragung von kleinen Datenmengen, für die keine eigene Gesprächs-Verbindung aufgebaut werden soll, stehen in GSM zwei Dienste zur Verfügung: *Short Message Service* (SMS) und *Unstructured Supplementary Service Data* (USSD). Diese beiden werden in den folgenden zwei Abschnitten eingeführt.

1.5.1 Short Message Service (SMS)

SMS (*Short Message Service*) ist ein Übertragungs-Dienst für kurze Nachrichten zwischen zwei Endgeräten (vgl. [gsm03.40]). Die Übertragung geschieht nach dem *Store-and-Forward*-Prinzip über *SMS Centres* (SMSC).

Das bedeutet, dass Sender und Empfänger keine Verbindung zueinander aufbauen, sondern dass die Nachricht zunächst vollständig dem zuständigen SMSC des Absenders übergeben wird. Dieses SMSC routet sie dann über weitere SMSCs des eigenen Netzes und ggf. auch anderer Netze, bis sie beim SMSC angekommen ist, das für den Empfänger zuständig ist. Dieses SMSC übergibt sie dann vollständig dem Empfänger. Die Weg-Information erhält das SMSC vom HLR.

Der *Short Message Service* stützt sich auf zwei Basis-Dienste:

- *Short Message Mobile Terminated Point-to-Point* (SM MT) liefert eine Kurzmitteilung an ein Endgerät aus (*delivery*) und stellt Informationen über die Zustellung bereit (*deliver report*).
- *Short Message Mobile Originated Point-to-Point* (SM MO) überträgt eine Kurzmitteilung von einem Endgerät an das GSM-Netz (*submission*) und stellt Informationen über die Weiterleitung bereit (*submit report*).

Im ersten Fall heißt das übertragene Datenpaket SMS-DELIVER-TPDU bzw. SMS-DELIVER-REPORT-TPDU; im zweiten Fall SMS-SUBMIT-TPDU bzw. SMS-SUBMIT-REPORT-TPDU. Ihr Aufbau ist sehr ähnlich.

Exemplarisch soll im folgenden der Aufbau einer SMS-SUBMIT-TPDU, also eines Datenpakets, das ein Endgerät an ein SMSC überträgt, aufgezeigt werden.

Name	**Länge**	**Beschreibung**
TP-Message-Type-Indicator	2 Bit	hier: SMS-SUBMIT-TPDU
TP-Reject-Duplicates	1 Bit	1 bedeutet, dass das SMSC die Nachricht verwerfen soll, wenn schon eine mit gleicher *Message Reference* von demselben Absender vorliegt
TP-Validity-Period-Format	2 Bit	gibt an, ob in der TPDU ein *Validity Period* Feld vorhanden ist und wie es ggf. codiert ist
TP-Reply-Path	1 Bit	siehe [gsm03.40]
TP-User-Data-Header-Indicator	1 Bit	1 bedeutet, dass die *User Data* einen Header enthält
TP-Status-Report-Request	1 Bit	1 bedeutet, dass die MS einen SUBMIT REPORT zugestellt bekommen möchte
TP-Message-Reference	1 Byte	eindeutiger Identifikator, mit dem die Kurzmitteilung referenziert werden kann
TP-Destination-Address	2-12 B	Adresse (MSISDN) des Empfängers
TP-Protocol-Identifyer	1 Byte	Art der Daten, die in der Kurzmitteilung transportiert werden soll (siehe unten)
TP-Data-Coding-Scheme	1 Byte	Codierung der *User Data* (z.B. 7-, 8- oder 16-Bit) (s.u.)
TP-Validity-Period	1-7 Byte	Gültigkeitsdauer der Nachricht; d.h. die Nachricht soll nach der angegebenen Zeit verworfen werden, falls sie nicht zugestellt werden konnte.
TP-User-Data-Length	1 Byte	Länge der *User Data*
TP-User-Data	0-140 B	zu transportierende Daten

Tabelle 3: Aufbau einer SMS-SUBMIT-TPDU

Anhand des **TP-Protocol-Identifier** unterscheidet das Mobiltelefon u.a., ob es sich um eine „normale“ Kurzmitteilung handelt, die dem Benutzer angezeigt wird, oder ob die Nachricht für eine SIM-Anwendung bestimmt ist.

TP-User-Data kann bis zu 140 Oktette (Bytes à 8 Bit) lang sein. Welche Information damit dargestellt wird, gibt **TP-Data-Coding-Scheme** an.

Für herkömmliche Kurzmitteilungen, also solche die Text enthalten, wird 7-Bit-Codierung verwendet. Damit lassen sich 160 Textzeichen in den 140 Oktetten unterbringen.

Binäre Daten (zum Beispiel Logos, Klingeltöne) liegen in der Regel im 8-Bit-Format vor und werden auch mit dieser Codierung übertragen.

Zur Zeit noch wenig üblich ist die Codierung von Texten mit 16 Bit pro Zeichen, mit der sich auch andere als lateinische Schriftzeichen darstellen lassen. In eine Kurzmitteilung würden so nur 70 solcher Zeichen passen. Diese Codierung wird momentan für sogenannte *Flash SMS*

genutzt. Das sind Kurzmitteilungen, die dem Empfänger unmittelbar im Display angezeigt werden.

TP-Data-Coding-Scheme ist für den Transport der dargestellten Information unerheblich und wird erst vom Endgerät ausgewertet.

1.5.2 Unstructured Supplementary Service Data (USSD)

Es gibt zwei Modi von USSD (*Unstructured Supplementary Service Data*): *MMI-mode* und *application mode.*

MMI-mode USSD transportiert Strings zwischen Benutzer und Mobilfunknetz. Der Benutzer gibt die Strings, die er an das Netz senden möchte, über die Tastatur ein. Strings, die das Netz an den Benutzer sendet, werden auf dem Display angezeigt.

Mit *application mode USSD* wird der Transport von (binären) Daten zwischen dem Mobilfunknetz und dem mobilem Endgerät ermöglicht. Es ist dafür gedacht, dass Anwendungen im Mobilfunknetz mit Anwendungen auf dem Endgerät Daten austauschen können. Deshalb ist es für diese Arbeit sehr interessant, denn über USSD wäre es zum einen möglich, dem Endgerät mitzuteilen, eine Übertragung der Orts-Information anzustoßen, und zum anderen könnte darüber auch die Übertragung der Orts-Informationen an das Hintergrund-System erfolgen.

Die Kommunikation über die Luftschnittstelle findet über die Signalisierungskanäle statt. Die maximal erzielbare Datenrate liegt bei 600 bis 1000 Bits pro Sekunde. Damit eignet es sich ausschließlich für die Übertragung von sehr kleinen Datenmengen, wie Steuer- und Status-informationen.

Empfängt ein MSC USSD von einer MS, routet es die Daten an das VLR weiter. Wenn das VLR die Daten nicht an das adressierte Ziel weiterleiten kann, routet es sie an das HLR weiter.

USSD-Übertragungen finden nur innerhalb des Mobilfunknetzes statt. Wenn eine externe Applikation – zum Beispiel über das Internet – Daten mit einer Applikation auf dem Endgerät austauschen möchte, ist dafür ein *Gateway* notwendig, das die Daten umwandelt und umadressiert.

Die MS werden wie gewohnt über ihre MSISDN adressiert. Eine externe Adressierung ist für USSD nicht spezifiziert und muss von den Mobilfunkprovidern proprietär gelöst werden. (Beispielsweise könnte eine IP-Adresse oder Anbieter-ID den Daten vorangestellt werden, die das Gateway interpretiert und zur externen Adressierung nutzt.)

1.6 SIM

Die *Mobile Station* (MS) in GSM besteht aus dem *Mobile Equipment* (ME) und dem *Subscriber Identity Module* (SIM), die über eine serielle Schnittstelle miteinander kommunizieren (vgl. Abschnitt 1.3.3). Die Steuerung geht dabei immer vom ME aus. In der Kommunikationbeziehung wird daher das ME als *Master* bezeichnet, das SIM als *Slave*.

Beim SIM handelt es sich um eine Chipkarte im ID-000-Format (vgl. [chip]), die in erster Linie die Aufgabe hat, den Zugang zum Mobilfunknetz nur berechtigten Personen zu gewähren und eine funktionsfähige Gebührenabrechnung zu schaffen.

Die SIM-Karte ist nicht nur eine einfache Speicherkarte, sondern verfügt neben einem EEPROM der Größe 8, 16 oder 32 Kilobyte auch über einen Mikroprozessor und eine Schnittstelle zum ME. Das ME kann also nicht direkt auf den Speicher des SIM zugreifen, sondern kommuniziert ausschließlich über die in [gsm11.11] spezifizierten Befehle mit der SIM-Karte.

Dadurch ist sie eine mächtige Komponente, die unter anderem für die Erfüllung der folgenden Aufgaben zuständig ist:

- Prüfung der Zugangsberechtigung: Nach der Initialisierung des SIM muss sich der Benutzer über die Tastatur des ME gegenüber dem SIM anhand einer PIN authentifizieren. Das SIM blockiert nach dreimaliger Falscheingabe und kann nur durch die Eingabe einer speziellen (längeren) Entsperr-Nummer wieder entriegelt werden.
- Authentifikation: Auf dem SIM ist der geheime Schlüssel gespeichert, mit dem das Mobilfunknetz in einem *Challenge-Response*-Verfahren überprüft, ob eine MS zum Einbuchen berechtigt ist. Die Authentifikation ist einseitig, d.h. das SIM kann die „Echtheit“ der BTS, bei der es sich einbucht, – also ob es sich tatsächlich um eine BTS des vermuteten Anbieters handelt – nicht überprüfen.
- Verschlüsselung: Auf dem SIM sind die permanenten Schlüssel gespeichert, aus denen der temporäre Schlüssel errechnet wird, mit dem Sprach- und Datenverbindungen über die Luftschnittstelle verschlüsselt werden. Die eigentliche Verschlüsselung findet (im Gegensatz zu anderen Chipkarten-Systemen) im ME statt, weil der Mikroprozessor für die Echtzeitverschlüsselung nicht leistungsfähig genug ist.
- Speicherung von *Short Messages* und Telefonbucheinträgen: Das SIM speichert außerdem benutzerspezifische Daten wie zum Beispiel Kurznachrichten und Rufnummern.
- Speicherung der Teilnehmerdaten: im wesentlichen die Rufnummer (*Mobile Station ISDN Number*, MSISDN) und Teilnehmerkennung (*International Mobile Subscriber Identity*, IMSI)

Das SIM hat ein hierarchisch aufgebautes Dateisystem mit einem *Master File* (MF) und vier *Dedicated Files* (DF), in denen sich die *Elementary Files* (EF) mit den Daten der Anwendungen befinden.

Für den Zugriff auf das SIM sind in [gsm11.11] 22 Kommandos spezifiziert, die u.a. den Zugriff auf die Dateien, die Verwaltung der PIN, das Anstoßen der Verschlüsselungs-Algorithmen und die Übertragung von Daten vom und zum ME ermöglichen.

Für *SIM Application Toolkit* (SAT) sind fünf Befehle relevant:

- TERMINAL PROFILE
  Mit diesem Kommando teilt das ME dem SIM seine Funktionalität mit. Es wird während der Initialisierungsphase des SIM gesendet und enthält als Parameter ein Bitfeld, das das Vorhandensein bzw. Nichtvorhandensein jeder einzelnen Funktionalität anzeigt.
  Die Verarbeitung des TERMINAL PROFILE ist wichtig, um unvorhergesehene Zustände beim Ablauf der SIM-Anwendung auf Mobiltelefonen, die bestimmte SAT-Befehle nicht unterstützen, zu vermeiden. Falls das SIM während der Initialisierung kein TERMINAL-PROFILE-Kommando empfängt, muss es davon ausgehen, dass das ME *SIM Application Toolkit* überhaupt nicht unterstützt.
- STATUS
  Da im Kommunikationsprotokoll zwischen ME und SIM das ME als *Master* fungiert, können SAT-Befehle nicht von sich aus auf das *Mobile Equipment* zugreifen. Stattdessen sendet das ME in regelmäßigen Abständen (üblicherweise 30 Sekunden) ein STATUS-

Kommando an das SIM („Polling“). Das SIM teilt dem ME in seiner Antwort mit, ob ein *SIM Application Toolkit* Kommando ansteht oder nicht.

- FETCH
  Diese Funktion transportiert ein Kommando vom SIM zum ME. Das ME sendet das Kommando, wenn das SIM auf eine STATUS-Anfrage signalisiert hat, dass es ein SAT-Kommando ausführen möchte.
- TERMINAL RESPONSE
  Diese Funktion transportiert die Antwort auf ein vorher ausgelesenes SAT-Kommando vom ME zum SIM.
- ENVELOPE
  Diese Funktion transportiert alle Daten vom ME zum SIM, die nicht Antworten auf SAT-Kommandos sind. Das ist zum Beispiel die Benachrichtigung über einen abgelaufenen Timer oder eine eingegangene Kurzmitteilung.

1.7 SIM Application Toolkit

1.7.1 Motivation und Einführung

Im Laufe des Betriebs von GSM trat immer mehr das Bedürfnis in den Vordergrund, das SIM nicht nur für die im letzten Abschnitt genannten Aufgaben zu verwenden. Ein Mobiltelefon wäre beispielsweise ein adäquates Medium für die Abfrage des Kontostands, die Anforderung von anderen Nachrichten und Informationen, kurz: für sogenannte Mehrwertdienste. Da die technische Realisierung mit der bestehenden Funktionalität der SIM-Karte nicht möglich war, spezifizierte die ETSI 1996 in [gsm11.14] das *SIM Application Toolkit*.

SIM Application Toolkit stellt Mechanismen zur Verfügung, die es Applikationen auf der SIM-Karte erlauben, mit dem Mobiltelefon zu operieren und interagieren. Diese Mechanismen umfassen zum Beispiel die Darstellung von Text auf dem Display, das Senden einer Kurzmitteilung, den Aufbau von Sprach- und Datenverbindungen, die Darstellung von Benutzer-Dialogen und – was für diese Arbeit von zentraler Bedeutung ist – die Abfrage von Zellinformationen, mit Hilfe derer der Standort berechnet werden kann.

Der Programmcode der SIM-Anwendungen wird als Array von Befehlen fester Länge in strukturierten *Elementary Files* mit dem Attribut ‘*executable*’ auf dem EEPROM des SIM abgelegt.

1.7.2 Profile Download

Der Umfang der unterstützten SAT-Funktionen wird mit einer Klasse angegeben: I, I bis, II und III. Der Befehls-Umfang wird dem SIM bei seiner Initialisierung mittels des SIM-Kommandos TERMINAL PROFILE mitgeteilt (vgl. Abschnitt 1.6).

Die für diese Arbeit relevanten Funktionen und deren Stelle im Bitfeld des TERMINAL PROFILE sind in der folgenden Tabelle angegeben. (Das *least significant bit* trägt dabei die Nummer 1.)

Funktionalität	Byte	Bit
SMS-PP data download	1	2
Cell Broadcast data download	1	3
Menu selection	1	4
'9EXX' response code for SIM data download error	1	5
Timer expiration	1	6
Command result	2	1
Mobile originated short message control by SIM	2	4
SAT-Befehl DISPLAY TEXT	3	1
SAT-Befehl GET INPUT	3	2
SAT-Befehl SELECT ITEM	4	1
SAT-Befehl SEND SHORT MESSAGE	4	2
SAT-Befehl SEND USSD	4	4
SAT-Befehl SET UP MENU	4	6
SAT-Befehl PROVIDE LOCAL INFORMATION (Location Information)	4	7
SAT-Befehl PROVIDE LOCAL INFORMATION (Network Measurement Results)	4	8
SAT-Befehl SET UP EVENT LIST	5	1
Event: Mobile terminated call	5	2
Event: Location status	5	5
SAT-Befehl TIMER MANAGEMENT (start, stop)	8	1
SAT-Befehl TIMER MANAGEMENT (get current value)	8	2
BCCH channel list neues Codierungsschema	9	3
SAT-Befehl PROVIDE LOCAL INFORMATION (Timing Advance)	9	5

Tabelle 4: TERMINAL PROFILE

1.7.3 Syntax

1.7.3.1 BER-TLV

Befehle (vom SIM zum ME), Antworten und asynchrone Daten (vom ME zum SIM) wie etwa eine Kurzmitteilung oder die Benachrichtigung über eine vom Benutzer getroffene Menüauswahl werden als ein BER-TLV-Datenobjekt codiert. Das Aneinanderhängen von mehreren BER-TLV-Objekten in einem Befehl ist nicht zulässig.

