

# Universität Stuttgart

## Fakultät Informatik

**Prüfer:** Prof. Dr. Kurt Rothermel  
**Betreuer:** Dipl. inf. Peter Coschurba

**begonnen am:** 01.09.2000  
**beendet am:** 28.02.2001

**CR-Klassifikation:** C.2.4, H.4.3

Studienarbeit Nr. 1800

### Design und Entwicklung einer Anwendung für GeoMail und GeoMessages

Alexander Till

Institut für Parallele und Verteilte Höchstleistungsrechner  
Breitwiesenstr. 20-22  
D-70565 Stuttgart

### Zusammenfassung

Mit der Verbreitung von mobile Rechnern, die mit Positionssensoren ausgestattet sind, entsteht die Möglichkeit die geografische Position von mobilen Benutzern sehr genau zu bestimmen. Gleichzeitig können die mobilen Geräte über drahtlose Netzwerke mit anderen Rechnern kommunizieren. Damit entsteht eine neue Welt von Anwendungsmöglichkeiten, die davon profitieren, die eigene Position und die Position anderer Objekte in der Umgebung zu kennen. Dem Benutzer können Informationen über seine Umgebung, über verfügbare Netzwerke, Services und andere mobile oder stationäre Rechner angeboten werden. Diese Entwicklungen lassen die Möglichkeit zu, Nachrichten, z.B. Emails, an bestimmte geografische Zielgebiete zu senden, unabhängig von der Kenntnis, welche Empfänger sich dort aufhalten. Das Konzept des Senden von Nachrichten aufgrund geografischer Adressinformation wird GeoCast genannt. Diese Studienarbeit befaßt sich mit dem Design eines Systems mit dem es möglich sein soll, Nachrichten auf der Basis von geografischen Informationen zu senden. Bestehende Grundlagen zur Positionierung, zur Adressierung und Routingverfahren werden vorgestellt. Mögliche Anwendungsszenarien werden aufgezeigt. Auf der Basis dieser Anwendungsszenarien wird untersucht, welche Methoden und Konzepte wünschenswert sind, um einen brauchbaren GeoCast-Service anzubieten. Die Implementierung eines prototypischen GeoCast Systems, das eine geeignete Auswahl der Adressierungsmöglichkeiten realisiert, und die Implementierung einer Beispielanwendung sind die Schwerpunkte der Studienarbeit.

Diese Studienarbeit ist in engem Zusammenhang mit der Diplomarbeit von Gerald Grau<sup>1</sup> zu sehen, dessen Aufgabe es ist, den "Unterbau", das lokationsabhängige Routing der Nachrichten, zu realisieren.

---

<sup>1</sup>Diplomarbeit: "Entwicklung effizienter Methoden zur geographischen Nachrichtenweiterleitung", Gerald Grau

# Inhaltsverzeichnis

<b>1</b>	<b>Einleitung</b>	<b>4</b>
<b>2</b>	<b>Anwendungsszenarien</b>	<b>5</b>
<b>3</b>	<b>Grundlegende Technologien</b>	<b>11</b>
3.1	Broadcast, Unicast, Multicast, Geocast . . . . .	11
3.2	Geografische Routingverfahren . . . . .	11
3.3	Bestimmung der Position . . . . .	12
3.4	Koordinatensysteme . . . . .	15
3.5	Architekturansätze . . . . .	16
<b>4</b>	<b>Adressierungskonzepte</b>	<b>18</b>
4.1	Geometrische Objekte als Adressgebiet . . . . .	18
4.2	Logische Bezeichner als Adresse . . . . .	20
4.3	Context Awareness-Kriterien . . . . .	20
4.4	Zeitkomponente . . . . .	22
4.5	Zusammensetzung geografischer Objekte . . . . .	22
4.6	Verknüpfung geografischer Objekte . . . . .	23
<b>5</b>	<b>Bewertung und Auswahl der Konzepte</b>	<b>24</b>
<b>6</b>	<b>Spezifikation GeoClient</b>	<b>28</b>
6.1	Überblick . . . . .	28
6.2	Aspekt Empfang . . . . .	28
6.3	Aspekt Senden . . . . .	29
6.4	Schnittstelle zum Routing-System . . . . .	29
6.5	Komponenten mit denen ein GeoClient kommunizieren muß . . . . .	30
6.6	Algorithmen . . . . .	30
<b>7</b>	<b>Spezifikation Anwendung</b>	<b>33</b>
7.1	Überblick . . . . .	33
7.2	Benutzungsoberfläche . . . . .	33
7.3	Datenstrukturen . . . . .	35
<b>8</b>	<b>Entwurf</b>	<b>37</b>
8.1	Entwurfspakete . . . . .	37
8.2	Kommunikationsprotokoll zwischen GeoApplication und GeoClient . . . . .	52
8.3	Kommunikationsprotokoll zwischen GeoClient und GeoNode . . . . .	54
<b>9</b>	<b>Fazit - Ausblick</b>	<b>57</b>
<b>A</b>	<b>XML Messages - Schnittstelle zwischen GeoClient und GeoNode</b>	<b>59</b>

# 1 Einleitung

## Motivation

Das Internet, ein Verbund von Computernetzwerken, ist ständig am wachsen und beeinflusst unser berufliches und alltägliches Leben zunehmend. Eine Besonderheit der nahen Vergangenheit dieser Entwicklung ist die Miniaturisierung von Rechnern und die wachsende Verfügbarkeit drahtloser Kommunikationsnetzwerke. Rechner werden immer mobiler und bleiben dabei ständig und zu akzeptablen Kosten, in Datennetze eingebunden. Parallel dazu gibt es immer genauere Systeme (z.B. [GPS]), die eine Bestimmung der Position eines mobilen Gerätes erlauben. Bis vor kurzem militärischen Zwecken vorbehalten, stehen diese Dienste mittlerweile für zivile Anwendungen zur Verfügung.

Diese Kombination von Mobilität und Positionsinformationen läßt eine Fülle neuartiger Anwendungsszenarien zu. Verschiedene einzelne Ansätze, die mit diesen neuen Möglichkeiten experimentieren, sind im Entstehen. Um global brauchbare Anwendungen zu realisieren, wird es notwendig sein, eine passende Infrastruktur zu schaffen, die an Mobilität von Rechnern und Benutzern, an ständige Verfügbarkeit von Daten, Informationen und das Wissen über die geografischen Positionen sämtlicher Objekte, angepasst ist. Die Realisierung einer solchen Infrastruktur ist ein Ziel des Forschungsprojekts Nexus<sup>1</sup> an der Universität Stuttgart.

Ergänzend zu den bestehenden Kommunikationsmethoden wie broadcast (Senden an alle Empfänger in einem (Sub-)Netz), unicast (Senden an einen einzigen Empfänger) und multicast (Senden an eine Gruppe) wurde eine neue Methode eingeführt, die das Senden an ein geografisches Gebiet realisiert. Für diese Methode hat sich die Bezeichnung GeoCast etabliert.

## Ziele der Studienarbeit

Die Entwicklung einer Adressierungskomponente für das GeoCast-System ist ein Hauptziel dieser Arbeit. Die Adressierungskomponente soll dabei über entsprechende Schnittstellen mit anderen Nexus-Komponenten zusammenarbeiten. Zuerst soll ein Überblick über mögliche Adressierungskonzepte gegeben werden. Diese Konzepte sollen bewertet werden, wobei für die vorgesehene Implementation eine geeignete Auswahl getroffen werden soll. Zusätzlich soll eine Anwendung entwickelt werden, die das Versenden und Empfangen von GeoMail und GeoMessages erlaubt.

---

<sup>1</sup>Universität Stuttgart, Fakultät Informatik, IPVR, Forschungsprojekt Nexus, <http://www.nexus.uni-stuttgart.de/>

## 2 Anwendungsszenarien

Keine neue Technologie hat Zukunft ohne eine breite Basis von Anwendungsmöglichkeiten. Vor der Erfindung der elektrischen Sprachübermittlung (Funk, Telefon) war die verbale Kommunikation zwischen Menschen nur möglich, wenn diese sich gegenseitig innerhalb ihrer Hör- bzw. Rufweite befanden. Bevor die Möglichkeit bestand, email (oder Faxe) zu senden, konnten Texte und Bilder nur auf Papier, per normaler Post, ausgetauscht werden. Dass heißt: eine große Anzahl von traditionellen Formen des zwischenmenschlichen Austausches, wurden durch moderne Kommunikationsmittel vereinfacht, ergänzt oder gar ersetzt. Die Ersetzung durch elektronische Medien hatte aber einen Nachteil: Es sind meist nur Punkt-zu-Punkt Verbindungen möglich. Zum Vergleich: Während eine Person zu einer Versammlung von Menschen sprechen kann, ist es (auch technisch) schwieriger mit einer Gruppe von Personen auf elektronischer Ebene (telefonisch oder per email) in Kontakt zu treten. Vor allem wenn die Namen, Telefonnummern oder Adressen der Einzelnen aus einer Gruppe nicht bekannt sind, und die Gruppe beispielsweise durch ihren Aufenthaltsort charakterisiert ist. Auch Chat-Systeme bieten hier keine Alternative, da die Teilnehmer explizit aktiv in Bereitschaft sein müssen (indem sie ein Chatfenster eines bestimmten Kanals geöffnet haben) um Nachrichten zu empfangen. Damit sind einige Funktionen, wie beispielsweise das Mitteilen von Information via Megafon oder das Verteilen von Wurfsendungen an bestimmte Ortsbezirke, auf elektronischer Ebene noch nicht möglich. Diese Lücke kann von einem GeoCast-System geschlossen werden. In diesem Kapitel werden Beispiele gesammelt, die geeignet sind darzustellen welche bekannten Kommunikationsformen, die bisher noch nicht oder nur unzureichend auf elektronischer Ebene umgesetzt sind, mittels eines GeoCast-Systems realisiert werden können. Es kann sein, daß nach Bereitstellung der Möglichkeit ortsbasierende Nachrichten zu verschicken, weitere Anwendungsgebiete entdeckt werden, an die heute noch keiner denkt. Diese Entwicklung konnte schon bei vielen technischen Neuerungen beobachtet werden. Beispielsweise könnten Spiele erfunden werden, die auf den Aufenthaltsorten der Mitspieler basieren, etc.