BER steht dabei für *Basic Encoding Rules*. Gemeint ist damit ein Dokument der ASN.1-Spezifikation, in der die entsprechende Syntax spezifiziert ist. TLV für *Tag-Length-Value*, was schon grob die Syntax beschreibt:

Tag	Length	Value
(1 Byte)	(1 oder 2 Byte)	(n Byte)

Abbildung 4: Aufbau eines BER-TLV-Datenobjekts

Das *tag* ist ein eindeutiger Identifikator, der die Art des Datenobjekts angibt. Die *tags*, die für *SIM Application Toolkit* definiert sind, werden in der folgenden Tabelle aufgelistet.

Beschreibung	Tag
Proactive SIM command tag	‘D0’
SMS-PP download tag	‘D1’
Cell Broadcast Download tag	‘D2’
Menu Selection tag	‘D3’
Call control tag	‘D4’
MO Short message control tag	‘D5’
Event download tag	‘D6’
Timer expiration	‘D7’

Tabelle 5: BER-TLV tags

length gibt die Länge des Value-Teils in Bytes an. Die Werte 0 bis 127 werden direkt in einem Byte codiert. Die Werte 128-255 werden in zwei Byte codiert: Das erste Byte enthält ‘81’ und das zweite Byte die Länge. Diese Schreibweise wird von ASN.1 vorgeschrieben, um beliebige Längen zu ermöglichen. Davon wird hier allerdings kein Gebrauch gemacht, da der Value-Teils ohnehin höchstens 255 Byte umfassen darf.

Der Value-Teil ist eine Liste von SIMPLE-TLV-Datenobjekten, die im folgenden Abschnitt beschrieben werden.

1.7.3.2 SIMPLE-TLV

Ein SIMPLE-TLV-Datenobjekt ist genauso codiert wie ein BER-TLV-Datenobjekt. Allerdings können mehrere SIMPLE-TLV-Datenobjekte innerhalb des Value-Teils des BER-TLV-Datenobjekts aneinandergehängt werden:

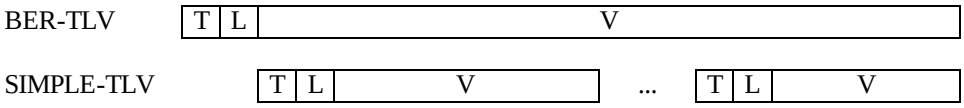

Abbildung 5: BER-TLV-Datenobjekt mit SIMPLE-TLV-Datenobjekten

Die folgende Tabelle enthält alle Tag-Werte, die für diese Arbeit relevant sind.

Beschreibung	Tag
Command details tag	‘01’ oder ‘81’
Device identity tag	‘02’ oder ‘82’
Result tag	‘03’ oder ‘83’
Alpha identifier tag	‘05’ oder ‘85’
Address Tag	‘06’ oder ‘86’
USSD string tag	‘0A’ oder ‘8A’
SMS TPDU tag	‘0B’ oder ‘8B’
Text string tag	‘0D’ oder ‘8D’
Item tag	‘0F’ oder ‘8F’
Response length tag	‘11’ oder ‘91’
Location Information tag	‘13’ oder ‘93’
Network Measurement Results tag	‘16’ oder ‘96’

Default text tag	‘17’ oder ‘97’
Items Next Action Indicator tag	‘18’
Event list tag	‘19’ oder ‘99’
BCCH channel list tag	‘1D’ oder ‘9D’
Icon identifier tag	‘1E’ oder ‘9E’
Item Icon identifier list tag	‘1F’ oder ‘9F’
Timer identifier tag	‘24’ oder ‘A4’
Timer value tag	‘25’ oder ‘A5’
Timing Advance tag	‘2E’ oder ‘AE’

Tabelle 6: SIMPLE-TLV tags

Die SIMPLE-TLV-Datenobjekte müssen bei Kommando-Aufrufen in der spezifizierten Reihenfolge übergeben werden.

1.7.4 SIM Application Toolkit Kommandos

SIM Application Toolkit spezifiziert insgesamt 31 Kommandos. Für diese Arbeit sind davon folgende relevant:

- GET INPUT
- SET UP MENU
- SELECT ITEM
- SEND SHORT MESSAGE
- SEND USSD
- PROVIDE LOCAL INFORMATION
- SET UP EVENT LIST
- TIMER MANAGEMENT

Davon werden die nicht-trivialen zusammen mit den dazugehörigen Datenobjekten in den folgenden Abschnitten genauer beschrieben. Die Syntax und Semantik aller Befehle und Datentypen sind in [gsm11.14] spezifiziert.

1.7.4.1 Datenobjekte für alle Kommandos

Die beiden Datenobjekte *Command details* und *Device identity* müssen bei jedem SAT-Kommandoaufruf als erste beiden Datenobjekte übergeben werden. Eine TERMINAL RESPONSE enthält als erste drei Datenobjekte die genannten beiden und *Result*.

(a) Command details

Byte	**Beschreibung**	**Länge**
1	Command details tag	1
2	Länge = 3	1
3	Command number	1
4	Type of command	1
5	Command Qualifier	1

Tabelle 7: Command details

Die *Command number* ist eine Referenznummer, anhand derer die SIM-Anwendung Antworten des ME den übergebenen Befehlen zuordnen kann. Das ME übernimmt das gesamte *Command details* Datenobjekt in seine TERMINAL RESPONSE.

Nach dem derzeitigen Stand der Spezifikation ist die *Command number* allerdings überflüssig, da die Funktionsaufrufe synchron erfolgen, das bedeutet, dass das ME auf jeden Aufruf unmittelbar die dazugehörige TERMINAL RESPONSE zurücksendet.

Type of command gibt an, welches SAT-Kommando ausgeführt werden soll. Die Zuordnung von für diese Arbeit relevanten Kommandos zu *Type of command* ist in der folgenden Tabelle gegeben:

Kommando	**Type of command**
SET UP EVENT LIST	‘05’
SEND USSD	‘12’
SEND SHORT MESSAGE	‘13’
GET INPUT	‘23’
SELECT ITEM	‘24’
SET UP MENU	‘25’
PROVIDE LOCAL INFORMATION	‘26’
TIMER MANAGEMENT	‘27’

Tabelle 8: Codierung *Command number*

Im *Command Qualifier* erwarten einige SAT-Kommandos Details zum Aufruf.

Bei GET INPUT beispielsweise wird in diesem Byte festgelegt, welche Zeichen eingegeben werden dürfen (nur Buchstaben, nur Ziffern, beides), wie die Eingabe codiert werden soll (7-, 8- oder 16 Bit) und ob die Eingabe auf dem Display angezeigt oder mit Sternchen verdeckt werden soll.

Bei PROVIDE LOCAL INFORMATION muss man in diesem Byte angeben, welche Daten man auslesen möchte. ‘00’ steht dabei für *Location information* (MCC, MNC, LAC, CID); ‘02´ steht für *Network Measurement Results* (also Kanäle und Signalstärke von bis zu sechs umliegenden Zellen); ‘05’ steht für *Timing Advance*.

(b) Device identity

Byte	**Beschreibung**	**Länge**
1	Device identity tag	1
2	Länge = 2	1
3	Source device identity	1
4	Destination device identity	1

Tabelle 9: Device identity

Source und *Destination device identity* sind wie folgt codiert:

- ‚01’ = Tastatur
- ‚02’ = Display
- ‚81’ = SIM
- ‚82’ = ME
- ‚83’ = Netz

In der Regel ist bei einem Befehlsaufruf das *Source device* das SIM und das *Destination device* das ME und in der TERMINAL RESPONSE umgekehrt.

(c) Result

Byte(s)	Beschreibung	Länge
1	Result tag	1
2	Länge (X)	1
3	General result	1
4..(2+X)	Additional information	X-1

Tabelle 10: Result

General result gibt an, ob das Kommando erfolgreich oder teilweise erfolgreich ausgeführt werden konnte (‘0X’ und ‘1X’) oder ob ein vorübergehendes (‘2X’) oder generelles (‘3X’) Problem vorlag.

Additional information ist kommando- und problemspezifisch.

1.7.4.2 PROVIDE LOCAL INFORMATION

Der Befehl PROVIDE LOCAL INFORMATION veranlasst das ME, aktuelle Lokalisierungsinformationen an das SIM zu senden. Dabei können folgende Werte übermittelt werden:

- *Mobile Country Code* (MCC), *Mobile Network Code* (MNC), *Location Area Code* (LAC) und *Cell ID* der aktuellen *serving cell*
- *International Mobile Equipment Number* (IMEI)
- *Network Measurement Results* und *channel list*
- aktuelles Datum, Uhrzeit, Zeitzone
- aktuelle Spracheinstellung
- *Timing Advance*

Welche Informationen davon geliefert werden sollen, wird im *Command details* Datenobjekt angegeben (vgl. Abschnitt 1.7.4.1). Außer der für alle Kommandos vorgeschrieben *Command details* und *Device identity* müssen beim Aufruf keine weiteren Datenobjekte übergeben werden.

Einige Werte der *Network Measurement Results* sind ungültig, wenn im Moment der Abfrage keine dedizierte Verbindung, d.h. keine Sprach- oder Datenverbindung, besteht.

Der Wert für *Timing Advance* wird nur bei einer Verbindung (SMS, Sprach- oder Datenverbindung) gemessen. Der übertragene Wert ist deshalb unter Umständen nicht aktuell.

Die TERMINAL RESPONSE enthält neben den für alle Kommandos vorgeschriebenen Datenobjekten *Command details*, *Device identity* und *Result* die Datenobjekte *Local Information*, *Network Measurement Results*, *BCCH channel list* bzw. *Timing Advance*, je nachdem, welche Informationen angefordert wurden. Der genaue Aufbau wird im folgenden im Detail erklärt.

(a) Local Information

Byte(s)	Beschreibung	Länge
1	Location Information tag	1
2	Länge = 7	1
3..5	Mobile Country & Network Code (MCC & MNC)	3
6..7	Location Area Code (LAC)	2
8..9	Cell Identity Value (Cell ID)	2

Tabelle 11: Local Information

MCC und MNC sind im BSC-Format (d.h. jede *digit* bezeichnet eine dezimale Ziffer) wie folgt codiert:

8	7	6	5	4	3	2	1	
MCC digit 2				MCC digit 1				octet 1
MNC digit 3				MCC digit 3				octet 2
MNC digit 2				MNC digit 1				octet 3

Tabelle 12: Mobile Country Code und Mobile Networt Code

Der LAC und die *Cell ID* sind *most significant bit first* codiert.

(b) Network Measurement Results

Byte(s)	Beschreibung	Länge
1	Network Measurement Results tag	1
2	Länge = 16	1
3-18	Network Measurement Results	16

Tabelle 13: Network Measurement Results

Das Feld *Network Measurement Results* ist wie folgt codiert:

8	7	6	5	4	3	2	1	
BA-USED	DTX USED	RXLEV-FULL-SERVING-CELL						octet 1
	MEAS-VALID	RXLEV-SUB-SERVING-CELL						octet 2
	RXQUAL-FULL SERVING-CELL			RXQUAL-SUB SERVING-CELL			NO-NCELL-M	octet 3
NO-NCELL-M	RXLEV-NCELL 1							octet 4
BCCH-FREQ-NCELL 1						BSIC-NCELL 1		octet 5
BSIC-NCELL 1	RXLEV-NCELL 2							octet 6
RXLEV-NCELL 2	BCCH-FREQ-NCELL 2					BSIC-NCELL 2		octet 7
BSIC-NCELL 2				RXLEV-NCELL 3				octet 8
RXLEV-NCELL 3	BCCH-FREQ-NCELL 3						BSIC-NCELL 3	octet 9
BSIC-NCELL 3					RXLEV-NCELL 4			octet 10
RXLEV-NCELL 4	BCCH-FREQ-NCELL 4							octet 11

BSIC-NCELL 4		RXLEV-NCELL 5	octet 12
RXLEV-NCELL 5	BCCH-FREQ-NCELL 5		octet 13
BCCH-FREQ-NCELL 5	BSIC-NCELL 5	RXLEV-NCELL 6	octet 14
RXLEV-NCELL 6	BCCH-FREQ-NCELL 6		octet 15
BCCH-FREQ-NCELL 6	BSIC-NCELL 6		octet 16

Tabelle 14: Codierung Network Measurement Results

Alle Werte sind *most significant bit first* codiert. Bei den Werten, die über die Byte-Grenze hinweg gehen, steht der *high part* im ersten Byte, der *low part* im zweiten.

Für diese Arbeit sind folgende Werte relevant:

- **RXLEV-FULL-SERVING CELL** gibt die Signalstärke der BTS der *serving cell* an.
- **MEAS-VALID** gibt an, ob die *Network Measurement Results* gültig sind.
- **NO-NCELL-M** gibt an, von wie vielen Nachbarzellen Messdaten verfügbar sind.
- **RXLEV-NCELL n** gibt die Signalstärke des von der n-ten Nachbarzelle empfangenen Signals an.
- **BCCH-FREQ-NCELL n** verweist auf die Stelle der Kanalnummer der n-ten Nachbarzelle in der *BCCH channel list*. Die Kanalnummer ist also nicht unmittelbar gegeben, sondern muss in einem weiteren Datenobjekt (siehe folgender Abschnitt) nachgeschlagen werden.

(c) BCCH channel list

Die *BCCH channel list* ist eine Liste von Kanalnummern, in denen (mindestens) die Kanäle der empfangbaren Nachbarzellen vorkommen.

Byte(s)	**Beschreibung**	**Länge**
1	BCCH channel list tag	1
2	Länge = X	1
3-X+2	BCCH channel list	X

Tabelle 15: BCCH channel list

Für die *BCCH channel list* gibt es mehrere mögliche Codierungen: Zum einen die, die das ME auch intern benutzt (vgl. [gsm04.08]) und die sich in sechs weitere unterteilt, und zum anderen eine vereinfachte, die nur von SAT verwendet wird. Ob das ME die vereinfachte Codierung unterstützt, gibt ein Bit im TERMINAL PROFILE an (vgl. Abschnitt 1.7.2).

Jede Kanalnummer ist 10 Bit lang.

Bei der vereinfachten Codierung sind die Kanalnummern fortlaufend und über Byte-Grenzen hinweg *most significant bit first* codiert abgelegt. Ggf. unbenutzte Bits im letzten Byte sind mit Nullen aufgefüllt.

Für die ME-interne Codierung gibt es sechs verschiedene Formate. Welches Format benutzt wird, ist in den Bits 8, 7, 4, 3 und 2 des ersten Byte angegeben.

Bit 8	Bit 7	Bit 4	Bit 3	Bit 2	Format
0	0	(Daten)			bit map 0
1	0	0	(Daten)		1024 range
1	0	1	0	0	512 range
1	0	1	0	1	256 range
1	0	1	1	0	128 range
1	0	1	1	1	variable bitmap

Tabelle 16: Codierung des *BCCH channel list* Formats

Beim *bit map 0* Format entspricht jedes Bit genau einer Kanalnummer und gibt an, ob der Kanal in der Liste vorkommt oder nicht. Bit 4 des ersten Byte entspricht Kanal 124; Bit 1 des 16. Byte entspricht Kanal 1. Da mit diesem Format nur die ersten 124 GSM-Kanäle referenziert werden können, ist es nur für GSM900 geeignet.

Bei den Formaten *1024 range*, *512 range*, *256 range* und *128 range* werden die Kanäle mit einem komplizierten Rechenverfahren extrahiert, das in dieser Ausarbeitung nicht berücksichtigt wird, da keines der getesteten Mobiltelefone diese Formate verwendet hat.

Das *variable bitmap* Format ist die logische Erweiterung des *bit map 0* Formats für GSM1800 und GSM1900. Hier ist die niedrigste vorkommende Kanalnummer von Bit 1 des ersten Byte bis Bit 8 des dritten Byte abgelegt. Die darauffolgenden Bits geben das Vorhandensein (‘1’) bzw. Nichtvorhandensein (‘0’) der Kanäle in aufsteigender Reihenfolge relativ zum niedrigsten Kanal an.

(d) Timing Advance

Byte	Beschreibung	Länge
1	Timing Advance tag	1
2	Länge = 2	1
3	ME Status	1
4	Timing Advance	1

Tabelle 17: Timing Advance

Das Byte für *ME Status* ist 0, falls das Endgerät im *idle*-Zustand ist, und 1, falls es nicht im *idle*-Zustand ist. Die Werte 2-255 sind reserviert.

Der *Timing Advance* kann Werte von 0 bis 63 annehmen (vgl. Abschnitt 1.4). Die zwei *most significant* Bits sind reserviert.

1.7.4.3 SEND SHORT MESSAGE

Beim SAT-Befehl SEND SHORT MESSAGE müssen dem ME die für alle Befehle notwendigen Datenobjekte *Command details* und *Device identity* übergeben werden. Es kann ein *Address* Datenobjekt folgen, was die Adresse des SMSC angibt. (Falls es fehlt, wird die im ME gespeicherte Adresse verwendet.) Darauf muss ein *SMS TPDU* Datenobjekt folgen. Der Aufbau einer *SMS transport protocol data unit* ist in Abschnitt 1.5.1 angegeben.

1.7.4.4 SET UP EVENT LIST

SIM Application Toolkit bietet der SIM-Anwendung die Möglichkeit, sich über bestimmte Ereignisse informieren zu lassen. Zu diesen Ereignissen gehört beispielsweise ein eingehender Anruf oder eine Aktualisierung der Ortsinformation (*location update*; *Handover* zu einer anderen Zelle, nicht (!) neue *Measurement Results*).

Mit dem SAT-Befehl SET UP EVENT LIST aktiviert die SIM-Anwendung den Empfang von bestimmten Events; die Benachrichtigung selbst erfolgt dann mittels des SIM-Befehls ENVELOPE (vgl. Abschnitt 1.6) durch das ME.

Der SAT-Befehl muss neben den für alle Befehle notwendigen Datenobjekten *Command details* und *Device identity* das Datenobjekt *Event list* enthalten. Das ist eine Liste von Bytes, wobei jedes Byte genau einem abonnierten Event entspricht. ‘00’ steht dabei beispielsweise für *mobile terminated call*, also einen eingehenden Anruf; ‘03’ für *Location update.*

Die Datenobjekte des ENVELOPE sind in Abschnitt 1.7.9 beschrieben.

1.7.5 Empfang von Kurzmitteilungen

Wenn das ME eine Kurzmitteilung empfängt, überprüft es zunächst das Feld *TP-Protocol-Identifier* (vgl. Abschnitt 1.5.1). Findet es dort den Wert für *SIM data download* vor, so informiert es den Benutzer nicht über den Empfang und zeigt die Nachricht auch nicht an, sondern leitet die SMS-DELIVER-TPDU in einem ENVELOPE-Kommando an das SIM weiter.

Allein der Absender der Kurzmitteilung kann also bestimmen, ob seine Nachricht an die SIM-Karte zur Verarbeitung weitergeleitet wird. Die SIM-Anwendung hat keine Möglichkeit, den grundsätzlichen Empfang aller Kurzmitteilungen zu aktivieren.

Abhängig von der Antwort der SIM-Anwendung bestätigt es dem Netz in einer SMS-DELIVER-REPORT-TPDU den Empfang bzw. meldet einen Fehler.

Der ENVELOPE enthält folgende Datenobjekte:

- *Device identities* (*Source*: *Network*; *Destination*: SIM)
- *Address*: die Adresse des SMSC, das sie Nachricht zugestellt hat (Codierung vgl. [gsm04.08])
- SMS TPDU: die SMS-DELIVER-TPDU (vgl. [gsm03.40] und Abschnitt 1.5.1)

1.7.6 Empfang von Cell Broadcast Messages

Cell Broadcast ist ein unbestätigter Broadcast-Dienst für kurze Nachrichten (82 Oktette oder 93 Textzeichen). *Cell Broadcast Messages* werden von sogenannten *Cell Broadcast Centres* (CBC) in einem festgelegten Gebiet ausgesendet. Dieses Gebiet kann sich von einer einzelnen Zelle bis hin zum gesamten Netz erstrecken.

Um mehrere verschiedene Informations-Dienste mittels *Cell Broadcast* nebeneinander zu ermöglichen, enthält jede Nachricht eine Kanalnummer. Der Kanal, auf dem O_2 die Koordinaten der Zelle überträgt, hat zum Beispiel die Nummer 221. Andere Mobilfunkbetreiber nutzen *Cell Broadcast*, um die aktuelle Ortsvorwahl oder Nachrichten und Verkehrsfunk zu übertragen.

Ob eine *Cell Broadcast Message* an die SIM-Karte weitergeleitet wird, bestimmt die SIM-Anwendung selbst, indem sie die entsprechenden Kanäle in das *Elementary File* EF_{CBMID} einträgt (vgl. Abschnitt 1.6).

Beim Empfang einer *Cell Broadcast Message* überprüft das ME zuerst, ob die Kanalnummer in EF_{CBMID} steht und leitet die Nachricht ggf. in einem ENVELOPE-Kommando an das SIM weiter. Falls der Kanal nicht von der Anwendung abonniert ist, stellt es fest, ob der Benutzer den Empfang abonniert hat und zeigt die Nachricht ggf. an. Das bedeutet, dass Kanäle, die die SIM-Anwendung angefordert hat, für den Benutzer nicht mehr sichtbar sind.

Der ENVELOPE enthält folgende Datenobjekte:

- *Device identities* (*Source*: *Network*; *Destination*: SIM)
- *Cell Broadcast page*: die CB-Nachricht (vgl. [gsm03.41])

1.7.7 Menü-Auswahl

Mit dem SAT-Befehl SET UP MENU kann eine SIM-Anwendung ein zusätzliches Menü im Mobiltelefon einrichten (vgl. Abschnitt 1.7.4).

Wenn der Benutzer einen Punkt aus diesem Menü auswählt, informiert das ME die SIM-Anwendung in einem ENVELOPE-Kommando über die Auswahl.

Der ENVELOPE enthält folgende Datenobjekte:

- *Device identities* (*Source*: *Keypad*; *Destination*: SIM)
- *Item identifier*: eine Identifikationsnummer, die der Entwickler beim Einrichten des Menüs jedem einzelnen Menüpunkt zugeordnet hat.

Es ist nicht möglich, mit einem einzelnen SAT-Befehl eine ganze Menüstruktur aufzubauen. Die SIM-Anwendung muss vielmehr auf ein MENU SELECTION Ereignis einen SELECT ITEM Befehl senden, um eine Auswahlliste anzuzeigen, wenn sie ein verschachteltes Menü erzeugen soll.

1.7.8 Ablaufen eines Timers

Mit dem SAT-Befehl TIMER MANAGEMENT kann eine SIM-Anwendung Timer starten, zurücksetzen und deren Werte abfragen (vgl. Abschnitt 1.7.4). Es stehen acht Timer zur Verfügung.

Wenn ein Timer abläuft, informiert das ME die SIM-Anwendung in einem ENVELOPE-Kommando darüber.

Der ENVELOPE enthält folgende Datenobjekte:

- *Device identities* (*Source*: ME; *Destination*: SIM)
- *Timer identifier*: die Nummer des Timers (1..8)
- *Timer value*: die Zeit seit Starten des Timers

1.7.9 Event download

Mit dem SAT-Befehl SET UP EVENT LIST kann eine SIM-Anwendung dem ME mitteilen, über welche Ereignisse es informiert werden möchte (vgl. Abschnitt 1.7.4).

Ein mögliches Ereignis ist beispielsweise eine Aktualisierung der Ortsinformation (*Location update*). Damit ist das Umbuchen in eine neue Zelle gemeint, nicht aber eine Änderung der *Measurement Results* oder des *Timing Advance*.

Tritt ein abonniertes Ereignis ein, informiert das ME die SIM-Anwendung in einem ENVELOPE-Kommando darüber.

Der ENVELOPE enthält im Fall eines *Location Update* folgende Datenobjekte:

- *Event list*: nur ein Event: *Location Status*
- *Device identities* (*Source*: ME; *Destination*: SIM)
- *Location status*: ‘00’ = kein Netz, ‘01’ = *limited service*, also nur Notrufe möglich, ‘02’ = normaler Betrieb
- *Location information*: (vgl. Abschnitt 1.7.4) (entfällt, falls nicht normaler Betrieb)

Kapitel 2

Anforderungen

Es ist ein System zur terminal-basierten Positionsbestimmung von GSM-Mobiltelefonen zu entwerfen und zu realisieren. Die vom Mobiltelefon (unter anderem zum Zweck des *Handovers*) ständig gemessenen und zur Positionsbestimmung nützlichen Daten sollen an ein Hintergrundsystem, den *Location Server*, übertragen werden. Die Übertragung ist durch eine Anwendung auf der SIM-Karte des Mobiltelefons mittels *SIM Application Toolkit* Funktionen zu realisieren. Der Location Server nimmt die Daten des Mobiltelefons entgegen, berechnet daraus dessen Position und verwaltet die Ortsdaten in einer Datenbank.

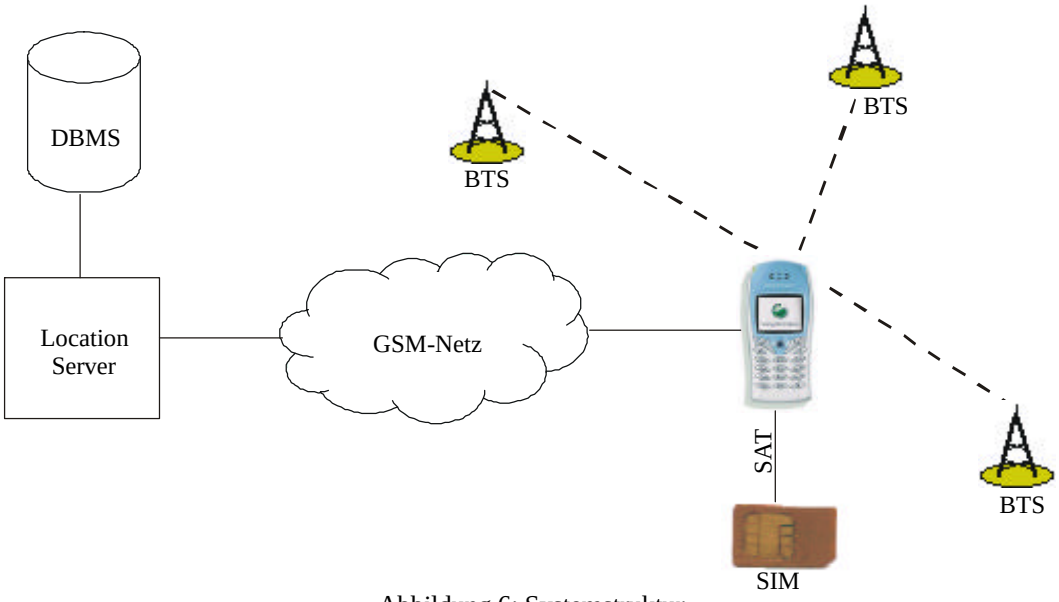

Abbildung 6: Systemstruktur

2.1 SIM-Anwendung

Die SIM Anwendung soll die zur Lokalisierung notwendigen Daten vom Mobiltelefon auslesen und an das Hintergrundsystem senden können.

Das soll entweder
– vom Hintergrundsystem angestoßen werden,
– vom Benutzer angestoßen werden oder
– automatisch in vom Benutzer über einen Menüeintrag festlegbaren Intervallen geschehen.

Das Hintergrundsystem soll die Daten anfordern können entweder
– per SMS,
– per USSD oder

– durch einen Anrufaufbau, bei dem die SAT-Anwendung das Hintergrundsystem an der übermittelten Rufnummer erkennt.

Die Messdaten sollen an das Hintergrundsystem versandt werden entweder
– per SMS oder
– per USSD.

In einem weiteren Menüeintrag soll der Benutzer bestimmen können,
– ob er das Auslesen der Messdaten generell verbietet, oder
– ob er nur das Auslesen der aktuellen Zelle zulässt, oder
– ob er das Auslesen aller Messdaten zulässt.

2.2 Location Server

Der Location Server fragt über die serielle Schnittstelle entweder ein per Datenkabel angeschlossenes Mobiltelefon oder ein Funkmodem ständig nach neuen SMS ab, liest ggf. neue SMS aus und legt die Daten strukturiert in Objekten ab. (Im Fall der Übertragung per USSD baut der Location Server eine Socket-Verbindung zum USSD-Gateway des Mobilfunkproviders auf und nimmt die USSD-Strings von dort entgegen.)

Die Messdaten übergibt der Location Server dem in [sep] entwickelten Algorithmus, der daraus die geographische Position des Benutzers errechnet.

Der Algorithmus greift zur Berechnung auf eine Datenbank zu, die die Daten über alle BTS des Mobilfunknetzes und deren geographische Position bereitstellt.

Das Ergebnis des Algorithmus legt der Location Server zusammen mit der MSISDN des Benutzers in einer Datenbank ab.

Einstellungen (zum Beispiel Dateinamen) liest der Location Server aus einer Konfigurations-Datei.

Eine andere Anwendung kann den Location Server auffordern, Messdaten von einem GSM-Endgerät anzufordern, indem sie eine Datei in einem Spool-Verzeichnis anlegt, das der Location Server ständig ausliest.

Kapitel 3

Design und Realisierung

3.1 Systemweite Design-Entscheidungen

3.1.1 Datenübertragung und -anforderung

Für die Übertragung der Messdaten von der MS zum Hintergrundsystem kommen grundsätzlich zwei Arten in Frage: SMS und USSD (vgl. Abschnitt 1.5).

Während SMS schon lange im Einsatz ist und heute von praktisch jedem Mobiltelefon unterstützt wird, handelt es sich bei *Unstructured Supplementary Service Data* (USSD) um einen Dienst, der im Moment noch nicht für die Datenübertragung zu netzexternen Systemen genutzt wird.

Ein klarer Vorteil von SMS ist, dass dieser Dienst eine Kommunikation von Endgerät zu Endgerät ermöglicht und die Daten vom Hintergrundsystem somit einfach mit Hilfe eines Mobiltelefons oder Funkmodems ohne Mitwirken des Mobilfunkproviders empfangen werden können. USSD ist dagegen für die Kommunikation zwischen Mobiltelefon und Mobilfunknetz gedacht. Damit ein netzexternes System (beispielsweise ein Internet-Host) über USSD mit einer MS kommunizieren kann, muss es ein USSD-Gateway nutzen, das der Provider zur Verfügung stellen muss.

Ein Vorteil von USSD gegenüber SMS ist die Geschwindigkeit. SMS überträgt Daten über die *Service Centres* nach dem *Store-and-Forward*-Prinzip. Damit ist weder eine maximale Übertragungsdauer noch die Beibehaltung der Reihenfolge garantiert. Bei USSD wird dagegen ein Signalisierungskanal zwischen MS und Netzanwendung (hier: USSD-Gateway) aufgebaut.

Die Entscheidung für einen Dienst muss auch davon abhängen, ob Kunden unterschiedlicher Mobilfunkanbieter das System benutzen sollen. Während SMS zwischen verschiedenen Mobilfunk-Netzen – sogar über Ländergrenzen hinweg – übertragen wird, muss der Location Server im Fall von USSD Verbindungen zu den USSD-Gateways aller in Frage kommender Anbieter aufnehmen und, weil in Moment kein Protokoll für die Kommunikation zwischen USSD-Gateway und Internet-Host festgelegt ist, unter Umständen viele verschiedene Protokolle implementieren.

Da über die Tariffierung von USSD im Moment noch nichts bekannt ist, lässt sich nicht sagen, welche Übertragung weniger Kosten verursachen würde. Während aber Kurznachrichten grundsätzlich demjenigen in Rechnung gestellt werden, der sie versendet, ist zu erwarten, dass die Abrechnung von USSD flexibler sein wird, so dass der Anbieter des Systems alle Kosten tragen kann und dem Mobilfunkkunden keine Kosten pro Lokalisierung entstehen.

O_2 ist der erste Anbieter in Deutschland, der USSD über ein *Gateway* externen Dienst-Anbietern zur Verfügung stellen wird. Leider ist dieses *Gateway* während der Bearbeitung dieser Bachelor

Thesis erst in Planung und wird frühestens im August 2002 für erste Versuche zur Verfügung stehen.

Die Design-Entscheidung fällt deshalb vorerst zu Gunsten von SMS zur Datenübertragung aus. Aufgrund der genannten Vorteile von USSD sollte der Ansatz im weiteren Verlauf des COSMOS-Projekts aber auf jeden Fall weiter verfolgt werden.

Eine weiterer Ansatz, der zumindest für die Datenanforderung in Frage kommen würde, wäre ein Anrufaufbau mit Übertragung der Rufnummer. Er verursacht überhaupt keine Kosten, solange das Mobiltelefon ihn nicht annimmt, und reicht für die Datenanforderung völlig aus, da hier keine weiteren Daten übertragen werden müssen.