### **Ankündigungen, Ortsbedingte Informationen**

Ankündigungen von Veranstaltungen, Vorträgen, Vorlesungen die sehr kurzfristig und nur für eine bestimmte geografisch eingrenzbar Personengruppe von Interesse sind, können durch eine Geocast-Nachricht realisiert werden. Beispielsweise sollen alle Personen in einem Gebäude über einen interessanten Vortrag eines Gastdozenten informiert werden. Zu diesem Zweck wird eine Nachricht an das geografische Gebiet, welches das Gebäude einnimmt, gesendet. Ereignisse, die sich in Beziehung zu einem bestimmten Gelände befinden, sind besonders geeignet. (siehe Abbildung 1)

Beispiele: Attraktionen in einem Unterhaltungspark, Ereignisse auf der Expo, Veranstaltungen auf dem Universitätsgelände etc. Bisher realisiert durch Plakate, Handzettel, Mundpropaganda, statische Anwendungen (Expo) die in der Nähe des Ereignisortes verteilt werden, sind geografisch adressierte Nachrichten ideal für gezielte Ankündigungen.

Abbildung 1: Verteilung ortsbedingter Informationen an die relevante Umgebung

## Verkehrskontrolle

Für die Steuerung von Verkehr ist die Möglichkeit, Nachrichten an bestimmte Gebiete zu senden, äußerst nützlich. Beispielsweise könnten gezielt Fahrzeuge in einem bestimmten Bereich dazu aufgefordert werden, eine Umleitung zu benutzen oder eine bestimmte Höchst- oder Mindestgeschwindigkeit einzuhalten. Fahrzeuge, die sich einem Stau nähern, können rechtzeitig gewarnt werden. Verkehrshinweise, wie man sie heute über das Radio empfängt, können gezielt, nur an die Empfänger gesendet werden, die aufgrund ihrer Position (Bewegungsrichtung und Geschwindigkeit) davon betroffen sind (siehe Abbildung 2). Bisher wird ein solcher Service nur durch das Radio realisiert. Der Nachteil dabei ist, daß auch Zuhörer, die gar nicht betroffen sind, durch Verkehrshinweise belästigt werden. Einige Ansätze im Bereich Autonavigation versuchen dem Fahrer Verkehrshindernisse mitzuteilen. Dies funktioniert meist nur bei längerfristig bekannten Ausnahmefällen, wie Baustellen etc., Reaktionen auf kurzfristig auftretende, dynamische Verkehrslagen sind oft nicht möglich.

## Warnung vor Gefahren

Bei gefahrbringenden Ereignissen wie Gebäudebrand, Gasgeruch oder Bombenalarm, müssen viele Menschen, in einem bestimmten Gebiet, schnell informiert werden. Andererseits sollen Unbetroffene nicht unnötig in Panik versetzt werden. Die Adresse einer warnenden Nachricht kann das betroffene Gebäude, Gebiet oder ein lokal begrenzter Umkreis um den Gefahrenpunkt sein (siehe Abb.3).

Ein weiteres anschauliches Beispiel ist eine Anwendung die dem herkömmlichen Warndreieck entspricht. Ein elektronisches Warndreieck, informiert alle Empfänger, die sich in der Nähe des Gefahrenpunkts befinden oder die sich auf diesen Punkt zu bewegen. Die

### Abbildung 2: Adressierung zur Steuerung des Verkehrs

Empfänger könnten beispielsweise, abhängig von ihrer Entfernung und Position, einen Vorschlag erhalten, wie sie den Gefahrenpunkt umgehen können. (z.B.: "Autofahrer, die sich dem Unfall auf der A8 aus Richtung München nähern benutzen bitte die Umleitung über die Ausfahrt XY") An Personen, die sich sehr dicht an diesem Punkt befinden könnten Hinweise zu Verhaltensmaßnahmen gesendet werden. (z.B.: "Im Falle eines Brandes: Bei starker Rauchentwicklung bitte flach auf dem Boden bewegen")

### Abbildung 3: Warnungen werden an die unmittelbare Umgebung des Gefahrenpunktes gesendet

Bisher wurde eine derartige Warnung meistens durch eine Alarmsirene gegeben, die verschiedene Tonfolgen produziert. Der Nachteil dieser Methode ist, daß Menschen die unterschiedlichen Signaltöne kennen müssen, sonst wird die Bedeutung des Alarmsignals nicht verstanden.

## Hilferuf

Im Fall eines medizinischen Notfalls wird eine Nachricht an den nächstgelegenen Arzt, an das naheste Krankenhaus, oder an Personen in einem bestimmten Umkreis gesendet, die erste Hilfe leisten könnten. (siehe Abbildung 4)

### Abbildung 4: Beispiel: Medizinischer Hilferuf

Bei anderen Notfällen wie z.B. einem Überfall, kann eine Nachricht an umliegende Polizeistationen gesendet werden oder an Personen im Fluchtgebiet des Täters, die später als Zeugen zur Identifizierung beitragen können.

Neben dem einfachen Rufen, Pfeiffen oder einem Taschensignalgeber gibt es bereits für medizinisch gefährdete Personen Systeme, die auf Knopfdruck über Funk einen Arzt informieren. Es ist jedoch gut möglich, daß sich in direkter Nähe eine genauso geeignete Person aufhält. Während ein weiter entfernter Helfer erst zum Notfallort fahren muß, erfährt die örtlich nähere Person nichts. Da viele Notfälle zeitkritisch im Sekundenbereich sind, sind diese Systeme durch GeoCast verbesserungsfähig.

## Wetterservice

Kunden erhalten, abhängig von ihrer geografischen Position, eine Prognose, wie sich das Wetter an dieser Stelle in den nächsten Stunden entwickelt. Wetterstationen, die über Wissen von Position, Bewegungsrichtung und Geschwindigkeit von Regengebieten verfügen, könnten Kunden mit sehr genauen, kurzfristigen Prognosen über die Entwicklung der Wetterlage (oder Änderungen dieser) an einem bestimmten geografischen Ort, versorgen. (siehe Abbildung 5)

Abbildung 5: Eine mögliche Oberfläche eines Wetterservices für Handydisplays

### **Anforderung von ortsbedingten Informationen**

Ortsunkundige Personen, die gerne Essen gehen würden, aber kein Wissen über Restaurants der nahen Umgebung haben, könnten z.B. eine Nachricht an ein bestimmtes Stadtviertel, eine Strasse oder an einen Umkreis, mit bestimmten Radius um sie herum senden. Restaurants, deren Eigenschaften mit den in der Nachricht angegebenen übereinstimmen, könnten auf diese Nachricht (automatisiert) antworten. Der Absender erhält z.B. Informationen über Ort, Öffnungszeiten, Speiseangebot, Anzahl freier Tische oder Spezialitäten. An der Stelle von Restaurants sind auch jede andere Form von Dienstleistungs- und Unterhaltungseinrichtungen möglich.

Im Vergleich zu Virtual Information Towers (VIT), erhält der Benutzer hier die Informationen erst nach Anfrage. Während bei VIT die Informationen einen Radius haben innerhalb dem sie gesehen werden können, ist es in diesem Szenario eher so, daß der Benutzer seinen Sichtradius beeinflussen kann, um dann Informationen an geografischen Positionen zu bekommen, wenn er (aktiv) dannach Ausschau hält. Dieser Aspekt wird manchmal schon von bestehenden Systemen berücksichtigt und mittels Filter- und Suchfunktionen ermöglicht (siehe ViLiS<sup>1</sup>).

Abbildung 6: Vergleich des VIT-Konzepts mit GeoCast

---

<sup>1</sup>Studienprojekt ViLiS WS 99/2000 an der Universität Stuttgart, Institut IPVR, Abteilung VS

## Zeugenbefragungen

Eine nützliche Möglichkeit besteht darin, Personen zu erreichen, die sich zu einem bestimmten Zeitpunkt in der Nähe eines bestimmten Ereignisses aufgehalten haben. Dieses Ereignis kann z.B. ein Verbrechen (siehe oben "Hilferuf") oder schlicht der Verlust eines Gegenstandes sein. Interessant wäre es, wenn dieses Ereignis in der Vergangenheit liegt, die Personen zu erreichen, die sich zum damaligen Zeitpunkt dort aufhielten.

Zum aktuellen Zeitpunkt wird ein Aufruf zu einer derartigen Befragung durch Zeitungsartikel, Radio- / Fernsehmeldungen, "Steckbriefen" oder einer Person, die viele Leute am Ort des Geschehens befragt, umgesetzt.

## Fun-Services

Verschiedene Dienste, die in den Bereich "fun services" fallen sind möglich. Zum Beispiel: "Who is around"-Anfrage. Ein Benutzer schickt eine Mail an seine Umgebung. Andere Personen empfangen die mail. Falls sich der Sender in einer Liste von Bekannten befindet, wird automatisiert eine Antwort gesendet.

Zweites Beispiel: flirt-matching. Ähnlich einem existierenden Dienst, der mit kleinen Infrarot-Sendern arbeitet, kann der Benutzer persönliche Eigenschaften von sich und seinem/seiner Traumpartner/in angeben. Diese Informationen werden periodisch in die nahe Umgebung gesendet. Befindet sich eine Person, die den selben Service benutzt und deren Eigenschaften auf das Partnerprofil passen, in diesem Bereich, werden beide durch eine Nachricht informiert und können in Kontakt treten. Ähnliche Dienste sind auf der Basis von SMS-Nachrichten auf Mobiltelefonen im Entstehen.

## 3 Grundlegende Technologien

In diesem Kapitel werden bestehende Technologien, die für GeoCast relevant sind, kurz vorgestellt. Damit sind sowohl allgemeine Konzepte wie Routingverfahren als auch speziellere Techniken, die im Zusammenhang mit bereits existierenden GeoCast Systemen entwickelt wurden, gemeint.

### 3.1 Broadcast, Unicast, Multicast, Geocast

**Broadcast** bezeichnet das Senden von Nachrichten an alle Empfänger in einem (Sub-)Netz. Eine Broadcast-Sendung wird auch kurz 1:m - Kommunikation genannt. Ein Sender sendet eine Nachricht ohne die genaue Anzahl der Empfänger zu kennen. Er kann davon ausgehen, daß alle potentiellen Empfänger die Nachricht erhalten.