SIM Application Toolkit bietet die Möglichkeit, das SIM bei eingehenden Anrufen zu informieren und übergibt dabei auch die MSISDN des Anrufers. Da die SIM-Anwendung allerdings keine Möglichkeit hat, den Anruf abzulehnen oder ihn vor dem Benutzer zu „verstecken", müsste der Location Server den Rufaufbau im selbem Moment unterbrechen, in dem die Signalisierung beim Mobiltelefon ankommt.

Dieser Ansatz wurde während der Entwicklung nicht weiter verfolgt und soll als Anregung für den weiteren Projekt-Verlauf verstanden werden.

3.1.2 Daten über Sendestationen

Elementare Grundlage für die Positionsbestimmung durch Verarbeitung der Zellinformation und Signalstärken ist eine Datenbank, die alle Daten über die *Base Transceiver Stations* enthält, d.h. *Location Area Codes* (LAC), *Cell Identificators* (CID), Kanalnummern und insbesondere deren Standorte.

Eine solche vollständige Datenbank hat einzig und allein der Mobilfunkbetreiber. Es gibt zwar einige Projekte, bei denen Interessierte per Internet solche Daten zusammentragen, aber so eine „private Sammlung" kann niemals vollständig und aktuell sein. Obwohl Mobilfunkbetreiber natürlich nicht sehr häufig neue BTS aufstellen oder alte deaktivieren, so passen sie aber sehr regelmäßig (und zwar im schlimmsten Fall täglich) die Zuordnung von Kanalnummern zu BTS und von BTS zu *Location Areas* an, um auf das sich schnell ändernde Kundenverhalten zu reagieren und die knappen Frequenzbänder immer optimal zu nutzen.

Ein Dienst zur Positionsbestimmung nach dem in dieser Arbeit verwendeten Verfahren ist also in jedem Fall auf die Kooperation des Mobilfunkproviders oder – wenn er seinen Benutzerkreis nicht einschränken möchte – aller Betreiber angewiesen.

Die hier testweise verwendeten Daten stammen aus dem Systementwicklungsprojekt vom Februar 2002 (vgl. [sep]), bei dem die Bearbeiter mit ihrer Windows-CE-Software in München in der näheren Umgebung des Stammgeländes der Technischen Universität selbst Daten gesammelt und automatisiert gespeichert haben. Die geographische Information waren dabei die Koordinaten im Gauß-Krüger-Format, die O_2 über *Cell Broadcast* aussendet.

Die Funktionalität, dass die Zell-Datenbasis mit jeder Anfrage, die eine bis dahin unbekannte *serving cell* enthält, automatisch erweitert wird, ist nach dem in dieser Arbeit verwendeten Verfahren leider ausgeschlossen, da *SIM Application Toolkit* nur die Kanalnummer der Nachbarzellen, nicht aber der *serving cell* liefert. In [sep] standen LAC, CID, Kanalnummer und geographische Position der *serving cell* gleichzeitig zur Verfügung und konnten als vollständiger Datensatz ggf. in die Datenbasis übernommen werden.

3.1.3 Datenformat zur Übertragung

Da die SIM-Karte über wesentlich weniger Rechenleistung verfügt als das Hintergrundsystem, soll die SIM-Anwendung die ortungsrelevanten Daten überhaupt nicht interpretieren oder verändern.

Die TERMINAL RESPONSEs, in denen das Mobiltelefon die ortungsrelevanten Daten verpackt, soll sie unverändert, d.h. im TLV-Format (vgl. Abschnitt 1.7.3), aneinander hängen.

Eine *broadcast message* ist normalerweise 7-Bit-codiert. Die Gauß-Krüger-Koordinaten, die über den Kanal 221 empfangen werden, sind zwölf Zeichen lang und belegen somit in 7-Bit-Codierung die ersten elf Oktette des *Content of Message.* Nur diese elf relevanten Oktette sollen übertragen werden.

Damit das Format zu den anderen Daten passt und der Location Server die Kurzmitteilung auch interpretieren kann, wenn TERMINAL RESPONSES und Koordinaten anders angeordnet sind, wird ihnen das Tag ‘BB’ und die Länge ‘0B’ vorangestellt. Das Tag ‘BB’ ist in [gsm11.14, 13.3] als *not used* verzeichnet, wird also nicht verwendet und ist auch nicht reserviert.

3.2 SIM-Anwendung

3.2.1 Design

3.2.1.1 Zugriff durch den Benutzer

Die Interaktion zwischen der SIM-Anwendung und dem Benutzer geschieht über ein Menü, das in die vorhandene Menüstruktur des Mobiltelefons eingebunden wird. Das Menü samt Untermenüs ist in der folgenden Grafik dargestellt.

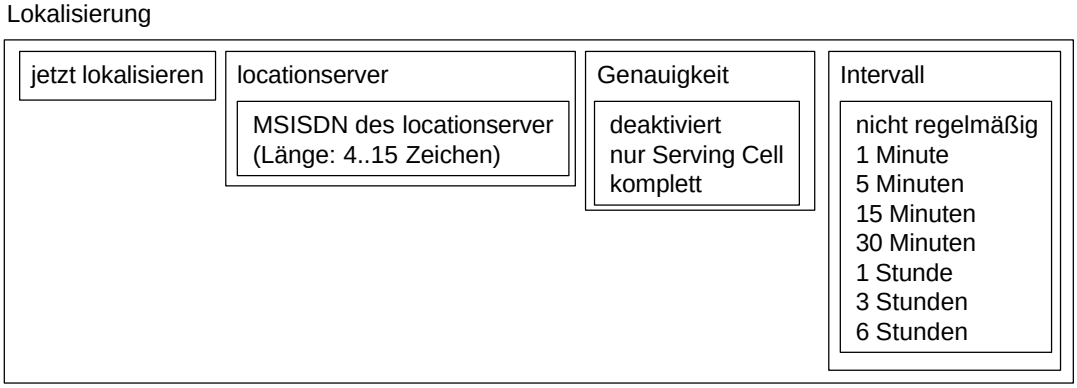

Abbildung 7: Menüführung der SIM-Anwendung

3.2.1.2 Datenhaltung

Die SIM-Anwendung muss einige Daten permanent speichern können, so dass sie auch nach dem Ausschalten des Mobiltelefons erhalten bleiben.

Das sind:

- MSISDN des Location Server
- Genauigkeit
- Intervall

Sie muss außerdem die zuletzt empfangenen Koordinaten der *serving cell* temporär speichern können.

3.2.1.3 Ereignisse

Auf das Menü-Ereignis „MSISDN des Location Server“ muss die Anwendung mit einer Eingabe-Aufforderung reagieren und den eingegeben Nummern-String permanent speichern. Auf das Menü-Ereignis „Genauigkeit“ und „Intervall“ muss sie mit einer Auswahl-Liste reagieren und abhängig von der getroffenen Auswahl die permanente Variable „Genauigkeit“ bzw. „Intervall“ verändern.

Wie die SIM-Anwendung auf den Empfang einer *cell broadcast message* oder einer Kurznachricht reagieren muss, ist in den folgenden zwei Flussdiagrammen dargestellt.

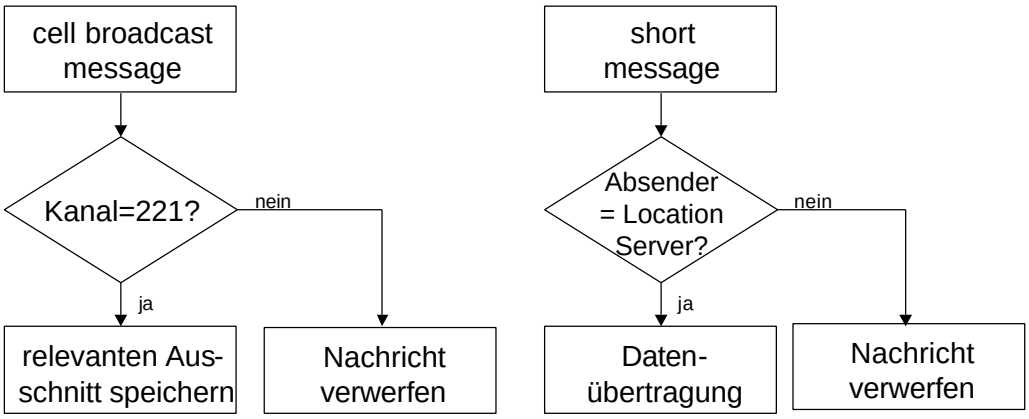

Abbildung 8: Empfang von cell broadcast message und Kurznachricht

Die ortungsrelevanten Daten soll die Anwendung auslesen und an den Location Server übertragen, wenn

- der Benutzer den Menüpunkt „jetzt lokalisieren“ auswählt,
- ein Timer-Ereignis eintritt, oder
- ein Kurznachricht vom Location Server eintrifft (siehe Abbildung 8).

Das soll wie folgt ablaufen: Falls der Benutzer die Übertragung aller ortungsrelevanten Daten wünscht (Genauigkeit „komplett“), ruft die Anwendung den SAT-Befehl PROVIDE_LOCAL_INFORMATION dreimal auf – jeweils mit den *Command details* für die Abfrage von *Local Information*, *Network Measurement Results* und *Timing Advance* – und hängt die

zurückgelieferten TERMINAL RESPONSEs zusammen mit den Gauß-Krüger-Koordinaten – falls vorhanden – wie in Abschnitt 3.1.3 festgelegt aneinander.

Falls der Benutzer die Genauigkeit auf die *serving cell* beschränkt hat, sollen nur *Local Information* und die Gauß-Krüger-Koordinaten übertragen werden.

Hat der Benutzer die Lokalisierung ganz untersagt (Genauigkeit „deaktiviert“), so werden überhaupt keine Daten übertragen.

3.2.2 Probleme bei der Realisierung und Kompromiss

Im Lauf des Projekts ist der Partner O_2 von seiner Kooperation zurückgetreten. Auch die Firma Giesecke & Devrient, die die SIM-Karten mit der Anwendung herstellen sollte, konnte nicht rechtzeitig eine konkrete Zusage machen, so dass das Teilprojekt, das in dieser Thesis bearbeitet wurde, leider nicht real umgesetzt werden konnte.

Die Anforderungen für den Teil des Systems, der auf der SIM-Karte läuft, wurden deshalb abgeändert: Die Funktionalität sollte in einer Software für das Betriebssystem PalmOS entwickelt werden, die über eine serielle Verbindung mit dem Mobiltelefon kommuniziert.

Ziel war dabei zum einen, *SIM Application Toolkit* Funktionen für so viele Funktionen wie möglich zu verwenden, d.h. neben der Abfrage der zur Lokalisierung notwendigen Daten sollte auch die Menüführung, der Versand der Kurznachrichten und die Steuerung der Timer über SAT-Funktionen realisiert werden.

Außerdem sollte die Software nicht objekt-orientiert sein und nur elementare Datentypen verwenden, damit sie zu einem späteren Zeitpunkt einfach auf ein SIM-Karten-Betriebssystem portiert werden kann.

Es gibt sehr wenige Mobiltelefone, die über die serielle Schnittstelle Zugriff auf die SAT-Funktionen bieten. Zwei davon sind die Siemens-Modelle C35 und S35, mit denen das vorliegende System entwickelt wurde.

3.2.3 Modularität

Die vorliegende Palm-OS-Software wurde streng modular entwickelt. Die Beziehung der Module zueinander zeigt folgende Grafik.

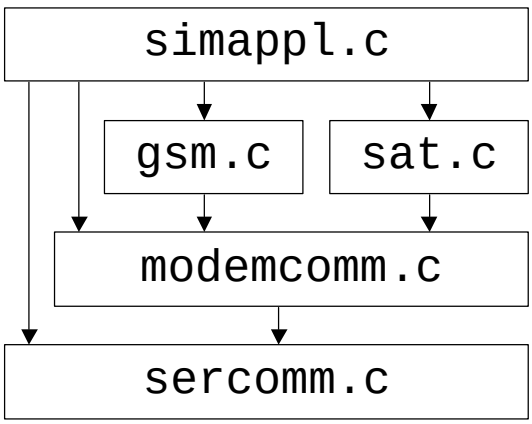

Abbildung 9: Beziehung der Module der SAT-Anwendung

Alle Funktionen, die die serielle Kommunikation mit dem Mobiltelefon zur Verfügung stellen, befinden sich in der Quellcode-Datei `sercomm.c`. Das umfasst im wesentlichen das Öffnen und Schließen der Schnittstelle und der Empfang und das Senden von (beliebigen) Daten.

Die Funktionen, die die modemspezifische Kommunikation zur Verfügung stellen, bauen auf die oben genannten Funktionen auf und befinden sich in der Quellcode-Datei `modemcomm.c`. Das umfasst die Initialisierung der AT-Kommunikation, das Senden von AT-Kommandos mit Auswertung der Antwort und das Senden einer PDU an das Mobiltelefon.

Die Quellcode-Datei `sat.c` enthält die Aufrufe der von der Anwendung genutzten *SIM Application Toolkit* Funktionen, also Benutzerführung, Timer Management, Abfrage der Lokalisierungs-Informationen, Übersetzung der Fehlercodes und einige Hilfsfunktionen, und stützt sich auf die Prozeduren aus `modemcomm.c`.

In `gsm.c` sind die Funktionen enthalten, die spezifisch für GSM sind und ebenfalls auf `modemcomm.c` aufbauen. Dazu gehört eine Prozedur zum Versenden von SMS, das Aktivieren und Deaktivieren der Benachrichtigung der Software beim Eingang neuer Kurznachrichten oder Broadcast-Nachrichten und das Abonnieren des Broadcast-Kanals 221, auf dem O_2 die Position der *serving cell* sendet.

`simappl.c` ist schließlich das Herzstück der Anwendung. Sie enthält die Initialisierung, die Handler-Prozedur, die eingehende Daten des Mobiltelefons verarbeitet, und eine Prozedur, die die Lokalisierungsdaten abfragt und per SMS versendet.

Alle Funktionen, die spezifisch für PalmOS sind, also Darstellung von Informationen auf dem Bildschirm, das Behandeln von Events des Betriebssystems, Menüführung, Formulare, usw. finden sich in den restlichen Dateien und sind nicht Gegenstand dieser Ausarbeitung.

3.2.4 Allgemeines

Um die in Abschnitt 3.2.1.2 aufgezählten Daten permanent zu speichern, greift die Palm-Anwendung auf die *Application Preferences Database* von PalmOS zurück. Das ist eine Datenbank, die PalmOS verwaltet und in der jede Anwendung eine kleine Menge Daten ablegen kann, die zwischen Schließen und erneutem Öffnen erhalten bleiben sollen.

Die *preferences* müssen in keinem bestimmten Format vorliegen, sondern jede Anwendung kann – in C typischerweise mit einer `typedef struct` Anweisung – das Format selbst festlegen und übergibt diese Struktur zusammen mit deren Größe einer Systemfunktion `PrefSetApp-Preferences` bzw. bekommt beides von der Systemfunktion `PrefGetAppPreferences` übergeben.

Die Palm-Software wird über drei Menü-Punkte gesteuert. Über *Options* > *Settings* kann man Einstellungen für die serielle Schnittstelle vornehmen (physisch oder über Infrarot, Geschwindigkeit, Bytelänge, Parität und Anzahl Stop-Bits). Auch diese Einstellungen werden in den *application preferences* permanent gespeichert. Über *Connection* > *Connect* wird die Verbindung zum dem Mobiltelefon aufgebaut und die Funktionalität der SIM-Anwendung simuliert. Über *Connection* > *Disconnect* wird die Verbindung unterbrochen und die Simulation beendet.

Während der Simulation geben Meldungen in einem scrollenden Textfeld im Hauptfenster der Palm-Anwendung Auskunft über eingegangene Ereignisse und den momentanen Vorgang.

3.2.5 Serielle Kommunikation

Die serielle Kommunikation stützt sich auf die Funktionen des neuen *Serial Manager*, der seit der Version 3.3 von PalmOS angeboten wird. Eine Überprüfung der PalmOS-Version wird beim Start des Programms durchgeführt. Die zur Verfügung gestellten Funktionen sind unkomplizierter als bei DOS/Windows, bieten dafür aber auch weniger Funktionalität.

So fehlt zum Beispiel die Möglichkeit, beim Betriebssystem eine Handler-Prozedur zu registrieren, die aufgerufen wird, sobald eingehende Daten an der Schnittstelle bereitstehen. Um eingehende Daten vom Mobiltelefon zu verarbeiten, muss deshalb in der *Event Loop*, die ohnehin jedes Palm-Programm hat, regelmäßig überprüft werden, ob Daten bereitstehen, und darauf entsprechend reagiert werden.

`Err OpenSerial()`

Die Prozedur `OpenSerial` liest aus den Benutzereinstellungen (*preferences*) die Einstellungen für die serielle Verbindung aus (also Port, Geschwindigkeit, Parität, Stoppbits und Anzahl der Datenbits) und öffnet die serielle Schnittstelle durch einen Systemaufruf. Der Rückgabewert wird ausgewertet, und im Fehlerfall wird eine Fehlermeldung angezeigt und der Port geschlossen.

`void CloseSerial()`

Die Prozedur `CloseSerial` wartet, bis der Ausgabe-Puffer leer ist, d.h. bis alle anstehenden Daten gesendet wurden, und schließt dann die serielle Schnittstelle.

`UInt8 readByte()`

Die Prozedur `readByte` liest ein Byte von der seriellen Schnittstelle. Dabei wird nicht überprüft, ob ein Byte im Eingangs-Puffer bereitsteht, sondern ggf. eine Timeout-Fehlermeldung ausgegeben und ein Null-Byte zurückgegeben. Auch alle anderen Fehlersituationen werden erkannt und dem Benutzer in einem Dialog mitgeteilt.

`Boolean inputAvailable()`

Die Prozedur `inputAvailable` überprüft, ob Daten im Eingangs-Puffer der seriellen Schnittstelle bereitstehen und gibt dementsprechend `true` oder `false` zurück.

`void getLine(char *buffer)`

Die Prozedur `getLine` stützt sich auf `readByte` und liest solange Bytes in den Rückgabe-Puffer, bis sie einen *linefeed* erkennt. *Linefeeds* und *carriage returns* am Anfang werden nicht in den Puffer geschrieben.

Die Speicherverwaltung von PalmOS erlaubt es leider nicht, mit einer `new`-Anweisung Speicher auf dem Heap zu allokieren wie beim PC und einfach einen Pointer darauf zurückzugeben. Deshalb muss der Speicherbereich, in den `getLine` schreibt, von der aufrufenden Prozedur reserviert werden.

`void putLine(const char *text, Boolean carriageReturn)`

Die Prozedur `putLine` sendet die übergebene Folge von Zeichen an die serielle Schnittstelle. Falls der Parameter `carriageReturn true` ist, wird danach auch ein *carriage return* gesendet, der den Abschluss eines AT-Befehls anzeigt.

3.2.6 Modem-Kommunikation mit AT-Befehlen

Die Schnittstelle zwischen einem Terminal und einem Modem geht auf die ITU zurück, die in ihrem *ITU-T Draft new Recommendation V.25ter: „Serial asynchronous dialling and control“* die ausschließlich auf alpha-numerischen Zeichen basierende Syntax für Befehle (*commands*), Antworten (*responses*) und asynchrone Meldungen festlegt.

[gsm07.07] und [gsm07.05] erweitert diese Syntax um mächtigere Befehle, mit denen der Zugriff auf das Mobiltelefon und die SIM-Karte ermöglicht wird. Diese Syntax ist verbindlich für alle Hersteller.

Viele Hersteller von Mobiltelefon erweitern die Syntax noch weiter, um eigene spezifische Befehle zu ermöglichen. So führt Siemens beispielsweise bei seinen Mobiltelefonen C35, S35 und neuere einen Befehl ein, mit dem auf *SIM Application Toolkit* Funktionen zugegriffen werden kann (vgl.[siemens]).

Der Anfang einer Kommandozeile wird mit dem *command line prefix* (den zwei Zeichen „AT“) gekennzeichnet. Danach können beliebig viele *commands* folgen.

Es gibt

- *basic commands*, die aus ihrem Namen und ggf. einem Gleichzeichen („=“) und einem Parameter bestehen und die nicht abgeschlossen werden.
- *extended commands*, die aus einem Pluszeichen („+“), ihrem Namen und ggf. einem Gleichzeichen („=“) und beliebig vielen durch Kommas getrennten Parametern bestehen und mit einem Semikolon abgeschlossen werden.
- *read commands*, die sich von den *basic commands* und *extended commands* dadurch unterscheiden, dass sie statt des Gleichzeichens und der Parameter ein Fragezeichen haben. Sie weisen das Modem an, einen Wert auszulesen und zurückzuliefern.
- *test commands*, die sich von den *basic commands* und *extended commands* dadurch unterscheiden, dass sie statt der Parameter ein Fragezeichen haben. Sie fragen vom Modem Wertebereiche von Parametern ab.

Die Kommandozeile wird mit einem *carriage-return*-Zeichen (CR) abgeschlossen.

Anfang und Ende einer *information response* und eines *result code* sind mit einem *carriage-return*- (CR) und einem *line-feed*-Zeichen (LF) gekennzeichnet.

Eine *information response* auf ein *extended read command* ist aufgebaut aus einem Pluszeichen („+“), dem Namen, einem Doppelpunkt, einem Leerzeichen und den durch Kommas getrennten Rückgabewerten.

Ein *result code*, der auf jedes Kommando folgt, besteht nur aus den Strings „`OK`“ oder „`ERROR`“.

Falls ein *extended command* oder ein herstellerspezifisches Kommando eine PDU verlangt (beispielsweise eine SMS Transport PDU oder einen SAT-Befehl), so sendet das Mobiltelefon anstatt einer *information response* zuerst ein Größerzeichen („>“) und ein Leerzeichen. Es erwartet dann die PDU in hexadezimaler Darstellung, abgeschlossen durch einen 0x1A-Zeichen

(Strg+Z). Das Mobiltelefon sendet daraufhin die *information response* (bei einem SAT-Befehl beispielsweise die TERMINAL RESPONSE) und den *result code.*

Die Umsetzung der AT-Syntax auf die Signalisierung im Mobiltelefon übernimmt der *Terminal Adapter* (TA), der in der Regel im Mobiltelefon integriert ist. Das Gerät, das von außen auf das Mobiltelefon zugreift, wird als *Terminal Equipment* (TE) bezeichnet. Das Mobiltelefon selbst heißt hier *Mobile Equipment* (ME). Die folgende Grafik zeigt den Kommunikationsweg:

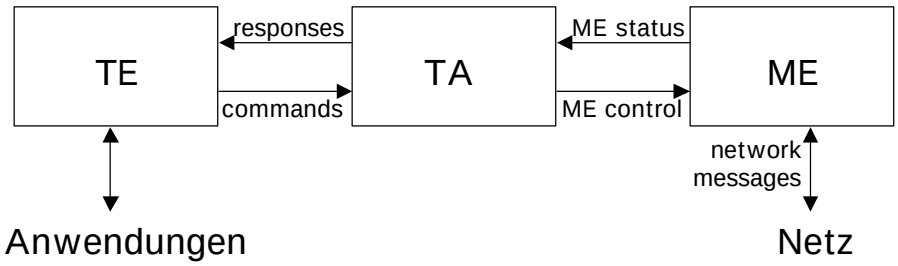

Abbildung 10: Anordnung von TE, TA und ME (aus [gsm07.07])

Die Verbindung zwischen TE und TA kann durch ein serielles Kabel erfolgen oder mit Infrarot-Übertragung realisiert werden.

`Boolean InitModem()`

Die Prozedur `InitModem` soll nach dem Öffnen der seriellen Schnittstelle die Kommunikation zwischen der Palm-Anwendung und dem Mobiltelefon initialisieren. Dazu sendet es ein Z-Kommando, das alle (AT-)Parameter auf ihre Standard-Werte zurücksetzt, erwartet den *result code* `OK` zurück, sendet dann ein E0-Kommando, das das Echo der gesendeten Befehle abschaltet und erwartet wiederum den *result code* `OK` zurück.

Im Fehlerfall gibt die Prozedur eine Fehlermeldung aus und liefert als Ergebnis `false` zurück, sonst `true`.

`void chat(const char *atCommand, char *result1, char *result2, Boolean acceptError)`

Die Prozedur `chat` sendet den *command line prefix* `AT`, gefolgt von dem als Parameter übergebenen AT-Befehl und einem carriage-return-Zeichen. Die empfangenen Zeilen (*information response*, PDUs und/oder *result code*) schreibt es an die Speicherstellen, die mit den Pointern `result1` und `result2` angegeben sind, außer `result1` und/oder `result2` sind gleich `NULL`. Falls als Parameter `acceptError false` übergeben wird, meldet die Prozedur dem Benutzer einen Fehler, wenn im *result code* der String `ERROR` enthalten ist.

`void sendPdu(const char *atCommand, const char *response, const char *pdu, char *result)`

Die Prozedur `sendPdu` ist zum Übergeben einer PDU an das Mobiltelefon vorgesehen. Sie sendet den *command line prefix* `AT`, gefolgt von dem als Parameter übergebenen AT-Befehl und einem *carriage-return*-Zeichen, erwartet ein Größerzeichen („>“) und ein Leerzeichen zurück und sendet dann die als Parameter erhaltene PDU, gefolgt von einem 0x1A-Zeichen (Strg+Z).

Anschließend vergleicht die Prozedur, ob der Anfang der vom Mobiltelefon empfangenen Zeile mit dem String `response` übereinstimmt, schreibt dann den Rest der Zeile an die Speicherstelle, die mit dem übergebenen Pointer `result` bezeichnet ist, und erwartet abschließend den Empfang des *result code* `OK`.

Im Fehlerfall wird eine Meldung ausgegeben und `result` vor dem Rücksprung nicht verändert.

3.2.7 SIM Application Toolkit Funktionen

Die acht Prozeduren, die die SIM Application Toolkit Funktionen aufrufen, laufen alle analog ab. Sie erwarten als einzigen Parameter einen Pointer, wo die TERMINAL RESPONSE des Mobiltelefons abgelegt werden kann und geben keinen Wert zurück.

Zuerst wird jeweils der SAT-Befehl zusammengesetzt. Alle Befehle müssen die Datenobjekte *Command Details* und *Device identities* enthalten. Daneben erwarten die meisten SAT-Befehle noch spezifische Datenobjekte. Dann wird die PDU, die den Befehl enthält, gesendet und die empfangene TERMINAL RESPONSE an den übergebenen Pointer `result` geschrieben.

- SET_UP_MENU veranlasst das Mobiltelefon, einen neuen Menüpunkt im Hauptmenü anzulegen, der wiederum Menüpunkte enthält. Die Namen von Menü und Menüpunkten sind in der Prozedur fest codiert, was bei einer SIM-Anwendung aufgrund des beschränkten Befehlsumfangs und Speicherplatzes und der geringen Rechenleistung Sinn macht.
- GET_INPUT veranlasst das Mobiltelefon, vom Benutzer die Eingabe der MSISDN des Location Server abzufragen. Text, Mindest- und Höchstlänge der Eingabe sind fest codiert. Als Standard-Wert wird die aktuelle MSISDN übergeben, die dem Benutzer angezeigt werden soll.
- SELECT_ITEM_Genauigkeit und SELECT_ITEM_Intervall veranlassen das Mobiltelefon, dem Benutzer eine Auswahl von fest vorgegebenen Optionen anzubieten. Im ersten Fall wählt der Benutzer eine von drei Genauigkeitsstufen der Ortung aus. Im zweiten Fall wählt er, wie oft eine regelmäßige Übertragung der Daten erfolgen soll. Die Strings sind wieder fest codiert.
- TIMER_MANAGEMENT liest aus der globalen Variable das Intervall aus und setzt einen Timer auf die entsprechende Zeit bzw. deaktiviert ihn.
- PROVIDE_LOCAL_INFORMATION_Local_Information liest *Mobile Country Code*, *Mobile Network Code*, *Location Area Code* und *Cell ID* aus.
- PROVIDE_LOCAL_INFORMATION_Network_Measurement_Results liest den Empfangspegel der *serving cell* sowie Kanalnummern und Empfangspegel von bis zu sechs Nachbarzellen aus.
- PROVIDE_LOCAL_INFORMATION_Timing_Advance liest den *Timing Advance* aus.

Beide Siemens-Telefone, die zum Test zur Verfügung standen, haben den SAT-Befehl SEND SHORT MESSAGE nicht unterstützt: Obwohl das entsprechende Bit im TERMINAL PROFILE gesetzt war, lieferte jeder Aufruf, obwohl er genau der Spezifikation (vgl. [gsm11.14]) entsprach, den *Result code „Command data not understood by ME*“ zurück. Die Prozedur zum Versenden der Positions-Information verwendet deshalb das entsprechende GSM-Kommando, das jedes Mobiltelefon über die AT-Schnittstelle unterstützt, und wird in Abschnitt 3.2.8 beschrieben.

Ebenso verhielt es sich mit der SAT-Funktion „TIMER MANAGEMENT“, die – abhängig vom *Command Details* Datenobjekt – Timer starten oder deaktivieren soll. Auf die Unterstützung von Timern zum regelmäßigen automatischen Versand der Positions-Information wurde in der Palm-Software ganz verzichtet, weil sie in der Simulations-Phase als weniger wichtig betrachtet wurde. D.h. nach der entsprechenden Menü-Auswahl werden die SAT-Funktionen aufgerufen und die Fehlermeldung ausgegeben, und die *Event Loop* interpretiert TIMER EXPIRATION ENVELOPES, die das Mobiltelefon bei Ablauf eines Timers normalerweise senden würde, nicht.

`Boolean checkSATFunctionality()`

Die Prozedur `checkSATFunctionality` sendet ein `SSTK` *test command* an das Mobiltelefon. Falls dieses Zugriff auf *SIM Application Toolkit* Funktionen über die AT-Schnittstelle erlaubt, sendet es das TERMINAL PROFILE zurück, das Auskunft darüber gibt, welche SAT-Befehle es unterstützt, sowie den *result code* `OK`. Falls es *SIM Application Toolkit* über AT nicht unterstützt und somit auch das *test command* nicht versteht, antwortet das Mobiltelefon mit dem *result code* `ERROR`. Dementsprechend der Antwort liefert die Prozedur `true` oder `false` zurück.

`Boolean TranslateSATResponse(const char *response, char *message)`

Die Prozedur `TranslateSATResponse` sucht in einer TERMINAL RESPONSE, die als erster Parameter übergeben wird, nach einem *Result* Datenobjekt und schreibt dessen Bedeutung an die Speicherstelle, die mit dem übergebenen Pointer `message` angegeben ist. Als Ergebnis liefert sie `true`, falls das *Result* Datenobjekt keinen Fehler angezeigt hat, ansonsten `false`.

3.2.8 GSM Funktionen

`void SEND_SHORT_MESSAGE(char *userdata, char *result)`

Die Prozedur SEND_SHORT_MESSAGE sendet eine Kurznachricht, deren *user data* sie als Parameter übergeben bekommt, mittels des entsprechenden GSM-AT-Kommandos, das jedes Mobiltelefon unterstützt. Den *result code* schreibt es an die durch `result` bezeichnete Speicherstelle.

Ein TE, das über eine serielle Schnittstelle mit einem Mobiltelefon verbunden ist, hat Zugriff auf gespeicherte und neu eingehende Kurzmitteilungen (SMS) und *broadcast messages*. Die AT-Befehle, die die Prozeduren `enableMessageRouting`, `enableBC221Reception` und `disableMessageRouting` benutzen, sind in [gsm07.05] spezifiziert.

Das TE kann auf zwei unterschiedliche Arten auf eingehende Nachrichten zugreifen: Entweder informiert der TA das TE über eingegangene und auf der SIM-Karte gespeicherte Nachrichten, so dass das TE die Nachrichten mit einem Befehl auslesen kann, oder er sendet die SMS-TPDU direkt an das TE. Im letzten Fall ist das TE allerdings dafür verantwortlich, mit einer SMS-DELIVER-REPORT-TPDU zu antworten, sonst versucht das SMSC die Zustellung mehrmals hintereinander. Bei *broadcast messages* ist das nicht der Fall, da es sich um einen unbestätigten Dienst handelt.

Für den SMS-Empfang wurde deshalb die Benachrichtigung gewählt und für den Empfang von *broadcast messages* die unmittelbare Weiterleitung der TPDU durch den TA.

3.2.9 Hauptfunktionalität

Die Hauptfunktionalität der Anwendung gliedert sich (zeitlich) in drei Teile.

Im Initialisierungs-Teil, der über den Menüpunkt *Connection > Connect* angestoßen wird, wird zunächst eine serielle Verbindung zum Mobiltelefon aufgebaut. Dann wird die Kommunikation über die AT-Schnittstelle initialisiert, das Telefon auf seine Fähigkeit, SAT-Kommandos über AT auszuführen, überprüft und der Timer entsprechend des zuletzt festgelegten Intervalls gesetzt. Abschließend wird das Menü im Mobiltelefon hinzugefügt und der Empfang von neuen Kurznachrichten und *broad cast messages* aktiviert.