Die Bits der Zieladresse von Broadcast-Messages haben alle den Wert 1. Damit ist die Adresse 255.255.255.255 als lokale Broadcastadresse reserviert.

**Unicast** bezeichnet das Senden an genau eine Adresse. Diese Form der Kommunikation findet zwischen zwei Kommunikationspartnern statt. Sender und Empfänger werden jeweils durch eine IP-Adresse bezeichnet. Man spricht in diesem Fall auch von einer 1:1 (eins zu eins) Kommunikation.

**Multicast** bezeichnet das Senden an eine Gruppe von Empfängern. Die Empfänger müssen sich vorher an eine bestimmte Multicast-Adresse und einen Port angemeldet haben. Die Adresse ist dabei eine bestimmte Form der IP-Adresse bei der das high-order bit auf 1 gesetzt ist. Diese Kommunikationsform wird auch als n:m - Kommunikation bezeichnet, da mehrere Sender an mehrere Empfänger senden können.

**Geocast** bezeichnet das Senden von Nachrichten an ein geografisches Zielgebiet. Einzig und allein dieses Zielgebiet beschreibt die Empfänger. Dem Sender ist im Regelfall unbekannt wieviele Empfänger die Nachricht bekommen werden (0 bis extrem viele). Ein mögliche Kurzform ist 1:GeoRegion. Es können zwar mehrere Sender Nachrichten an die gleiche Region schicken, im Normalfall erwartet ein Sender aber keine Antwort, oder aber eine Antwort, die direkt an ihn (seine spezielle IP-Adresse) gesendet wird. Sieht man sämtliche Kommunikationsteilnehmer, die sich in einem geografischen Gebiet aufhalten und dieses als Zielgebiet verwenden, als Gruppe, kann auch die Bezeichnung n:GeoRegion (oder n:m) benutzt werden. Diese Kommunikationsform ist in der Realität zu vergleichen mit der Vorstellung einer Gruppe von Personen, die sich mit (einstellbaren) Megafonen unterhält. Da damit vielleicht Empfänger erreicht werden, die Nachrichten nicht mitbekommen sollen oder wollen, ist es sinnvoll, weitere Kriterien für die Adressierung zuzulassen und sich Gedanken über Sendeberechtigungen zu machen.

### 3.2 Geografische Routingverfahren

In den letzten Jahren wurden bereits einige Vorschläge gemacht, geografische Informationen dazu zu benutzen, Nachrichten an adressierbare Gebiete auf der Erde zu senden. Einige

Verfahren wurden prototypisch implementiert. In diesem Abschnitt werden die wichtigsten Ansatzrichtungen zu geografischen Routingverfahren aufgezählt.

**Das Cartesische Routing** [Finn87] ist eines der ersten Verfahren, das Pakete aufgrund ihres geografischen Zielgebietes verteilt. Dabei handelt es sich um ein Punkt-zu-Punkt routen das auf der geografischen Position des Senders, der dazwischenliegenden Router und des Zielgebiets basiert. Die Adresse einer Nachricht besteht aus der Angabe einer Position bestehend aus Längen- und Breitengrad und eines weltweit eindeutigen Bezeichners. Um die Nachricht weiterzuleiten, kennt jeder kartesische Router die geografische Position aller direkt benachbarten Router. Die Nachricht wird an den Router weitergegeben, der am nächsten an der Zieladresse liegt. Es existieren Algorithmen für den Fall, daß keine Router in der gleichen Richtung wie das Ziel liegen, die aber hier nicht näher erläutert werden. Siehe dazu [Finn87].

**Die Erweiterung des IP Protokolls** [IPv6RFC1883] schlägt eine Reservierung eines Adressbereichs für geografische Adressen vor. Die IP Adressen werden dabei auf ähnliche Weise gebildet, wie Telefonnummern von der Firma Bell Telephone System erzeugt werden.

**GeoCast** [Navas97], der Ansatz von Navas und Imielinski, versucht ein allgemeineres Konzept zu unterstützen, bei dem Nachrichten an alle Empfänger in einem geografischen Gebiet gesendet werden können, auch wenn die Hosts keine geografische Information besitzen. Dieses System besteht aus drei Komponenten: GeoHosts, GeoNodes und GeoRouters. Ein Anwendung, die eine Nachricht mit einer geografischen Nachricht senden will, kontaktiert einen lokalen GeoHost und erhält die Adresse einer GeoNode. Die Nachricht wird dann an die GeoNode geschickt, welche die Nachricht an einen GeoRouter weiterleitet. Die GeoRouter sind hierarchisch gegliedert und haben die Aufgabe die Nachricht vom Sender ins Zielgebiet zu transportieren. Dort wird die Nachricht an die betreffenden GeoNodes geschickt. Die GeoNodes speichern die Nachricht lokal, weisen ihr einen Multicast-Kanal zu und tragen sie in die Liste der verfügbaren Nachrichten ein. Die GeoNode sendet periodisch die Liste der verfügbaren Nachrichten an eine allen bekannte Multicast Adresse. Ausserdem wird die Nachricht periodisch an den zugewiesenen Multicast-Kanal gesendet. Die GeoHost empfangen die Liste der verfügbaren Nachrichten und finden heraus, ob sie im Zielgebiet liegen. Wenn ein Client eine der Nachrichten empfangen will, tritt er in den Multicastkanal ein und erhält über ihn die Nachricht.

### 3.3 Bestimmung der Position

Für viele Anwendungen ist es wichtig, die genaue Position von Objekten auf der Erde zu bestimmen. Dabei kann es sich sowohl um fest stehende Gebäude, Straßen und Grenzen etc. handeln als auch um mobile Objekte wie Flugzeuge, Schiffe, Fahrzeuge oder Personen mit mobilen Rechnern, Notebooks, PDAs oder Mobiltelefonen mit Internetanbindung. Eine möglichst genaue Bestimmung der Position ausserhalb wie innerhalb von Gebäuden ist

dabei wertvoll. Ein weltweit verfügbares System ist für die Positionsbestimmung ausserhalb von Gebäuden, das Global Positioning System ([GPS]) .

### **Global Positioning System**

Das GPS wurde für das amerikanische Militär entwickelt und wird heute noch vom U. S. Department of Defense betrieben. Zivile Anwendungen dürfen die GPS - Infrastruktur benutzen, jedoch mit kleinen Einschränkungen in der Genauigkeit der Positionsangabe. Das System besteht hauptsächlich aus 24 Satelliten und Empfängern auf der Erde. Die Satelliten umkreisen die Erde auf bestimmten orbitalen Bahnen und senden periodisch präzise Zeitsignale mit Positionsangaben. Die Empfänger auf der Erde können die Dauer der Laufzeit messen, die die Signale vom Satellit zum Empfänger benötigt haben. Mit Hilfe dieser Laufzeit und der Geschwindigkeit elektromagnetischer Wellen, kann die Entfernung zwischen Satellit und Empfänger berechnet werden. Befindet sich der Empfänger in Reichweite von mindestens 4 Satelliten, kann die X,Y und Z Position relativ zu den bekannten Position der GPS-Satelliten bestimmt werden. Ausserdem werden die Zeitsignale dazu benutzt die aktuelle Zeit T zu bestimmen.

Abbildung 7: Berechnung der Position und Zeit mit 4 Satellitensignalen

Die Konstellation der Satellitenbahnen gewährleistet, daß immer 5 bis 8 Satelliten von jedem Punkt der Erde sichtbar sind. Geländeunebenheiten oder Hindernisse wie Bäume und Gebäude können sogenannte Funkschatten verursachen. Dies hat zur Folge, daß nicht immer 4 Satelliten empfangen werden können. Insbesondere innerhalb von Gebäuden ist der Empfang stark eingeschränkt.

## Positionierung innerhalb von Gebäuden

**Das Active Badge System** [ActiveBadge] wurde von der Firma Olivetti entwickelt und ermöglicht die Positionierung von mobilen Objekten innerhalb von Gebäuden. Bei diesem System sind Räume mit Infrarotempfängern ausgestattet. Personen tragen sog. Badges (engl.: Leuchtfeuer), Infrarotemitter, die regelmässig eine eindeutige Kennnummer aussenden. Die Signale werden von den stationären Infrarotsensoren empfangen und über ein Netzwerk an das System gemeldet. Somit kann das System Bewegungen von Personen, die ein Badge tragen, verfolgen. Diese Positionierungsart ist zellbasiert, d.h. das System kann nur aussagen, ob sich ein Badge innerhalb der Reichweite (Zelle) eines Sensors befindet. Da Räume mit Empfängern und mobile Objekte mit Infrarotsendern ausgestattet werden müssen, wird das System für ein großes Gebiet relativ teuer und eignet sich nicht sehr für ein weltweit funktionierendes System.

**Imaging Sensors** ist ein System, das Bilder von Digitalkameras auswertet, die in mobile Rechner eingebaut sind. Es wird versucht aus den Bildern der Digitalkameras Objekte zu extrahieren, deren Positionen dem System bekannt sind. Für diese Aufgabe sind verschiedene (aufwendige) Schritte zur Bildverarbeitung und Objekterkennung abzuarbeiten. Ein Vorteil dieses Konzeptes ist, daß aufgrund der Objekte, die die Kamera sieht, neben der Position auch die Blickrichtung ermittelt werden kann.

**Map Matching** ist ein Verfahren, das angewendet werden kann, wenn andere Messinstrumente wie Digitalkompass und Pedometer zu Verfügung stehen. Wenn bekannt ist, wie weit und in welche Richtung ein mobiles Objekt sich bewegt hat, kann aus diesen Informationen die neue Position aus der vorhergehenden rekonstruiert werden. Map Matching kann auch im Freien verwendet werden. Für Fahrzeuge [Map] ist dieses System gut geeignet, da der zurückgelegte Weg sehr genau gemessen werden kann (Kilometerzähler). Selbst allein aus dem Bewegungsmuster heraus, ist es möglich die aktuelle Position des mobilen Objektes zu berechnen, wenn die möglichen Wege bekannt sind. Das Bewegungsmuster wird dann mit dem Wegemuster verglichen. Wird eine Übereinstimmung gefunden, ist damit der zurückgelegte Weg und die aktuelle Position bestimmt. Bei Fußgängern existiert das Problem, daß die Schrittlängen variieren und die Methode nicht leicht an die einzelnen Körpereigenschaften angepasst werden kann. Neben der ungenauen Messung, können Fußgänger Wege benutzen, die so nicht vorgesehen und in Karten verzeichnet sind.

**Radio Frequency Navigation** funktioniert nach einem ähnlichen Prinzip wie das Active Badge System. Statt Infrarotsignale werden hier Radiofunkwellen verwendet um die Position eines mobilen Gerätes zu bestimmen. Die Intensität der Verbindung zwischen Sender und Empfänger ermöglicht die Berechnung der Distanz zwischen beiden. Wie bei GPS oder im Mobiltelefonnetz kann damit, mittels Triangulation, die relativ genaue Position berechnet werden. Dieses System bietet gleichzeitig die Anbindung in ein Funkdatennetz und ist damit für viele Anwendungen geeignet.

### 3.4 Koordinatensysteme

Ein nicht triviales Problem ist Punkte auf der Erdoberfläche global einheitlich und eindeutig anzugeben. Da die Erde keine exakte Kugel ist sondern einen ellipsoiden Charakter aufweist, ist es schwierig ein einfaches geometrisches Koordinatensystem anzuwenden. Seit jeher mussten Kartographen mit dem Problem fertigwerden, die auf der unebenen Erdoberfläche gerade erscheinenden Linien auf einer ebenen zweidimensionalen Karte wiederzugeben. Im Lauf der Geschichte der Kartographie wurden eine Vielzahl von Kartenprojektionen entwickelt um die dreidimensionale Oberfläche der Erde möglichst verzerrungsfrei darzustellen.

In diesem Abschnitt soll ein kurzer Überblick über drei existierende Koordinatensysteme gegeben werden, die von aktuellen Systemen am häufigsten unterstützt werden.

#### UTM Koordinaten

Das Universale Transversale Mecatorsystem (UTM) wurde in den 40er Jahren von US-Streitkräften entwickelt. In diesem System ist die Erdoberfläche in Zonen aufgeteilt von 8 Grad Höhe und 12 Grad Breit. Es gibt spezielle Zonen in der Nähe der Pole (z.B. über 72 Grad Nord) und einige Ausnahmezonen im Nordatlantik und bei Norwegen. Jede Zone hat ein Unterkoordinatensystem, bestehend aus einem Zentralmeridian und dem Ursprung im Schnittpunkt von Zentralmeridian und Äquator. Positionen werden durch ihre Abstände zum Zentralmeridian und zum Äquator in Meter angegeben, wobei die Werte durch Addition von Korrekturwerten so korrigiert werden, daß keine negativen Abstände auftreten.

#### Gauß-Krüger-Koordinaten

Das Gauß-Krüger-System hat Ähnlichkeiten mit UTM und wird hauptsächlich im europäischen Raum eingesetzt. Es basiert auf einer transversalen Mercatorprojektion mit geodätischen Koordinaten. Im Gegensatz zu UTM gibt es keine rechteckigen Zonen. Im Gauß-Krüger-Koordinatensystem ist die Erdoberfläche in Meridianstreifen eingeteilt, die ,um die Hauptmeridiane im Abstand von 3 Grad, eine Ausdehnung von 1.5 Grad haben. (siehe Abb. Seite 16)

Zur Bestimmung der Position eines Punktes werden Rechtswerte und Hochwerte angegeben. Der Rechtswert einer geografischen Position ergibt sich aus dem Abstand des nächsten Hauptmeridians und dessen Rechtswert. Dabei entspricht der Meridian selbst einem Wert von 500 km, damit für Positionen links (westlich) des zentralen Meridians keine negativen Werte entstehen. Der Hochwert ergibt sich aus dem Abstand zum Äquator. Alle Abstände werden in Metern bzw. Kilometern angegeben. Die Höhe wird über Normal Null (NN) angegeben.

Abbildung 8: Meridianstreifen des Gauß-Krüger-Systems

### Geografische Koordinaten

Geografische Positionen können durch ihre geografische Länge und Breite (engl.: longitude, latitude) beschreiben werden. Die Breite bezieht sich dabei auf den Äquator und reicht von 90 Grad Nord (N) bis 90 Grad Süd (S). 1 Breitengrad entspricht dabei 111,3 km. Die Länge bezieht sich auf den Nullmeridian der durch Greenwich (England) verläuft. Längengrade liegen im Bereich von 180 Grad West (W) bis 180 Grad Ost (E). Das Gebäude der Fakultät Informatik der Universität Stuttgart (Breitwiesenstr. 20-22, Vaihingen) liegt dabei ungefähr bei Lat: 48.722373 Lon: 9.127597.

Weitere Informationen über Koordinatensysteme sind im Kapitel "Geodetic Datums" in [GPS] zu finden.

### 3.5 Architekturansätze

Es gibt verschiedene Architekturansätze für die Realisierung von GeoCast. Einige Architekturklassen sind in den oben beschriebenen bisherigen Ansätzen verwendet worden. Trotzdem die Architektur des im Rahmen dieser Studienarbeit zu entwickelnden Systems durch die Einbettung in Nexus als komponentenbasiertes System vorgegeben ist, sollen andere Ansätze hier kurz erwähnt werden.

#### Stand-alone Application

Eine Architekturform besteht in der Möglichkeit eine einzelne Anwendung zu implementieren. Dieser Anwendung müssen sämtliche Daten über Netze und deren geografischer Positionen verfügbar sein. D.h. eine sehr große Datenmenge muss aktuell gehalten werden. Mangelnde Flexibilität einer Einzelanwendung ist ein weiteres Problem. Die Installation ist relativ einfach, doch insgesamt überwiegen die Nachteile dieses Ansatzes die Vorteile.

## **Verteiltes System**

Funktionalitäten sind auf verschiedene Serverprozesse verteilt. Die einzelnen Komponenten können sinnvoll strukturiert werden. GeoClients kommunizieren (z.B. per RPC oder Corba) mit Serverprozessen. Lastverteilung, erhöhte Zuverlässigkeit, Verfügbarkeit und Skalierbarkeit können eingeplant werden und zählen damit zu den Vorteilen dieses Konzeptes.

## **Neue Kommunikationsprotokolle**

Die tiefgreifendste und langfristig sauberste Methode ist die Erweiterung bestehender Kommunikationsprotokolle um die Fähigkeit Nachrichten, basierend auf geografischen Informationen, zu Adressieren und zu Routen. Dieser Ansatz wurde von Navas und Imielinski [Navas97] vorgeschlagen. Da Änderungen in Protokollen der Netzwerkschicht vorgenommen werden muss abgewartet werden, wie die IETF über die gemachten RFCs (RFC 2009) entscheiden. Aufgrund des Eingriffs in Internetprotokolle, Routing-Software und Domain Name Service ist die Umstellung mit erheblichem Aufwand verbunden. Trotz allem sollte bedacht werden, daß sämtliche anderen Lösungen eines Tages von diesem, oder einem ähnlichen tiefgreifenden Ansatz, überholt werden könnten.

## **Entscheidung für die Einbettung in das NEXUS-Projekt**

Da es abzusehen ist, daß alle Studien- bzw. Diplomarbeiten über dieses Themengebiet im Rahmen des Nexus-Projekts stattfinden, muss die Nexus-Architektur berücksichtigt werden. In Nexus stehen Dienste zu Verfügung, die über definierte Schnittstellen, von Anwendungen benutzt werden können. Damit ergibt sich die Architektur eines GeoCast-Systems als eine in Nexus eingebettete Komponente.

## 4 Adressierungskonzepte

### 4.1 Geometrische Objekte als Adressgebiet

#### Angabe eines geografischen Punktes

Eine Möglichkeit einen Ort zu adressieren, ist die Angabe eines Punktes auf der Erdoberfläche, der durch Länge und Breite definiert ist. Diese beiden Koordinaten spezifizieren einen Punkt im zweidimensionalen Raum. Intuitiv wird angenommen, dass dieser Punkt auf der Erdoberfläche liegt, also einen bestimmten Abstand zu Normal Null (NN) hat. Zur Länge und Breite eines Punktes kann als dritte Koordinate die Höhe über NN dazugenommen werden. Alternativ könnte statt der Höhe über NN der Abstand zum Mittelpunkt der Erde verwendet werden. Eigenschaften der Erdkugel machen diese Größe jedoch ziemlich unbrauchbar. Die Erde beschreibt keine regelmäßige Kugel sondern ist vielmehr ein krummer Ellipsoid. Da Abstandsmessungen relativ zur Meereshöhe (NN) gebräuchlich sind (z.B. bei Höhenangaben), ist dieser Wert am geeignetsten um einen dreidimensionalen Punkt in der Nähe der Erdoberfläche zu beschreiben.

#### Kreis, Kugel, Zylinder

Ein Kreis kann durch einen Punkt (Länge und Breite) und einen Radius beschrieben werden. Zu beachten ist, daß es einen Unterschied macht, ob der Radius in Grad oder Metern angegeben wird.

Kreise sind sehr gut geeignet, um z.B. eine Nachricht an alle Empfänger im Umkreis um ein bestimmtes Objekt (der eigenen Position, eines Gebäudes etc.) herum zu senden. Soll eine Nachricht an Empfänger geschickt werden, die einen absoluten Abstand zu einem bestimmten Punkt haben, kann eine Kugel mit einem dreidimensionalen Punkt und Radius angegeben werden. Ein Zylinder hat einen Kreis als Grundfläche und eine Höhe.

Abbildung 9: Umkreis (geografisches Objekt, Radius) : geografisches Objekt

Beispiel: Adressierung aller GeoClients in Umkreis von 1 km um ein Gebäude (F)

#### Primitive geometrische Figuren (Dreieck, Viereck)

Die Betrachtung der zu adressierenden Gebiete zeigt, daß eine große Anzahl davon durch ein Rechteck beschreiben werden kann. Zu diesen Gebieten gehören: Gebäude Zimmer, Kraftfahrzeuge, Schienenfahrzeuge, gerade oder kurze Straßen, rechteckige Plätze. Für Dreiecke

gibt es so gut wie keine Verwendung. In der Praxis bilden viele Gebiete jedoch keine Rechtecke, sondern unregelmäßige Vierecke. Zudem sind in vielen Fällen weitere Punkte nötig um die Umrisse von adressierbaren Gebieten zu beschreiben. Regelmäßige (symmetrische) Vielecke sind dabei trivialerweise unbrauchbar. Unregelmäßige Vielecke sind im Prinzip geschlossene Polygone.