Der wichtigste Teil liegt im *Event Handler*. Er wird aufgerufen, wenn das ME Daten über die serielle Verbindung sendet, und verarbeitet diese Daten anschließend. Somit ist er zuständig für die Menüführung auf dem Telefon und für das dauerhafte Speichern der gewählten Einstellungen, das Speichern der geografischen Position der *serving cell*, die O_2 über den Broadcast-Kanal 221 sendet. Weiterhin stößt er der Datenübertragung zum Location Server an, und zwar entweder nach Eingang einer Kurzmitteilung von ihm, nach manueller Aufforderung durch den Benutzer oder bei Ablauf eines Timers.

Der *Event Handler* realisiert exakt das in Abschnitt 3.2.1.3 beschriebene Design.

Der Verbindungsabbau-Teil wird über den Menüpunkt *Connection > Disconnect* angestoßen. Hier wird der Empfang von Kurznachrichten und *broadcast messages* deaktiviert und die serielle Schnittstelle geschlossen.

3.2.10 Von der Simulation zur wirklichen SIM-Anwendung

Bei der Entwicklung der Palm-Software zur Simulation der SIM-Anwendung wurde bereits darauf geachtet, die Vorgänge auf der SIM-Karte möglichst genau zu simulieren. Dennoch werden bei der Portierung auf ein wirkliches SIM noch einige Details beachtet werden müssen.

3.2.10.1 Datenspeicherung

Die Benutzereinstellungen (MSISDN des Location Server, Intervall und Genauigkeit) können auf der SIM-Karte in *Elementary Files* (vgl. Abschnitt 1.6) gespeichert werden, damit sie beim Ausschalten des Mobiltelefons nicht verloren gehen.

Als Dateityp bietet sich *transparent* an. Im Gegensatz zu *linear fixed* und *cyclic* können bei diesem Dateityp unstrukturierte Daten, also beliebige Byte-Folgen abgelegt werden.

Bei der Wahl der *File ID* (entspricht dem Dateinamen) ist auf die in [gsm11.11] dokumentierten Reservierungen zu achten.

3.2.10.2 Profile Download

In der vorliegenden Simulation wird das TERMINAL PROFILE, in dem das ME dem SIM seine Fähigkeiten mitteilt (vgl. Abschnitt 1.7.2), nicht ausgewertet, da aufgrund anderer Einschrän-

kungen ohnehin nur zwei Telefon-Modelle benutzt werden konnten, deren genaue Funktionalität im Lauf der Entwicklung klar wurden.

Damit die SIM-Anwendung auf keinem Modell Fehlerzustände verursacht, sollte das TERMINAL PROFILE ausgewertet werden und beim Fehlen wichtiger SAT-Funktionen die Anwendung ganz deaktiviert werden. (Wenn das ME beispielsweise den Versand von Kurzmitteilungen per SAT nicht zulässt, kann die ganze Anwendung nicht laufen; wenn das ME dagegen nur das Auslesen des *Timing Advance* nicht unterstützt, ist das nicht schwerwiegend und wird im Location Server erkannt.)

Eine einfach zu implementierende Methode wäre, ein Referenz-TERMINAL-PROFILE abzulegen, bei dem die Bits für die Mindest-Anforderungen an die Funktionalität des ME gesetzt sind, und das bei der Initialisierung empfangene TERMINAL PROFILE mit diesem mathematisch UND zu verknüpfen. Ist das Ergebnis ungleich dem Referenz-Wert, so ist die SIM-Anwendung auf dem speziellen Mobiltelefon nicht lauffähig.

3.2.10.3 SMS- und Broadcast-Message-Empfang

Der SMS-Empfang muss von der SIM-Anwendung nicht beim ME angemeldet werden, sondern Kurzmitteilungen mit dem entsprechend gesetzten *Protocol Identifier* werden automatisch an das SIM „durchgereicht“. Der Location Server muss also bei der SMS, mit der er von der SIM-Anwendung die Daten anfordert, den *Protocol Identifier* auf den Wert *SIM Data Download* (‘3F’) setzen (vgl. Abschnitt 1.5.1).

Um den Empfang des *Cell Broadcast* Kanals zu aktivieren, auf dem O_2 die Gauß-Krüger-Koordinate der *serving cell* sendet, muss die Kanalnummer (*CB Message Identifier*) in dem *Elementary File* EF_{CBMID} eingetragen werden (vgl. [gsm11.11], 10.3.26).

3.2.10.4 Timer Expiration

Da die in der Simulation verwendeten Handy-Modelle den SAT-Befehl TIMER MANAGEMENT nicht unterstützt haben, ist in der Simulation der Ablauf eines Timers noch nicht berücksichtigt. In Abschnitt 1.7.8 ist der Inhalt des entsprechenden ENVELOPE beschrieben.

3.2.10.5 Poll Interval

Da im Kommunikationsprotokoll zwischen ME und SIM das ME immer als *Master* fungiert, sendet es in regelmäßigen Abständen (üblicherweise 30 Sekunden) ein STATUS-Kommando an das SIM („Polling“) (vgl. Abschnitt 1.6).

Da zum Abfragen der Zell- und Messdaten und dem anschließenden Versenden per SMS eine Folge von SAT-Kommandos notwendig ist, sollte das *Poll Interval* vorher auf den kleinstmöglichen Wert und anschließend wieder auf den normalen Wert gesetzt werden. Das ist mit dem SAT-Befehl POLL INTERVAL (vgl. [gsm11.14]) möglich.

Das Intervall sollte nach dem Ausführen der Operation unbedingt wieder auf einen normalen Wert gesetzt werden, da sonst die Akkulaufzeit des Mobiltelefons erheblich verkürzt wird.

3.3 Location Server

3.3.1 Genereller Ablauf

Der Location Server ist unter C++ und für das Betriebssystem Linux entwickelt worden. Die Entscheidung für die Programmiersprache fiel deshalb auf C++, weil der Algorithmus aus [sep], der eingebunden werden sollte, bereits in C++ vorlag und weil diese Sprache mächtigere Anweisungen zum bitweisen Speicherzugriff bietet, die für das Interpretieren der Zell- und Messdaten nötig waren.

Der Ablauf des Hauptprogramms ist in Abbildung 11 als Flussdiagramm dargestellt.

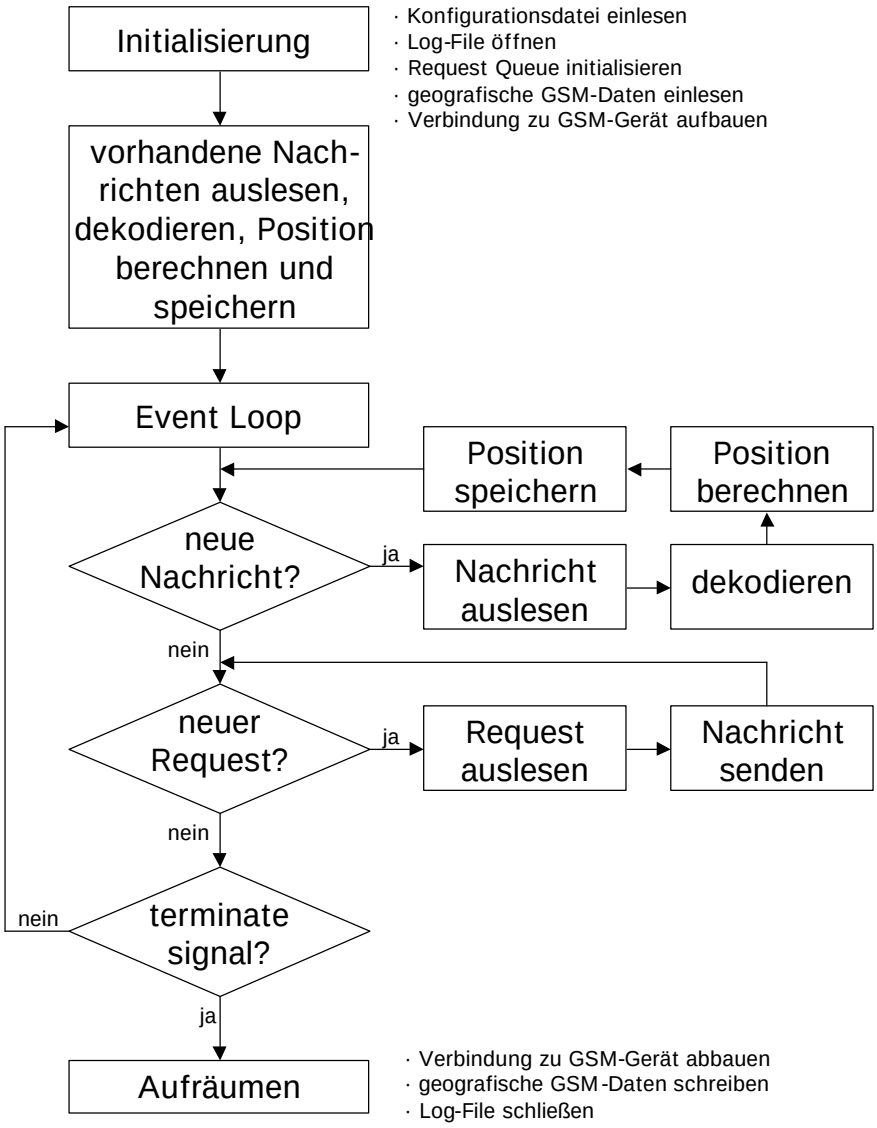

Abbildung 11: Hauptprogramm als Flussdiagramm

Im Initialisierungsteil wird die Konfigurations-Datei eingelesen, die Log-Datei initialisiert, d.h. schreibend geöffnet, die *Request Queue* (die Warteschlange für Lokalisierungs-Aufträge von

externen Anwendungen) initialisiert, die GSM-Daten (die geografischen Daten der bekannten BTS) geladen und die serielle Verbindung zum GSM-Gerät aufgebaut.

Im zweiten Schritt werden SMS-Nachrichten, die bereits vor Aufruf des Location Servers auf dem angeschlossenen GSM-Gerät gespeichert waren, ausgelesen, dekodiert, die Position des Absenders berechnet und in der Ortsdatenbank abgelegt.

Die *Event Loop* überprüft ständig das GSM-Gerät auf neue Nachrichten und die *Request Queue* auf neue Lokalisierungs-Wünsche von externen Anwendungen. Wenn eine neue Kurznachricht eingeht, wird diese ausgelesen, dekodiert, die Position des Absenders berechnet und in der Ortsdatenbank abgelegt. Wenn ein neuer Lokalisierungswunsch einer externen Anwendung vorliegt, wird dieser ausgelesen und eine Kurznachricht an das entsprechende Mobiltelefon gesendet.

Wenn das Programm in der *Event Loop* ein Terminierungssignal vom System (SIGHUP, SIGINT, SIGTERM) erhält, verlässt es die *Event Loop*, setzt die Verbindung zum GSM-Gerät zurück, schreibt die geografischen Daten der bekannten BTS und schließt die Log-Datei.

3.3.2 Konfiguration

Die Konfiguration des Location Server geschieht über eine Textdatei. Die Daten sind darin in zweilenweisen „Parameter=Wert“-Paaren abgelegt. Zeilen, die mit einem Nummernzeichen („#“) beginnen, werden ignoriert. Der Zugriff auf diese Konfigurationsdatei erfolgt über die Klasse `CConfig`.

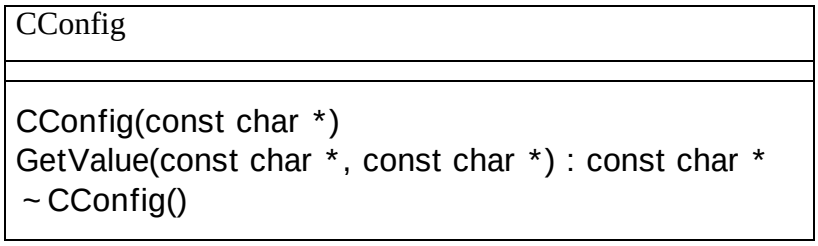

Abbildung 12: Klasse CConfig

`CConfig(const char *)`

Dem Konstruktor wird als Parameter ein Character Pointer auf den Dateinamen der Konfigurationsdatei übergeben. Er öffnet die Datei und liest die Parameter in eine interne Datenstruktur. Falls die Datei nicht geöffnet werden kann – beispielsweise weil sie nicht existiert – bleibt die interne Datenstruktur leer.

`const char *GetValue(const char *key, const char *defaultvalue)`

Diese Methode ermittelt den zugehörigen Wert zum übergebenen Parameter (key) aus der internen Datenstruktur und gibt einen Character Pointer auf ihn zurück. Falls der Parameter nicht gefunden werden kann, wird der Character Pointer auf `defaultvalue` zurückgegeben.

`~CConfig()`

Der Destruktor löscht die interne Datenstruktur.

3.3.3 Protokollierung

Der Location Server ist als Unix-Server-Prozess, also ohne Benutzer-Schnittstelle konzipiert. Daher werden Meldungen in eine Protokoll-Datei („Log-Datei“) geschrieben. Die Verwaltung der Log-Datei obliegt der Klasse `CLogFile`.

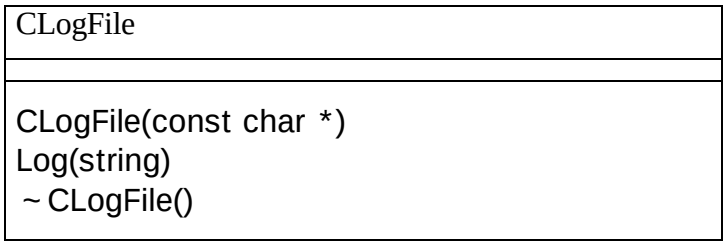

Abbildung 13: Klasse CLogFile

`CLogFile(const char *filename)`

Der Konstruktor öffnet die Log-Datei, deren Name ihm als Parameter übergeben wird. Falls die Datei nicht schreibend geöffnet werden kann, öffnet er stattdessen die Standard-Fehler-Ausgabe von Unix („`stderr`“).

`void Log(string line)`

Die Methode Log schreibt den übergebenen String, versehen mit einer genauen Zeitangabe, in die Log-Datei.

`~CLogFile()`

Der Destruktor schließt die Log-Datei.

3.3.4 Schnittstellen zu externen Applikationen

3.3.4.1 Anforderung einer Lokalisierung

Wenn eine andere Anwendung den Location Server auffordern möchte, die Lokalisierung eines Endgerätes anzustoßen, schreibt sie eine Datei in das in der Konfigurationsdatei festgelegte Spool-Verzeichnis. Der Name der Datei ist die MSISDN des Endgerätes, der Inhalt ist beliebig. Die Verwaltung des Spool-Verzeichnisses ist Aufgabe der Klasse `CRequestQueue`.

Die Klasse benutzt Dateizugriffs-Funktionen und -Prozeduren der GNU C Bibliothek `glibc`, die natürlich betriebssystemspezifisch sind. Deshalb müsste sie für eine Portierung auf andere Systeme umgeschrieben werden.

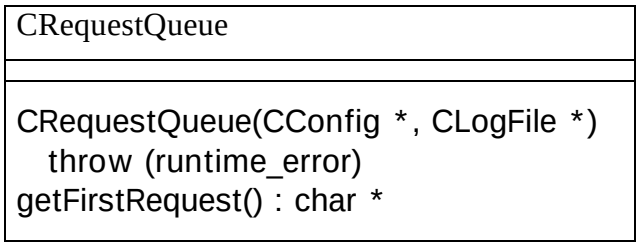

Abbildung 14: Klasse CRequestQueue

`CRequestQueue(CConfig *ConfigFile, CLogFile *LogFile) throw (runtime_exception)`

Dem Konstruktor werden Pointer auf Objekte der Klassen `CConfigFile` und `CLogFile` übergeben. Er ermittelt aus der Konfigurationsdatei den Namen des Spool-Verzeichnisses und überprüft, ob es existiert. Falls das nicht der Fall ist, versucht er, es zu erstellen. Der Vorgang wird in der Log-Datei protokolliert. Falls das Verzeichnis nicht erstellt werden kann, wird eine Exception ausgelöst.

`char *getFirstRequest()`

Diese Methode liest den Inhalt des Spool-Verzeichnisses. Die erste Datei, die nur aus Ziffern und „+“-Zeichen besteht, wird gelöscht und ihr Dateiname zurückgegeben. Falls die Datei nicht gelöscht werden kann, wird eine Exception ausgelöst. Andere Dateien werden ignoriert.