## **Polygone**

Praktisch alle geometrischen Primitive wie Dreiecke, Vierecke und Vielecke lassen sich durch geschlossene Polygone ersetzen. Ein Polygon ist dabei sehr flexibel. z.B. bleibt ein Polygon ein Polygon auch wenn weitere Punkte hinzugefügt oder entfernt werden. Zur Beschreibung von nicht rechteckigen Gebäuden, Räumen, Stassenzügen, Plätzen usw. sind Polygone bestens geeignet. Um die genaue Position eines Polygons in 3D zu definieren kann einem Polygon eine Höhe über NN zugewiesen werden. Diese Höhe beschreibt die Ebene in der dieses Polygon liegt. Zur genaueren Adressierung von dreidimensionalen Räumen kann dem Polygon eine Höhe zugewiesen werden. Es repräsentiert damit den Grundriss eines dreidimensionalen Raumes dessen Bodenebene sich in einem Abstand relativ zu NN und dessen Decke sich in einem Abstand zur Bodenebene befindet (siehe Abbildung refpolygo-nabb).

Abbildung 10: Polygon mit Höhe und Abstand zu NN

## **Schiefliegende oder sehr komplexe Räume**

Bisher wurde angenommen, daß alle adressierbaren Gebiete Grundflächen haben, die parallel zu NN liegen. D.h. Boden und Decken von Räumen werden als horizontal angenommen. Dies muß aber nicht immer der Fall sein. (z.B. bei eine Kirche mit schiefem Dach) Der Aufwand zur Modellierung komplexerer Räume mit schief liegenden Ebenen erfordert einen erheblichen Aufwand. Die Realisierung könnte so aussehen, daß dreidimensionale Gebilde durch Graphen definiert werden. Jeder Knotenpunkt eines solchen Graphen hätte dann 3 Koordinaten (Länge, Breite und Höhe) und mindestens zwei Kanten zu anderen Knoten. Die Schnittberechnungen mit anderen Objekten würde mit dieser Methode allerdings sehr zeitaufwendig und kompliziert werden.

## 4.2 Logische Bezeichner als Adresse

Der Benutzer soll sich nicht unbedingt um geografische Koordinaten kümmern müssen. D.h. es soll möglich sein, ein logisches Objekt (z.B. Fakultätsgebäude, Schlossplatz, etc) anzugeben, an das die Nachricht gesendet wird.

Beispiel:

- Informatik.Fakultät.Universität.Stuttgart.BW.ger
- Schlossplatz.Stuttgart.BW.ger

Spezielle logische Objekte können mobil sein. Um z.B. alle GeoClients in einem Zug oder Bus zu erreichen muss damit gerechnet werden, daß sich das geografische Gebiet möglicherweise zwischen Absenden und Empfangen der Nachricht weiterbewegt hat. Auch bei mobilen Objekten sollte eine logische Adressierung möglich sein.

Beispiel:

- S\_456.Linie80.Bus.VVS.Stuttgart.BW.ger
- S1.VVS.Stuttgart.BW.ger

Die Schwierigkeit besteht hier, ein bestimmtes ortsunabhängiges Objekt auszuwählen. Es reicht nicht, nur die eine Busroute oder S-Bahn-Linie anzugeben, wenn ein bestimmtes Fahrzeug adressiert werden soll. In diesem Fall müssen weitere eindeutige Attribute wie Autonummer oder Zugname eingesetzt werden, die eine Bestimmung der Position durch einen Lokationservice für mobile Objekte zulassen. Eine andere Möglichkeit ist, aufgrund der aktuellen Position und Geschwindigkeit des mobilen Objekts, eine Fläche zu berechnen, in der es sich nach einer gewissen Verzögerung, aufhalten wird. Eine Nachricht die an diese Fläche gesendet wird, erhält den Hinweis (Attribut), daß Empfänger sich innerhalb des Objekts aufhalten.

## 4.3 Context Awareness-Kriterien

Um eine genauere Auswahl unter den Empfängern in einem Zielgebiet treffen zu können, ist die Hinzunahme von weiteren Eigenschaften oder lokationsabhängigen Beziehungen denkbar. Der Abstand zu einem bestimmten Objekt wäre so eine Beziehung und kann mit der beschriebenen Umkreis-Funktion realisiert werden. Es gibt aber noch andere Eigenschaften, die von Interesse sind.

### Bewegungsrichtung

Möglich wäre eine Adressierung von Empfängern in einem Gebiet, die sich in eine bestimmte Richtung bewegen. Beispielsweise sollen nur Empfänger erreicht werden, die sich in einem bestimmten Gebiet in eine bestimmte Richtung bewegen (siehe Abb 4.3). Die Richtung könnte grob durch Himmelsrichtungen angegeben werden (Nord, Nord-Ost, Ost,

Süd-Ost, Süd, Süd-West, West, Nord-West). Alternativ ist eine Angabe der genauen Richtung mit einer Gradzahl (aus der Navigation bekannt) und einem Toleranzwert denkbar. ( $0^\circ$  entspricht Nord,  $90^\circ$  ost,  $180^\circ$  Süd usw.)

Abbildung 11: Adressierung von Objekten die sich in eine Richtung bewegen

Ein spezielles Anwendungsbeispiel ist die Stauwarnung. Wenn Kraftfahrer auf einer Autobahn vor einem Stau gewarnt werden sollen, wäre es sinnvoll nur die Fahrer zu erreichen, die sich auf den Stau zu bewegen. Autofahrer die bereits im Stau stehen oder Fahrzeuge, die sich auf der Gegenfahrbahn befinden, sollen die Stauwarnung nicht erhalten. Beispiel: Sende "Stauwarnung" an alle GeoClients auf der Autobahn A8, die in Richtung Ost fahren (bzw. Richtung  $90^\circ \pm 5^\circ$ ).

### **Bewegung auf ein Objekt zu**

Für den Fall, daß Empfänger erreicht werden sollen, die sich auf ein bestimmtes Objekt zu bewegen, sollte dieses Objekt spezifizierbar sein (siehe Abb. 4.3). Diese Adressierungsmöglichkeit ist wichtig bei Warnungen vor Gefahrenpunkten (brennendes Haus, Unfall), oder für Hinweise auf ein Objekt in Bewegungsrichtung (vgl. oben) ohne Relevanz des Richtungsvektors (geschlossene Brücke, sehenswertes Objekt ...).

Abbildung 12: Adressierung von Objekten die sich auf ein Objekt zu bewegen

### **Gruppen-Kriterien**

Möglicherweise sollen nicht allen potentiellen Empfängern in einem Gebiet Nachrichten gesendet werden. Um eine weitere Selektion zu ermöglichen, können Nachrichten Attribute mitgegeben werden, die bei der Entscheidung über die Annahme eine Rolle spielen. Attribute können z.B. bestimmte Gruppen-IDs, Alter, Berufsgruppe, Interessen etc. sein.

## 4.4 Zeitkomponente

### Lebenszeit einer Nachricht

Mit der Lebenszeit (Time To Live) kann einer Nachricht einer Zeitspanne zugewiesen werden. Dadurch wird bewirkt, dass alle Empfänger, die in den geografisch adressierten Bereich kommen, diese Nachricht erhalten, solange die Lebensdauer noch nicht abgelaufen ist.

### Gültigkeit einer Nachricht

Einer Nachricht kann ein Zeitpunkt zugewiesen werden, ab dem sie gültig ist. Empfänger erhalten diese Nachricht erst ab diesem Zeitpunkt. Als default Angabe kann für die Gültigkeit der Sendezeitpunkt angenommen werden, sofern kein anderer Wert angegeben wurde. Damit kann eine Nachricht sofort gültig sein oder es zu einem bestimmten Zeitpunkt in der Zukunft werden. Wird auch ein Zeitwert der bereits in der Vergangenheit liegt zugelassen, kann dieser so interpretiert werden, dass allen Empfängern, die sich zu diesem Zeitpunkt im adressierten Zielgebiet aufhielten, die Nachricht geschickt wird. Dadurch ergeben sich allerdings technische Probleme. Um diesen Aspekt realisieren zu können, müsste irgendwo protokolliert werden, wer sich wo und wann aufgehalten hat. Offensichtlich ergeben sich hierbei Konflikte mit dem Datenschutz die ebenfalls gelöst werden müssten.

## 4.5 Zusammensetzung geografischer Objekte

Die Möglichkeit geografische Objekte als Eckpunkte in Polygonen einzusetzen, ergibt eine weitere mächtige Adressierungsart. Ein Polygon kann damit durch eine Auflistung mehrerer geografischer Objekte gebildet werden und ist selbst ein geografisches Objekt. Man beachte, dass geografische Objekte durch logische Bezeichner oder durch geometrische Objekte (wie Polygone) spezifiziert sind (siehe Abb. 4.5).

Abbildung 13: Polygon(Hauptbahnhof, Neues\_Schloss, Rathaus, Punkt(L49.545,B9.841), Keplerstr)

Die Realisierung kann so aussehen, daß davon ausgegangen wird, dass jedes geographische Objekt einen Mittelpunkt hat. Dieser Mittelpunkt kann dazu verwendet werden der Eckpunkt eines Polygons zu sein. Die Fläche des so erzeugten Polygons plus der Fläche der geographischen Objekte bilden damit ein neues geographisches Gebiet. Die explizite Hinzunahme der eigentlichen Fläche ist notwendig, da alle Empfänger innerhalb des Objektes adressiert werden sollen. Bei blosser Verwendung der Mittelpunkte könnten Empfänger aus dem Ergebnispolygon herausfallen.

#### **4.6 Verknüpfung geographischer Objekte**

Damit eine Nachricht an mehrere Objekte gesendet werden kann, könnte die logische Verknüpfung von Objekten möglich sein. Damit ist ein weiteres Mittel gegeben, komplizierte Empfangsgebiete relativ einfach, unter Verwendung von Operatoren wie *and*, *or*, und *not*, anzugeben (siehe Abb. 4.6).

Abbildung 14: Verwendung logischer Operatoren

## 5 Bewertung und Auswahl der Konzepte

Da im Rahmen dieser Studienarbeit nicht alle wünschenswerten Möglichkeiten zur Adressierung implementiert werden können, muß eine Auswahl getroffen werden. Die Wahl grundlegender Systeme wie Koordinatensystem, Kommunikationstechnik etc. ist davon beeinflusst, welche Standards verbreitet sind oder von bestehender Nexus-Infrastruktur unterstützt werden.

Kriterien für die Wahl der Adressierungsarten sind:

- Brauchbarkeit im Hinblick auf die im Kapitel 2 angeführten Anwendungsszenarien
- technische Realisierbarkeit

### Wahl des Koordinatensystems

Aus der Vielzahl unterschiedlicher Koordinatensysteme soll ein System ausgewählt werden, das in der Implementierung unterstützt wird. Da GeoCast weltweit funktionieren soll, fallen Systeme, die nur regional verwendet werden, wie z.B. das Gauß-Krüger-System in Deutschland, weg. Die Vorteile der Geografischen Koordinaten (Länge und Breite) sind:

- global einheitlich und eindeutig
- verbreiteter Standard, der z.B. von GPS-Sensoren unterstützt wird.

Komponenten von Nexus arbeiten zum Großteil mit geodätischen Koordinaten. Damit steht die Verwendung geodätischer Koordinatensysteme im WGS84-Format fest.

### Wahl der Adressierung

Im Kapitel refAdressierungskonzepte Adressierungskonzepte wurden die Möglichkeiten beschrieben, die in der realen Welt vorkommenden Orte und Räume geometrisch zu beschreiben. Dazu müssen Punkte, die eine geografische Position darstellen, angegeben werden können.

#### Punkt

In vielen existierenden Systemen wird der Punkt als kleinste verwendbare geografische Adresse benutzt. Genau betrachtet, ist ein Punkt ein eindimensionales Objekt ohne Ausdehnung bzw. Flächeninhalt. Damit kann sich kein Empfänger auf oder in einem Punkt befinden. Die Verwendung eines einzelnen Punktes funktioniert in diesen Systemen nur, weil vorausgesetzt wird, daß es eine kleinste adressierbare Einheit (Fläche) gibt die durch diesen Punkt repräsentiert ist (z.B. 1 qm). Diese Annahme ist aber nicht unbedingt zutreffend. Die Positionsbestimmung wird immer präziser. In naher Zukunft wird es möglich sein centimetergenaue Angaben über eine geografische Position zu machen. Diese Argumentation und die mathematisch genaue Betrachtung der Eigenschaften eines Punktes,

schliessen ihn allein als geeigneten Typ für eine Adresse aus. Der Punkt kann jedoch als Ursprungspunkt einer Nachricht, als der Punkt von dem aus ein Sender eine Nachricht abgeschickt hat, verwendet werden, da dieser Punkt zur Auffindung des Senders (z.B. bei einem Hilferuf) genügt.

### **Kreis, Kugel, Zylinder**

Eine relativ häufig eingesetzte Methode, ist das Senden an alle Empfänger, die sich in einem gewissen Umkreis um den Sender, ein anderes markantes Objekt oder einen Punkt herum befinden. Um dies zu realisieren, müssen kreisförmige Gebiete möglich sein. Spielt die Höhe über NN eine Rolle, so nimmt der Zielraum einen kugelförmigen oder zylinderförmigen Charakter an. Die Betrachtung der Anwendungsszenarien zeigt, daß es keinen Fall gibt, bei dem eine Kugel als Zieladresse notwendig wäre. Ein Zylinder ist in den bekannten Fällen ausreichend. Eine Kugel kann zu Not durch ein zylindrisches Objekt angenähert werden. Kreis und Zylinder sind technisch relativ einfach realisierbar.

### **Dreieck, Viereck**

Primitive geometrische Figuren wie Dreiecke, Vierecke etc. brauchen nicht explizit betrachtet werden, da sie in der Klasse der Polygone enthalten sind bzw. mittels Polygonen abgebildet werden können. Koordinatenachsenparallele Rechtecke, die komplexere Polygone umschliessen, können als grobe Zielgebiete von Routingsystemen verwendet werden, spielen aber als adressierbares Zielgebiet keine Rolle.

### **Polygon**

Polygone können zweidimensionale Gebiete adressieren, da oft die dritte Dimension (Höhe) keine Rolle spielt. Für bestimmte Adressgebiete wie Stockwerke, Zimmer etc. wird jedoch ein Polygon benötigt, dessen Grundflächenebene bekannt ist und das eine Angabe der Höhe des zu adressierenden Raumes beinhaltet. Die Angabe und Verarbeitung von Polygonen ist technisch gut umsetzbar und wird bereits von einigen Teilsystemen bereitgestellt.

### **Schiefliegende Räume**

Probleme bereiten vor allem schiefliegende und besonders komplexe Raumgebilde. Diese Räume kommen in der Praxis nur sehr selten vor oder können durch normal liegende Figuren ausreichend angenähert werden. Deshalb kann in einer Realisierung die Adressierung schiefer Räume vernachlässigt (weggelassen) werden. Ebenfalls vernachlässigbar sind Räume, die an unterschiedlichen Stellen verschiedene Höhen haben. Grundflächen adressierbarer Orte haben damit immer einen Normalenvektor, der senkrecht zur Erdoberfläche steht. Die Berechnung, ob ein Empfänger innerhalb eines Raumes mit schiefen Grenzflächen liegt, ist technisch relativ aufwendig. Damit werden schiefliegende Räume nicht in die Liste der unterstützten Adressgebietstypen aufgenommen.

## **Logische Bezeichner**

Eine große Erleichterung für den Benutzer besteht in der Möglichkeit, Adressgebiete durch logische Bezeichner anzugeben. Wie bei der Adressierung von Webseiten, wo durch ein NamingService einprägsame Namen auf die entsprechenden IP-Adressen und Ports abgebildet werden (z.B. "www.informatik.uni-stuttgart.de" wird abgebildet auf 129.69.211.2) kann auch bei der geografischen Adressierung nicht vom Benutzer verlangt werden, Zielgebiete jedesmal durch Angabe von geometrischen Figuren und geodätischen Koordinaten anzugeben. Die technische Realisierung dieser Anforderung ist machbar, erfordert jedoch die Implementierung eines speziellen Services, der logische Adressen in geografische Adressen umwandeln kann.

## **Mobile adressierbare Objekte**

Die Adressierung mobiler Objekte ist in manchen Anwendungen sinnvoll. Beispielsweise kann es wünschenswert sein, Empfänger in Bussen, Bahnen, Schiffen oder Flugzeugen zu erreichen. Diese Möglichkeit erfordert jedoch einen hohen technischen Aufwand. Es muss ein Verzeichnis existieren, das Bewegungen mobiler Einheiten verfolgt, damit der aktuelle Standort ermittelt werden kann. Ausserdem müssen Geschwindigkeit und Bewegungsrichtung des Objektes zu Verfügung stehen, damit der potentielle Ort errechnet werden kann, an dem es sich zum Zeitpunkt des Eintreffens der Nachricht aufhalten wird, da bei schneller Fortbewegung die geografische Adresse dann veraltet sein kann. Die Adressierung mobiler Objekte soll vorerst durch das zu entwickelnde System noch nicht unterstützt werden. Eine Erweiterung um diese Funktionalität in der Zukunft ist möglich.

## **Context Kriterien**

Für einige Anwendungen, wie z.B. Verkehrssteuerung, sind neben dem Aufenthaltsgebiet auch Kriterien wie Geschwindigkeit und Bewegungsrichtung nützlich. Wie genau Empfänger Informationen über ihre Bewegungsrichtung bereitsteht, hängt von den Sensoren ab, mit denen ihre Kommunikationseinheiten ausgerüstet sind. Solche Kriterien können ohne großen technischen Aufwand den Nachrichten als Header im Adressteil hinzugefügt werden. Die Auswertung kann den Empfangsgeräten überlassen werden.

Für die Einschränkung der Empfänger durch weitere Kriterien (Attribute, Gruppen-IDs) können diese Kriterien ebenfalls an die adressierten Anwendungen weitergegeben werden, welche über die weitere Behandlung der Nachricht entscheiden. Dies muss nicht Angelegenheit des Routingverfahrens sein. Context Kriterien können durchaus wichtig sein und werden deshalb in die Adressierung mit aufgenommen.

## **Zusammensetzung und Logische Verknüpfung von Adressen**

Die Realisierung der logischen Verknüpfung von verschiedenen geografischen Gebieten erscheint sinnvoll. Die Erzeugung eines Adressgebietes, das ein Ergebnis einer logischen Verknüpfung ist kann dem Sender (der Sendersoftware) überlassen werden. Somit muss diese

Möglichkeit nicht speziell als Adressierungskonzept für GeoMessages integriert und implementiert werden.

Zur Notation der Adressgebiete siehe Spezifikation.

## 6 Spezifikation GeoClient

### 6.1 Überblick

Der GeoClient ist ein Prozess der auf einem mobilen oder stationären System läuft, das GeoMessages empfangen und senden können soll. Der GeoClient bildet damit die Schnittstelle zwischen Anwendungen und den GeoCastkomponenten, die für das Adressieren und Routen von GeoMessages zuständig sind. Abbildung 6.1 zeigt eine grafische Darstellung der Abhängigkeiten.

Abbildung 15: Überblick über die GeoCast Komponenten

### 6.2 Aspekt Empfang

Auf einem speziellen (noch festzulegenden) Port empfängt der GeoClient ein Message vom zugrundeliegenden Routing-System. Der GeoClient muss überprüfen, ob seine geografische Position tatsächlich im angegebenen Zielgebiet liegt. Ausserdem überprüft er die Gültigkeit der Nachricht und vergleicht weitere Kriterien wie Bewegungsrichtung bzw. Blickwinkel, Validität und ggf. weitere Gruppenkriterien. Sollte sich die Empfangseinheit ausserhalb des Zielgebiets befinden, die Nachricht ungültig sein oder Auswahlkriterien nicht übereinstimmen, wird die Nachricht verworfen. Im anderen Fall, wird die Nachricht an die, durch einen bestimmten Port identifizierte Anwendung auf diesem System, weitergesendet. Beispiel: GeoClient empfängt GeoMail. Bei Erfüllung der notwendigen Bedingungen wird die mail an Port 20 (mailport) gesendet bzw. dem mail-folder des User angehängt.