3.3.4.2 Die Ortsdatenbank

Der Location Server soll die berechneten und gespeicherten Positionen der GSM-Endgeräte anderen Anwendungen zur Verfügung stellen. Als unkomplizierte Lösung fiel die Entscheidung auf eine einfache Datenbank, auf die der Location Server als einzige Anwendung schreibend zugreift und aus der beliebig viele andere Anwendungen Positionen lesen können.

Falls der Location Server und andere Komponenten des Systems nicht auf demselben Rechner laufen, kann der Zugriff unkompliziert mittels eines verteilten Dateisystems (zum Beispiel NFS) ermöglicht werden, und es muss kein anwendungsspezifisches Protokoll zur Datenübertragung eingeführt werden.

Desweiteren hat eine Datenbank als Schnittstelle den Vorteil, dass andere Komponenten auch dann Positionen abrufen können, wenn der Location Server nicht läuft. Die Daten sind dann zwar unter Umständen nicht aktuell, aber dennoch ist dies wertvoller, als wenn eine Abfrage überhaupt nicht möglich wäre.

Zur Realisierung wurde die freie Bibliothek *gdbm* (*GNU Database Management*) gewählt, die auf praktisch jedem Linux-System zur Verfügung steht. „Frei“ bedeutet dabei im Sinne der GNU *General Public License*, Version 2, dass die Urheber die kostenlose Benutzung und Verbreitung der Bibliothek gestatten – auch zu kommerziellen Zwecken – und dass der Quellcode offengelegt ist und verändert werden darf.

gdbm ist im strengen Sinn keine Datenbank, sondern ein Verzeichnis. Eine Anwendung kann Paare von Schlüsseln und Werten willkürlicher aber fester Länge in einer *gdbm database* ablegen und Abfragen nach dem Schlüssen durchführen. Komplexe Befehle, wie sie von SQL bekannt sind, stehen nicht zur Verfügung.

Der Location Server verwendet als Schlüssel die MSISDN des GSM-Endgeräts. Als Wert wird eine Struktur verwendet, die die geographische Länge und Breite (im Gradmaß) und die Zeit der Positionsbestimmung umfasst.

Die Position wird nicht im Gauß-Krüger-Format, sondern im Gradmaß gespeichert, da dieses gebräuchlicher ist und insbesondere auch von Karten-Software verwendet wird.

Die Implementierung einer Schnittstelle zu einer mächtigeren relationalen Datenbank wie zum Beispiel *MySQL* wäre im Rahmen dieser Arbeit zu aufwendig gewesen.

Der Zugriff auf die Datenbank ist in der Klasse `CLocationDB` implementiert.

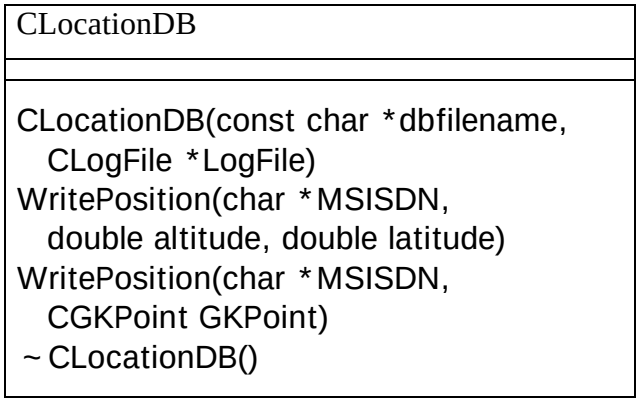

Abbildung 15: Klasse CLocationDB

`CLocationDB(const char *dbfilename, CLogFile *LogFile)`

Der Konstruktor erwartet als Parameter den Namen der Datenbank-Datei, sowie einen Pointer auf ein `LogFile`-Objekt. Er öffnet die *gdbm* Datenbank und speichert das Handle, das für die Zugriffe notwendig ist, in einer privaten Variable.

`WritePosition(char *MSISDN, double altitude, double latitude)`
`WritePosition(char *MSISDN, CGKPoint *GKPoint)`

Die Methode `WritePosition` schreibt die Position eines GSM-Endgeräts in die Ortsdatenbank. Als Parameter können neben der MSISDN entweder direkt geographische Länge und Breite übergeben werden oder der Pointer auf ein `GKPoint`-Objekt, also die Gauß-Krüger-Koordinate. Im letzteren Fall findet eine Umrechnung statt, und die erste Methode wird aufgerufen.

Zu Testzwecken wurden zusätzlich Methoden zum Lesen der geographischen Länge und Breite und des Zeitstempels implementiert, deren Benutzung sich anhand der Signatur leicht erschließen lässt.

3.3.5 Schnittstelle zum GSM-Gerät

Die Kommunikation zwischen dem Location Server und den GSM-Endgeräten läuft über Kurzmitteilungen. An den Location Server muss deshalb ein GSM-Gerät angeschlossen werden. Das kann entweder ein herkömmliches Mobiltelefon mit serieller Schnittstelle sein oder ein

sogenanntes Funkmodem, das sich von einem Mobiltelefon nur dadurch unterscheidet, dass es keine Sprechvorrichtung besitzt, dafür aber in der Regel eine Antenne, die sich entfernt aufstellen lässt.

3.3.5.1 Die Bibliothek gsmlib

Damit die Routinen zum Zugriff auf das GSM-Gerät nicht neu implementieren werden mussten, verwendet der Location Server die frei verfügbare Bibliothek *gsmlib* 1.9 von Peter Hofmann. Diese vorbildlich und vollständig objekt-orientierte Bibliothek ist für die Betriebssystem Linux und Windows geschrieben und enthält unter anderem Methoden zum Senden und Empfangen von Kurzmitteilungen sowie zum Zugriff auf den Kurzmitteilungsspeicher der SIM-Karte.

Die verwendeten Klassen werden im folgenden kurz erklärt.

(a) Klasse MeTa

Die Klasse `MeTa` repräsentiert den *Terminal Adapter* des GSM-Geräts und wickelt alle Zugriffe auf ihn ab. Sie bietet u.a. Methoden zum Ein- und Ausschalten der Benachrichtigung beim Eingang von Kurzmitteilungen an das Terminal (den PC) und zum Zugriff auf die Kurzmitteilungsspeicher. Sie ersetzt somit die eigene Implementierung der AT-Kommandos aus [gsm07.07] und [gsm07.05].

(b) Klasse SMSStore

Die Klasse `SMSStore` repräsentiert einen Kurzmitteilungsspeicher (entweder im ME oder auf der SIM-Karte). Der Zugriff auf die Kurzmitteilungen erfolgt durch einen C++-Vektor vom Typ `SMSMessage`.

(c) Klasse SMSMessage

Die Klasse `SMSMessage` repräsentiert eine Kurzmitteilung. Sie enthält u.a. Methoden zum Lesen und Schreiben aller relevanten Daten einer Kurzmitteilung (Absender, Typ, Inhalt, *Data Coding Scheme*) und zum Umwandeln von verschiedenen Codierungen. Sie erspart somit die eigene Implementierung von [gsm03.40] und [gsm03.38].

Die Klassen `SMSSubmitMessage` und `SMSDeliverMessage` sind von der Klasse `SMSMessage` abgeleitet.

3.3.5.2 Anstoßen der Übertragung der Messdaten

Die SIM-Anwendung wird vom Location Server durch eine Kurzmitteilung dazu aufgefordert, die zur Lokalisierung relevanten Daten zu versenden.

Das Senden der Kurzmitteilung erfolgt in der *Event Loop* einfach durch Anlegen eines `SMSSubmitMessage`-Objekts, Verknüpfen des Objekts mit dem `MeTa`-Objekt (Methode `setAt`) und Aufruf seiner `send`-Methode.

3.3.5.3 Empfang der Zell- und Messdaten

Die Übertragung der Zell- und Messdaten erfolgt ebenfalls per SMS.

Der Zugriff auf eingehende SMS ist in *gsmlib* so realisiert, dass dem `MeTa`-Objekt mit der Methode `setEventHandler` ein Pointer auf ein Objekt der Klasse `CEventHandler` übergeben wird. Bei eingeschaltetem *Message Routing* wird bei einer eingehenden Kurzmitteilung die Methode `SMSReceptionIndication` des `CEventHandler`-Objekts aufgerufen. Das *Message Routing*, also die Meldung eingehender Kurzmitteilungen, wird mittels der Methoden `setMessageService` und `setSMSRoutingToTA` eingeschaltet.

Die Methode `SMSReceptionIndication` des `CEventHandler`-Objekts schreibt die Speicherstelle der eingegangenen Kurzmitteilung in einen C++-Vektor. In der *Event Loop* wird dieser Wert aus dem Vektor ausgelesen und die Kurzmitteilung von der entsprechenden Speicherstelle der SIM-Karte gelesen, verarbeitet und nach erfolgreicher Verarbeitung gelöscht.

3.3.6 Berechnung der Position

Für die Berechnung der Position wurde der Algorithmus aus einem Systementwicklungsprojekt vom Februar 2002 mit einigen Anpassungen verwendet. Die genaue Dokumentation der originalen Klassen und Methoden können in [sep] nachgelesen werden. Diese Arbeit wird nur die Ideen des Algorithmus skizzieren und die vorgenommenen Anpassungen dokumentieren.

3.3.6.1 Algorithmus

Zur Zeit im Einsatz befindliche Verfahren zur Positionsbestimmung von GSM-Endgeräten verwenden als Basis für ihre Berechnung nur die Daten der Zelle, in die das Gerät eingebucht ist, und einige den *Timing Advance*. Dieses Verfahren ist vom Grundsatz her in seiner Genauigkeit sehr beschränkt: Es grenzt die geographische Position des Geräts auf eine Kreisfläche um die BTS mit einem Radius, der ungefähr der halben durchschnittlichen Entfernung zwischen zwei BTS entspricht, ein. Falls der *Timing Advance* mit berücksichtigt wird, kann der Radius der Kreisfläche (falls TA=0) oder des Rings (falls TA>0) auf 550 Meter eingegrenzt werden.

Die neue Idee an dem Verfahren, das in [sep] entwickelt wurde, ist, auch die Signalstärken zu den bis zu sechs umliegenden BTS in die Berechnung mit einzubeziehen, da sie ein ungefähres Maß für die Entfernung zwischen Endgerät und BTS sind.

Ein Problem, was es dabei vor der eigentlichen Berechnung zu lösen gilt, ist, die umliegenden BTS zu identifizieren, denn von ihnen liegen nur die Kanalnummern vor, die noch nicht auf die Zelle selbst schließen lassen. Das verwendete Verfahren löst das Problem, indem es aus der Datenbasis zunächst alle diejenigen Zellen auswählt, die die gesuchte Kanalnummer haben. In einem nächsten Schritt wählt es daraus diejenige Zelle aus, die den geringsten Abstand zur *serving cell* hat. Geht man davon aus, dass die Datenbasis vollständig und richtig ist, so ist die Auswahl offensichtlich korrekt.

Wie in Abschnitt 1.3.2 beschrieben, steht eine BTS in der Realität nicht in der Mitte einer Zelle, sondern eine BTS spannt mittels gerichteter Antennen bis zu drei Zellen auf, an deren Rand sie somit jeweils steht. Im nächsten Schritt werden deshalb die gefundenen Zellen (*serving cell* und Nachbarzellen) zu BTS zusammengefasst. Die Realisierung dieses Schrittes ist sehr O_2-spezifisch, da sie für die Erkennung, welche Zellen zu derselben BTS gehören, die *Cell Identifier*

betrachtet. (Bei O_2 sind die letzten vier Dezimalstellen der Cell Identifier von Zellen, die zu einer gemeinsamen BTS gehören, gleich. Das ist von O_2 willkürlich so festgelegt!)

Abschließend wird dann die eigentliche Berechnung durchgeführt. Zu diesem Zeitpunkt liegen die Koordinaten und die Signalstärken (RXL-Werte) von mindestens einer und maximal sieben BTS vor. Die Position des Endgeräts ist der Mittelpunkt der Koordinaten der gefundenen BTS, wobei die einzelnen BTS je nach Signalstärke unterschiedlich gewichtet werden.

Folgende Funktion wird dabei zur Gewichtung verwendet:

$$\text{weight}(RXL) = \sqrt{10^{\frac{RXL}{10}} \cdot 0{,}001}$$

Die Potenzierung des RXL-Werts ist notwendig, da er ein logarithmisches Maß für die am Gerät empfangene Leistung ist. Das Ziehen der Quadratwurzel leitet sich aus dem quadratischen Zusammenhang zwischen Entfernung und Signalstärke her. (Das Signal breitet sich mit Lichtgeschwindigkeit gleichmäßig in alle Richtungen aus. Die Energie verliert sich also gleichmäßig auf der Oberfläche einer Kugel, deren Radius mit Lichtgeschwindigkeit wächst. Die Kugeloberfläche hängt quadratisch vom Kugelradius ab.) Der konstante Faktor beeinflusst die Verhältnisse der Gewichte untereinander nicht und wurde eingeführt, damit die Zahlen nicht zu groß werden.

3.3.6.2 Decodierung der Kurzmitteilung

Die Interpretation der empfangenen Kurzmitteilung und das Ablegen der Werte in die Datenstrukturen, die der Algorithmus erwartet, geschieht in einem neu geschriebenen Konstruktor der Klasse `CGSMData`. (Die Klasse `CGSMData` ist die Repräsentation der rohen Zell- und Mess-Daten.)

`CGSMData::CGSMData(const char *RawSMSData, unsigned char length) throw (runtime_error)`

Der neue Konstruktor iteriert in einer `for`-Schleife über alle vorkommenden SIMPLE-TLV-Datenobjekte und ruft abhängig vom gefundenen *tag* jeweils die private Methode `ParseLocationInformation`, `ParseNetworkMeasurementResults`, `ParseBCCHChannelList`, `ParseTimingAdvance` bzw. `ParseGKPos` auf, die die Datenobjekte entsprechend der Spezifikation parsen (vgl. Abschnitte 1.7.4 und 1.5.1) und die Werte in die privaten Variablen speichern.

Abschließend erfolgt noch das Nachschlagen der Kanäle in der *BCCH channel list* anhand der Indizes.

Eine Ausnahme wird erzeugt, wenn kein *Location Information* Datenobjekt gefunden werden kann oder Fehler beim Parsen eines Datenobjekts auftreten. Ein Fehlen der anderen Objekte wird toleriert, wobei die entsprechenden Variablen dann leer bleiben. Das ist zum Beispiel der Fall, wenn der Benutzer am Mobiltelefon eingestellt hat, dass nur die Daten der *serving cell* und keine Mess-Daten übertragen werden sollen.

3.3.6.3 Anpassungen der bestehenden Klassen

(a) CString-Objekte

Die Klasse `CString` ist Teil der *Microsoft Foundation Classes* (MFC) und nur unter Windows-Betriebssystemen verfügbar. Die entsprechenden Objekte aus dem vorhandenen Quellcode wurden deshalb durch Character-Arrays ersetzt und die benutzten Methoden durch entsprechende ANSI-C-konforme (`strcpy`, `strcat`, `sprintf`, etc.).

(b) Protokollierung

Bei den Klassen, die Einträge in das Protokoll vornehmen sollen, wurde dem Konstruktor der Pointer auf ein `CLogFile`-Objekt übergeben, der in einer privaten Variable gespeichert wird.

(c) Ausnahmen

Bei einigen Methoden wurde die sinnvolle Generierung von *exceptions* hinzugefügt. (Die originalen Sourcen haben teilweise einfach eine Meldung auf die Standardausgabe geschrieben.)

(d) Klasse CCell: Country Code und Network Code

Die Klasse `CCell` wurde dahingehend erweitert, dass der *Mobile Country Code* (MCC) und *Mobile Network Code* (MNC) einer Zelle auch gespeichert werden. Das ebnet den Weg für den Fall, dass zu einem späteren Zeitpunkt mehrere verschiedene Netzbetreiber unterstützt werden sollen.

(e) CDataSource::IsInAvailableData

In der Methode `IsInAvailableData`, die das Vorhandensein einer gegebenen Zelle in der Datenbasis prüft, werden nur noch *Location Area Code* und *Cell ID* überprüft, da diese beiden Parameter die Zelle bereits eindeutig identifizieren.

Kanalnummer und Gauß-Krüger-Koordinaten werden nicht mehr verglichen, da *SIM Application Toolkit* die Kanalnummer der *serving cell* nicht übertragen kann und außerdem nicht davon auszugehen ist, dass der Empfang der *cell broadcast messages* immer funktioniert.