### 6.3 Aspekt Senden

Anwendungen können dem GeoClient Mails und Nachrichten übergeben um sie zu versenden. Der GeoClient überprüft die angegebene Zieladresse. Liegt kein Adressformat vor, das vom Routing-System direkt verarbeitet werden kann, werden weitere Services kontaktiert (Location Service, falls mobile Objekte adressiert werden sollen, SMS (Spatial Model Server) für stationäre, fixe Objekte) Sollte aus irgend einem Grund die Auflösung der geografischen Adresse nicht möglich sein, z.B. weil ein logisches Objekt nicht bekannt ist, schickt der GeoClient an die entsprechende Anwendung eine "address resolution failure"-Nachricht (vergleichbar mit den failure-Nachrichten des normalen email-Systems).

### 6.4 Schnittstelle zum Routing-System

Der Begriff für die Nexus-Komponente des Routing-Systems, die der GeoClient kontaktieren soll, ist: GeoNode. Diese Komponente wird von Gerald Grau im Rahmen seiner Diplomarbeit entwickelt und implementiert. Die hier beschriebene Schnittstelle zwischen GeoClient und GeoNode wurde von Gerald Grau und Alexander Till gemeinsam festgelegt.

Allgemeine Bemerkung: Aufgrund der verwendeten Kommunikationsmethode wurde entschieden, Nachrichten im XML-Format einzusetzen. Die Darstellung der XML-Nachricht befindet sich im Anhang A auf Seite 59. Für die genaue Realisierung siehe Sourcecode.

Ebenfalls im Anhang befindet sich eine grafische Darstellung des Nachrichtenflusses zwischen den verschiedenen Komponenten.

Jede Nachricht soll entweder einmal alle Empfänger im Empfangsgebiet erreichen (Message-Type = ONCE), oder die GeoNode sorgt dafür, daß während einer bestimmten Zeitperiode

alle Empfangseinheiten, die in dieses Gebiet kommen, die Nachricht erhalten (MessageType = PERIOD). Die Zeitperiode wird dabei durch einen Start- und einen Endzeitpunkt spezifiziert.

Empfangsgebiete können entweder durch ein Polygon (mehr als 2 geografische Positionen) oder durch einen Kreis angegeben werden. Ausserdem kann eine Höhe über NN und eine Raumhöhe angegeben werden. Diese Information ist für die GeoNode jedoch transparent. Als Adressgebiet wird der GeoNode nur ein, das Gebiet umschließendes Rechteck (BoundingBox), oder ein, das Gebiet umschließender Umkreis (BoundingCircle) übergeben. BoundingBox und BoundingCircle sind jeweils durch zwei geografische Positionen beschrieben.

Bei einer BoundingBox wird die erste GeoPosition als obere linke, die zweite GeoPosition als untere rechte Ecke interpretiert.

Bei einem BoundingCircle wird die erste GeoPosition als Kreismittelpunkt, die zweite GeoPosition als Punkt auf dem Kreis interpretiert.

Besitzen beide Positionen Höhen, so wird die tiefer liegende Höhe als Höhe der Grundfläche über NN, die darüber liegende als Höhe der Dachebene angenommen. Damit wird im Fall einer BoundingBox ein dreidimensionaler Raum (Quader), im Fall eines BoundingCircles ein Zylinder definiert.

## 6.5 Komponenten mit denen ein GeoClient kommunizieren muß

Die Phase, die beim GeoClient mit "Address Resolution" angegeben ist, erfordert die Kommunikation mit Komponenten des Nexus-Systems.

Dieser Abschnitt ist abhängig von der Notation der Zieladresse. Insbesondere ist wichtig, ob eine Adresse logisch angegeben werden kann (germany.ba-wue.stuttgart.breitwiesenstr.22) oder ob der GeoClient die Adresse soweit auflösen muss, dass dem darunterliegenden Routing-System nur ein Polygon (oder Kreis, oder Zylinder) übergeben wird. D.h. dass alle logischen Angaben eliminiert sind. Sollte dies der Fall sein, benötigt ein GeoClient Informationen wie:

- geografische Beschreibung jedes möglicherweise logisch adressierbaren Objektes (Länder, Staaten, Städte, Strassen, Gebäude, Zimmer)
- die Höhe der Erdoberfläche über NN zu Objekten auf der Erde.

Diese Informationen müssen dem GeoClient mittels (hierarchisch gegliederten) Servern angeboten werden.

## 6.6 Algorithmen

Der GeoClient berechnet beim Empfang einer Nachricht, ob er tatsächlich im adressierten Zielgebiet ist. Dabei kann ein Zielgebiet angegeben sein als: Kreis, Polygon oder Zylinder. Es wird davon ausgegangen, daß dem GeoClient seine aktuelle Position bekannt ist. Bei mobilen Geräten bekommt er z.B. seine Position über GPS-Sensor. Bei stationären Einheiten muss die feste geografische Position bekannt sein.

### Kreis

Der GeoClient befindet sich innerhalb des Zielgebietes, wenn sein Abstand zum Mittelpunkt des Kreises kleiner oder gleich dem Radius des Kreises ist.

Abstand (CIRCLE.CenterPosition, GeoClient.Position)  $\leq$  CIRCLE.Radius

### Zylinder

Der GeoClient befindet sich innerhalb des Zielgebietes, wenn der Abstand zum Mittelpunkt des Zylinders kleiner oder gleich dem Radius des Zylinders ist und wenn die Höhe des GeoClients größer oder gleich der Höhe der Grundfläche des Zylinders ist und wenn die Höhe des GeoClients kleiner oder gleich der Höhe der Grundfläche des Zylinders plus der Höhe des Zylinders ist.

Abstand (CYLINDER.CenterPosition, GeoClient.Position)  $\leq$  CYLINDER.Radius  
AND CYLINDER.CenterPosition.Height  $\leq$  GeoClient.Position.Height  
AND GeoClient.Height  $\leq$  (CYLINDER.CenterPosition.Heighth + CYLINDER.Height)

### Polygon ohne Höhe

Der GeoClient befindet sich innerhalb des Zielgebietes, wenn seine Position innerhalb des Polygons liegt. Um zu Berechnen, ob sich ein Punkt innerhalb eines Polygons befindet, wird ein Strahl (Gerade) mit einem Ursprung in diesem Punkt, in eine beliebige Richtung gebildet. Die Anzahl der Schnittpunkte dieses Strahls mit den Kanten des Polygons bestimmt, ob sich der Punkt innerhalb oder ausserhalb befindet. Bei ungerade Anzahl der Schnittpunkte liegt der Punkt innerhalb, bei gerader Anzahl (auch Null) ausserhalb. Abbildung 6.6 soll den Algorithmus verdeutlichen.

Abbildung 16: Punkte P2 und P5 liegen innerhalb eines Polygons

**Polygon mit Höhe**

Der GeoClient befindet sich innerhalb des Zielgebietes, wenn seine Position innerhalb des Polygons liegt (siehe **Polygon ohne Höhe**) und seine Höhe größer oder gleich der Höhe der Grundfläche des Polygons ist und wenn die Höhe des GeoClients kleiner oder gleich der Höhe der Grundfläche des Polygons plus der Höhe des Polygons ist.

```
GeoClient.Position innerhalb POLYGON
AND POLYGON.Position.Height <= GeoClient.Position.Height
AND GeoClient.Height <= (POLYGON.Position.Heigth + POLYGON.Height)
```

## 7 Spezifikation Anwendung

### 7.1 Überblick

Die grobe Beschreibung der Anwendung GeoMessenger wie es der Aufgabenstellung der Studienarbeit zu entnehmen ist: Es soll eine Anwendung entwickelt werden, die das Versenden und Empfangen von GeoMail und GeoMessages erlaubt. Unter GeoMail versteht man Nachrichten, die wie email, nach dem Empfang gespeichert und erst auf Wunsch des Nutzers angezeigt werden. GeoMessages hingegen werden sofort und ohne Benutzerinteraktion angezeigt. Die Anwendung soll auf einem mobilen Rechner laufen. Der mobile Rechner kann über einen GPS-Sensor verfügen, der die Anwendung mit Positionsinformation im WGS84-Format versorgt.

Die Anwendung GeoMessenger soll folgende Funktionalität bereitstellen:

- Senden und Empfangen von Mail, insbesondere GeoMail.
- Senden und Empfangen von GeoMessages (werden sofort angezeigt)
- Möglichkeit zur Angabe von Zielgebieten für GeoMail und GeoMessages
  - mittels Angabe eines logischen Bezeichners eines geografischen Gebiets.
  - mittels Angabe des Zielgebiets mit Hilfer einer Kartenkomponente

### 7.2 Benutzungsoberfläche

Das Aussehen der Benutzungsoberfläche kann bei verschiedenen Plattformen unterschiedlich aussehen. Die vorliegende Beschreibung ist eine erste Version.

## Allgemein

Die Bedienung der Anwendung richtet sich nach der Plattform auf welcher sie ausgeführt wird. Für die Ausgabe ist ein Monitor vorgesehen. Die Ausgabe auf einem Handheld-PC-Display wird nicht berücksichtigt. Für die Bedienung ist eine Tastatur und eine Maus vorgesehen. Begriffe wie Tastatur und Maus sind in ihrer erweiterten Bedeutung gemeint. Zum Beispiel kann statt einer Maus auch ein Trackball oder ein Touchpad bei einem Notebook eingesetzt werden, wenn der entsprechende Treiber installiert ist.

## Normalzustand

Im Normalzustand werden die folgenden Informationselemente angezeigt:

### Menüleiste

**File** hat einen Unterpunkt "Exit" mit dem das Programm beendet werden kann.

**Settings** ermöglicht die Anpassungen und Einstellungen der Anwendung. Folgende Unterpunkte befinden sich in diesem Menü:

**Mail** Einstellungen zum Empfang und zur Darstellung von Mails.

**GPS-Sensor** Einstellungen zum GPS-Sensor

**Look&Feel** Wahl des Look and Feel - Modes der Oberfläche.

**Books** ermöglicht das Speichern und Verwalten von Adressen (geografischen Positionen und Gebieten).

**Bookmark Position** ermöglicht das Speichern und Archivieren von Positionen (auf der Karte) an denen sich der Benutzer befand (real oder virtuell).