(f) CDataSource::CompleteCell

Der Klasse `CDataSource` wurde die neue Methode `CompleteCell` hinzugefügt, die eine gegebene Zelle um die Kanalnummer und Gauß-Krüger-Koordinate aus der Datenbasis ergänzt.

(g) CDataSource::CalculateRelevantStationPoints

Die Methode `CalculateRelevantStationPoints` wurde dahingehend erweitert, dass sie zu Beginn überprüft, ob die *serving cell* schon in der Datenbasis vorhanden ist. Falls nicht und falls die Daten vollständig sind (LAC, CID, Kanalnummer und Gauß-Krüger-Koordinaten) – was bei dem vorliegenden System niemals passieren wird, da *SIM Application Toolkit* die Kanalnummer der *serving cell* nicht liefern kann –, wird die Zelle in die Datenbasis eingefügt.

Falls der *serving cell* die Gauß-Krüger-Koordinate fehlt und sie nicht in der Datenbasis vorhanden ist, wird eine *exception* ausgelöst, denn unter dieser Bedingung kann die Position nicht berechnet werden.

Anschließend werden ggf. Kanalnummer und/oder geographische Position der Zelle aus der Datenbasis ergänzt (siehe (f)).

(h) Klasse CPositioning

Die Klasse `CPositioning`, die den Hauptschritt des Algorithmus (Berechnung des Mittelpunkts der BTS) enthält, wurde vereinfacht. So wurde zum Beispiel die gesamte Funktionalität zur regelmäßigen Berechnung entfernt, da der Algorithmus im vorliegenden System nur einmal pro eingehender Kurzmitteilung ausgeführt werden soll.

In den originalen Sourcen wurden die notwendigen Daten von einem `CGSM`-Objekt eingelesen, das ein lokal angeschlossenen GSM-Gerät repräsentiert. Hier wurde abstrahiert: In der modifizierten Version erwartet die Methode `calculatePosition` nun als Parameter ein `CGSMData`-Objekt, das die Zell- und Messdaten enthält, und funktioniert somit unabhängig davon, woher diese Daten stammen.

Kapitel 4

Bewertung und Ausblick

4.1 Unzulänglichkeiten der SAT-Implementierung

Bei der derzeitigen Realisierung als Palm-Anwendung, die die *SIM Application Toolkit* Funktionen über die AT-Schnittstelle anspricht, sind bei den verwendeten Telefon-Modellen Siemens C35 und S35 einige Mängel in der Implementierung der SAT-Befehle aufgefallen. Ob der Zugriff durch eine wirkliche SIM-Applikation besser funktioniert, ist schwer einzuschätzen.

- Bei dem Befehl GET INPUT wird der übergebene *Default Text* dem Benutzer nicht sofort angezeigt, sondern erst nachdem dieser die erste Taste gedrückt hat. Das ist für die eigentliche Funktionalität nicht sehr tragisch, lässt aber vermuten, dass der Hersteller wenig Wert auf die fehlerfreie Implementierung von SAT gelegt hat.
- Die Befehle TIMER MANAGEMENT und SEND SHORT MESSAGE werden nicht unterstützt, obwohl im TERMINAL PROFILE das entsprechende Bit für die Fähigkeit gesetzt ist. Das ME liefert in seiner TERMINAL RESPONSE den Fehlercode „Command data not understood by ME“ zurück.
- Der Befehl PROVIDE LOCAL INFORMATION liefert den Timing Advance nicht zurück. (Dass das Mobiltelefon diese Funktionalität nicht unterstützt, ist im korrekt TERMINAL PROFILE angegeben.)

4.2 Mögliche Verbesserungen

4.2.1 Systemweit: USSD

Die Übertragung der Daten zwischen Location Server und SIM-Anwendung per SMS verursacht für den Benutzer sehr hohe Kosten und wird für den normalen Betrieb nicht tragbar sein. Besonders unpraktisch ist zudem, dass der Betreiber des Dienstes die Kosten für die Kurzmitteilungen, die vom Endgerät versandt werden, nicht übernehmen kann und somit dem Benutzer Kosten entstehen, wenn ein anderer ihn orten will.

Die Möglichkeit, die Daten per USSD (vgl. Abschnitt 1.5.2) zu übertragen sollte daher weiter verfolgt werden, sobald ein Mobilfunkanbieter externen Dienstleistern ein dazu notwendiges *Gateway* anbietet.

4.2.2 SIM-Anwendung: Location Update

Die SIM-Anwendung könnte sich mittels *Event download* über jedes *Location update* informieren lassen und die geänderten Zellinformationen (die bei dem Event mitübertragen werden) zwischenspeichern, um sich bei der Versendung der Daten an den Location Server eine der drei Abfragen zu sparen.

Allerdings ist die Abfrage der *Network Measurement Results*, die sich praktisch ständig ändern, und des *Timing Advance* dennoch nötig, denn für beides gibt es kein Event.

4.2.3 Location Server

4.2.3.1 Algorithmus

Der Schritt des Algorithmus, der aus allen in Frage kommenden Nachbarzellen die mit der geringsten Entfernung auswählt, bedarf noch etwas Nachbesserung für den Fall, dass nicht die Daten aller Zellen vorliegen.

Derzeit wird eine nächste Nachbarzelle genau dann verworfen, wenn ihre Entfernung größer ist als das 3000-fache des *Timing Advance* (wobei der TA zu 1 korrigiert wird, falls er gleich 0 ist.) Dieses Entscheidungskriterium scheint nicht sinnvoll gewählt zu sein, denn es führt bei der Lokalisierung eines Endgeräts, das sich in weniger als 3000 Meter Entfernung zu den verfügbaren Zellen der Datenbasis befindet, dazu, dass falsche BTS in die Berechnung miteinbezogen werden.

Ein besseres Entscheidungskriterium wäre

$$(TA + 1) \cdot 550\ \text{m} + c$$

wobei c so groß gewählt werden muss, dass in ländlichen Gegenden weiter entfernte BTS nicht fälschlicherweise verworfen werden.

Eine weitere Verbesserung wäre, das Verhältnis der Signalstärken von *serving cell* zu fraglicher Nachbarzelle zu berücksichtigen, denn wenn die Signalstärken ungefähr gleich sind, ist nicht davon auszugehen, dass die Nachbarzelle viel weiter entfernt ist als die *serving cell*. Unterscheiden sich die beiden Signalstärken, dürfte das Verhältnis der beiden etwa gleich dem Verhältnis der Entfernungen der beiden Zellen zum Endgerät sein.

Ein noch besseres Entscheidungskriterium wäre demnach

$$(TA + 1) \cdot 550\ \text{m} \cdot c \cdot \sqrt{10^{\frac{RXL(sc) - RXL(nc)}{10}}}$$

wobei c ein Toleranzfaktor größer 1 ist, beispielsweise 1,5.

4.2.3.2 Ortsdatenbank in SQL

Die momentane Realisierung der Ortsdatenbank als GDBM-Datei ist Unix-spezifisch. Das macht die Interoperabilität zu anderen Komponenten, die nicht unter Unix laufen, unmöglich.

In einer Folge-Version könnte die `CLocationDB`-Klasse so umgeschrieben werden, dass sie die geographischen Positionen der GSM-Geräte in einer SQL-Datenbank ablegt, auf die Anwendungen jeder Plattform zugreifen können.

Das würde zudem auch komplexere Anfragen zulassen, wie zum Beispiel „liefere mir alle Benutzer, deren Abstand zur Position (*x*,*y*) nicht größer als *z* ist“ oder „liefere mir alle Benutzer, deren letzte Positionsbestimmung nicht länger als *n* Minuten zurückliegt“, um sie neu zu lokalisieren.

Desweiteren existieren zu SQL für nahezu alle Sprachen Programmierschnittstellen (API, *Application Programming Interface*), so dass die Daten beispielsweise auch durch webbasierte Technologien wie CGI und PHP genutzt werden könnten.

4.3 Praxistauglichkeit

Das vorliegende System ist natürlich recht unpraktisch zu bedienen, weil der mobile Teil dadurch, dass ein PDA mit dem Mobiltelefon verbunden sein muss, die Mobilität erheblich einschränkt und somit der Benutzer zur Positionsbestimmung aktiv werden muss. Es ist daher eher zur Demonstration der *SIM Application Toolkit* Funktionen geeignet und als Vorarbeit für eine spätere Realisierung auf einer SIM-Karte gedacht.

Wenn man sich aber die in dieser Arbeit umgesetzte Funktionalität als tatsächliche SIM-Anwendung vorstellt, bietet diese Art der Positionsbestimmung einige Vorteile gegenüber bisher üblichen.

So ist sie zum einen wesentlich preisgünstiger und handlicher für den Benutzer als GPS, da neben dem Mobiltelefon kein weiteres Gerät angeschafft und herumgetragen werden muss. Über ein Mobiltelefon verfügt ohnehin jedes Mitglied einer mobilen Community.

Der eingesetzte Algorithmus arbeitet genauer als auf dem Markt eingesetzte, da er nicht nur die *serving cell* auswertet, sondern auch die Signalstärken zu den umliegenden BTS. Der Gewinn ist in ländlichen Gegenden, wo BTS nicht so dicht aufgestellt sind wie in Großstädten, besonders groß.

Mit der Genauigkeit von GPS dagegen kann sich der Algorithmus bei weitem nicht messen: Obwohl er bei Tests im Rahmen des Systementwicklungsprojekts (vgl. [sep]) in 95% aller Fälle die Position bis auf maximal 100 Meter Abweichung bestimmt hat, wich der berechnete Standort im schlimmsten Fall um mehr als 200 Meter vom tatsächlichen ab. Bei GPS ist eine maximale Abweichung von 100 Metern garantiert und in vielen Fällen eine Genauigkeit bis auf 10 Meter möglich.

Eine Weiterentwicklung des Verfahrens (beispielsweise eine Erweiterung der Datenbasis um tatsächlich gemessene Referenz-Messwerte) kann die Exaktheit möglicherweise noch verbessern. Man darf allerdings nicht vergessen, dass die Positionsbestimmung in Mobilfunknetzen bei deren Entwicklung nicht vorgesehen war.

Als besondere Hürde hat sich während des Projekts die mangelnde Kooperativität der Mobilfunkbetreiber herausgestellt. Eine enge Kooperation wäre notwendig gewesen, weil die GSM-Norm nur wenige Schnittstellen nach außen vorsieht und somit dem Betreiber ein Quasi-Monopol für Mehrwertdienste schafft.

In dem hier entwickelten System ist die Zusammenarbeit an zwei sehr wichtigen Punkten notwendig: Erstens müssen für den sinnvollen Betrieb die vollständigen Daten über alle Sendemasten samt deren geographischer Position zur Verfügung stehen, und zweitens muss der Mobilfunkanbieter die mit der Anwendung produzierten SIM-Karten für sein Netz freischalten. Die endgültige Realisierbarkeit des in dieser Arbeit vorgestellten Problems wird also maßgeblich von der Unterstützung der Mobilfunkbetreiber abhängen.

Abkürzungen

API	Application Programming Interface
ASN.1	Abstract Syntax Notation Number 1
AuC	Authentication Centre
COSMOS	Community Online Services and Mobile Solutions
BER	Basic Encoding Rules
BCCH	Broadcast Control Channel
BSC	Base Station Controller
BTS	Base Transceiver Station
CB	Cell Broadcast
CBC	Cell Broadcast Centre
CID	Cell Identificator
COSMOS	Community Online Services and Mobile Solutions
DBMS	Database Management System, Datenbanksystem
DF	Dedicated File
EEPROM	Electrical Erasable Read Only Memory
EF	Elementary File
EIR	Equipment Identity Register
ETSI	European Telecommunications Standards Institute
GDBM	GNU Database Management
GSM	Global System for Mobile Communications
GMSC	Gateway Mobile Switching Centre
HLR	Home Location Register
IMEI	International Mobile Equipment Number
IMSI	International Mobile Subscriber Number
IR	Infrarot
ISDN	Integrated Services Digital Network
LA	Location Area
LAC	Location Area Code
MCC	Mobile Country Code
ME	Mobile Equipment, Mobiltelefon
MF	Master File
MFC	Microsoft Foundation Classes

MMI	Man to Machine Interface
MNC	Mobile Network Code
MS	Mobile Station
MSC	Mobile Switching Centre
MSISDN	Mobile Station ISDN Number
NFS	Network File System
PC	Personal Computer
PIN	Personal Identification Number
PDA	Personal Digital Assistant
PDU	Protocol Data Unit
PSTN	Public Switched Telephone Network
RXL	Reception Level
RXLEV	Reception Level
RXQUAL	Reception Quality
SAT	SIM Application Toolkit
SIM	Subscriber Identity Module
SM MT	Short Message Mobile Terminated
SM MO	Short Message Mobile Originated
SMS	Short Message Service
SMSC	Short Message Service Centre
SMS-G	Short Message Service Gateway
SQL	Simple Query Language
TA	(1) Timing Advance
	(2) Terminal Adapter
TE	Terminal Equipment
TLV	Tag-Length-Value
TPDU	Transport PDU
USSD	Unstructured Supplementary Services Data
VLR	Visitor Location Register

Literatur

[cosmos] Reichwald, Ralf; Krcmar, Helmut;Schlichter, Johann; Baumgarten, Uwe: Community Online Services And Mobile Solutions, Projektstartbericht des Verbundvorhabens COSMOS; München; 2001

[mvs] Baumgarten, Uwe: Skript zur Vorlesung „Mobile Verteilte Systeme“; SS 2001; Fakultät für Informatik, Technische Universität München (2001)

[gsm1] Wood, Lloyd: GSM overview (2000) http://www.ee.surrey.ac.uk/Personal/L.Wood/constellations/tables/gsm.html

[gsm2] Eberspächer, Vögel, Bettstetter: GSM, Global System for Mobile Communication; Stuttgart; B. G. Täuber; 2000

[gsm3] Hüttisch, Norbert: Das GSM-Glossar http://www.nobbi.com/glossar.htm

[chip] Rankl, Wolfgang; Effing, Wolfgang: Handbuch der Chipkarten; Wien, München; Hanser, 1999

[gsm11.11] ETSI TS 100 977, GSM 11.11 version 8.6.0, Digital cellular telecommunications system (Phase 2+), Specification of the Subscriber Identity Module – Mobile Equipment (SIM-ME) interface (2001)

[gsm11.14] ETSI TS 101 267, GSM 11.14 version 8.9.0, Digital cellular telecommunications system (Phase 2+), Specification of the SIM Application Toolkit for the Subscriber Identity Module – Mobile Equipment (SIM-ME) interface

[gsm04.08] ETSI TS 300 940, GSM 04.08 version 7.4.2, Digital cellular telecommunications system (Phase 2+), Mobile radio interface layer 3 specification

[gsm05.08] ETSI TS 300 578, GSM 05.08 version 4.22.1, Digital cellular communications system (Phase 2); Radio subsystem link control

[gsm22.090] ETSI TS 122 090, 3GPP TS 22.090 version 4.0.0, Digital cellular communications system (Phase 2); Unstructured Supplementary Service Data (USSD) – Stage 1

[gsm07.07] ETSI TS 100 916, GSM 07.07 version 7.7.0, AT command set for GSM Mobile Equipment

[gsm07.05] ETSI TS 100 585, GSM 07.05 version 7.0.1, Equipment (DTE – DCE) interface for Short Message Service (SMS) and Cell Broadcast Service (CBS)

[gsm03.38] ETSI TS 100 900, GSM 03.38 version 7.2.0, Alphabets and language-specific information

[sep] Marcus Adlwart, Stefan Claus, Christian Pfaller, Christian Wied: Terminalbasierte Standortbestimmung in GSM-Netzen: Entwicklung eines Verfahrens zur Positionsbestimmung. Systementwicklungsprojekt, Fakultät für Informatik, Technische Universität München (2002)

[palmos] Foster, Lennon R.: Palm OS Programming Bible; New York; Hungry Minds, Inc., 2000

[siemens] Siemens: Manual Reference AT Command Set for the SIEMENS Mobile Phones S35i, C35i, M35i

[gdbm] GDBM Dokumentation
http://www.mit.edu:8001/afs/athena.mit.edu/project/gnu/doc/html/gdbm_toc.html

[gsmlib] Bibliothek gsmlib
http://www.pxh.de/fs/gsmlib/index.html

[gk] Umrechnung der Gauß-Krüger-Koordinaten
http://www.tipps.delphi-source.de/sonstiges/tut20020202-1.shtml