**Bookmark AddressArea** ermöglicht das Speichern und Archivieren von geografischen Gebieten, für die ein Bezeichner gewählt werden kann und die nur lokal (dem Benutzer) bekannt sind.

**Edit Addressbook** ermöglicht das Ändern und Löschen der Einträge.

## Bedienelemente

### Inbox

Im Inbox-Panel befinden sich zwei Registerblätter. Auf dem ersten Registerblatt (Inbox) ist der Inhalt der mailbox zu sehen. Zeilenweise werden hier die emails in von anderen mail-readern gewohnter Weise dargestellt.

In der ersten Spalte (ein Buchstabe) wird durch ein "N" angezeigt, ob eine mail neu, also noch ungelesen ist. Bei bereits gelesenen mails steht hier ein Blänk (" "). Mails die zum Löschen markiert sind, erhalten ein "D". Bei mails, die beantwortet wurden steht ein "A".

In der zweiten Spalte steht die Nummer der mail.

In der dritten Spalte steht das Datum.

In der vierten Spalte steht der Name des Absenders (wie er aus der From-Angabe ausgelesen wurde)

In der fünften Spalte steht die Größe der mail.

In der sechsten Spalte steht das Subject.

**Anzeigen einer GeoMail:** Durch selektieren einer Zeile im Inbox-Fenster und [RETURN] oder durch Doppelklick auf eine Zeile.

**Anzeigen einer GeoMessage:** GeoMessages werden sofort nach ihrem Eintreffen in einem Dialogfenster angezeigt.

**Send** Das zweite Registerblatt (Send) deckt alle Angaben auf, die zum Verschicken von GeoMails oder GeoMessages notwendig sind.

Zuerst muss der Benutzer entscheiden, ob er eine GeoMail oder eine GeoMessage versenden will (Radiobox)

Dann muss ein Zielgebiet für die Nachricht spezifiziert werden. Dem Benutzer stehen dabei mehrere Möglichkeiten zur Auswahl:

- Er kann ein geografisches Objekt aus der Liste der lokal bekannten geografischen Gebiete wählen (über die Bookmarks).
- Er kann eine logische Bezeichnung für ein globales geografisches Objekt angeben (Bsp: "Earth.Germany.Ba-Wue.Stuttgart.Breitwiesenstr.20")
- Er kann ein Polygon oder Kreis angeben, dessen relevante Punkte, mit Hilfe der nebenstehenden Karte durch Klicken auf die Positionen der Ecken bzw. des Kreismittelpunktes, ermittelt werden können.

Default-Werte für Absender und Position des Senders (stationär oder vom GPS-Sensor) wurden bereits gesetzt, können aber manuell noch verändert werden.

Für das Validitätsdatum wird das aktuelle Datum gesetzt (manuelle Manipulation) Default-Wert für die Lebensdauer gibt es nicht, kann manuell gesetzt werden.

## 7.3 Datenstrukturen

### einfache Daten

Es wird davon ausgegangen, daß primitive Datentypen, wie Byte, Integer, Real, Float und String, bekannt sind.

**geografische Position:** Tupel, bestehend aus 3 Koordinaten (Länge, Breite, Höhe)

**geografisches Objekt:** Objekt mit Mittelpunkt (geografischer Position) und Ausdehnung (angegeben durch Polygon oder Kreisradius)

**geografischer Kreis:** Mittelpunkt (geografische Position) und Radius (Kommazahl, Meter)

**mobiles geografisches Objekt:** geografisches Objekt mit dynamischer Position, Bewegungsrichtung und Bewegungsgeschwindigkeit.

**Polygon:** Liste aller Eckpunkte eines Polygons. Anordnung der Eckpunkte erfolgt im Gegenuhrzeigersinn. Ein Eckpunkt ist vom Typ *geografische Position*. Ein Polygon hat mindestens 3 Eckpunkte.

### Daten einer GeoMessage

Jede GeoMessage besitzt:

- From: Absenderadresse (email-Adresse)
- From-pos: Absenderstandort (geografische Position, Authentifizierung)
- Date: Datum (Start der Gültigkeit)
- Expires: Datum (Verfallsdatum)
- To: Zielregion (geografisches Objekt)
- Subject: Überschrift (Zeichenkette)
- Message-ID: (Zeichenkette)
- Den Inhalt der Nachricht (Zeilen von Zeichenketten)

### Daten einer GeoMail

Jede GeoMail besitzt die gleichen Daten wie eine GeoMessage. Im Header der Nachricht wird als Zielapplikation "mail" angegeben.

## 8 Entwurf

### 8.1 Entwurfspakete

Im gesamten GeoCast-Projekt gibt es folgende Pakete:

**Communication** enthält Klassen, die von allen Klassen der anderen Packages benötigt werden. Dies sind Klassen, die zur Kommunikation der Komponenten untereinander benötigt werden.

**GeoApplication** enthält Basisklassen, die Anwendungsentwickler bei der Programmierung von Anwendungen, die in der Lage sind GeoMessages zu senden, unterstützen. Diese Basisklassen implementieren das Kommunikationsprotokoll zwischen einer GeoApplication und dem GeoClient.

**GeoClient** beinhaltet die Implementierung des GeoClients

**GeoLocationServer** enthält alle Klassen, die den LocationService implementieren.

**GeoMessenger** repräsentiert eine Beispielanwendung.

**GeoNode** stellt die Implementierung des Routingsystems bereit. Nachrichten mit einer geografischen Adresse, die von GeoClients an GeoNodes übergeben werden, werden von GeoNodes in das geografische Zielgebiet geroutet und dort existierenden GeoClients übergeben. *Dieses Paket wird von Gerald Grau im Rahmen seiner Diplomarbeit entwickelt und implementiert.*

---

### Communication

Das Paket "Communication" enthält sämtliche Klassen, die Objekte der anderen Pakete benötigen. Dies sind z.B. Klassen, die Funktionen für die Kommunikation zwischen den Objekten bereitstellen. Neben diesen Klassen, befinden sich hier auch Objekte, die zwischen verschiedenen Komponenten ausgetauscht werden (z.B. GeoAddress, GeoMessage). Der Entwurf der wichtigsten Klassen ist in diesem Abschnitt wiedergegeben.

#### Interface DatagramSocketListenerUser

Dieses Interface sollte von allen Klassen implementiert werden, die einen oder mehrere DatagramSocketListener verwenden. Das Interface deklariert nur eine Methode, die von der Subklasse implementiert werden muss.

**void receiveDatagram(java.net.DatagramPacket packet)** wird von einem DatagramSocketListener aufgerufen, wenn er ein Packet auf einem Socket empfängt. Aus dem übergebenen Argument "packet" kann der Inhalt und der Absender des Datagrammpakets entnommen werden.

## Class DatagramSocketListener

Die Klasse DatagramSocketListener beinhaltet Methoden, die es ermöglichen, Nachrichten, die von einem DatagramSocket empfangen werden, an einen DatagramSocketListenerUser weiterzugeben um sie dort zu verarbeiten. Der DatagramSocketListener ist dabei ein eigenständiger Thread, der an einem bestimmtem Port lauscht. Ausser DatagramSockets werden auch MulticastSockets unterstützt. MulticastSocket ist eine Subklasse von DatagramSocket. D.h. ein DatagramSocketListener kann auch einen MulticastSocket initialisieren und empfängt dann Multicast-Nachrichten. Für einen DatagramSocketListenerUser ist es transparent, ob eine Multicast-Nachricht oder ein Datagramm empfangen wurde. Multicastpakete und Datagramme haben in java den gleichen Typ (DatagramPacket).

### **DatagramSocketListener(DatagramSocketListenerUser myOwner, PrintStream log)**

initialisiert den DatagramSocketListener. Der Konstruktormethode muss als erstes Argument eine Referenz auf einen DatagramSocketListenerUser übergeben werden. Es ist wichtig, daß diese Referenz auf ein gültiges Objekt weist, da der DatagramSocketListener die Methode receiveDatagramm des Users aufruft, wenn ein Datagramm empfangen wird. Für den Fall, daß der DatagramSocketListener eine Protokolldatei (Logfile) schreiben soll, kann dem Konstruktor ein PrintStream als zweites Argument übergeben werden.

Diese Klasse implementiert die run-Methode des Runnable-Interfaces. Die run-Methode sollte jedoch nicht aufgerufen werden. Zum Starten einer Objektinstanz soll die start()-Methode aufgerufen werden (siehe auch java-Dokumentation zu Threads).

Achtung: Der Socket ist direkt nach der Erzeugung einer Instanz dieser Klasse noch nicht initialisiert. Er muss mit einer der folgenden drei Methoden erst initialisiert werden, bevor der Thread gestartet wird.

**initDatagramSocket()** initialisiert den DatagramSocket dieses Threads auf einen zufällig ausgewählten freien Port. Der Thread kann nach Aufruf dieser Methode gestartet werden. Falls kein freier Port verfügbar ist, wird eine SocketException geworfen.

**initDatagramSocket(int port)** initialisiert den Socket auf dem angegebenen Port. Sollte dieser Port nicht frei sein, wird eine SocketException geworfen.

**initMulticastSocket(int mcport, String multicastGroup, int defaultTTL)** initialisiert den Socket als MulticastSocket. Der Multicastkanal ist durch den Port (erstes Argument) und das zweite Argument multicastGroup, die Adresse der Multicastgruppe, spezifiziert. Als drittes Argument muss noch ein default Time-To-Live Wert angegeben werden, der bestimmt, wieviele Sprünge (Hops) ein Multicast-Datagramm-Paket machen darf. Damit wird die Zahl der Multicast-Router begrenzt die dieses Paket weiterleiten.

**DatagramSocket getSocket()** ist eine Methode, die den (bereits initialisierten) DatagramSocket zurückliefert. Damit können Methoden des Sockets des DatagramSocket-Listeners ausgeführt werden, z.B. wenn die lokale Adresse oder der Port des Sockets ermittelt werden soll. Vorsicht: Der Socket kann sich im blockierten Status (receive(packet)) befinden. Der Socket darf daher nicht geschlossen oder sonstwie beeinflusst werden, da der DatagramSocketListener sonst gestört wird und es möglich ist, daß ankommende Nachrichten nicht an den DatagramSocketListenerUser weitergegeben werden können.

**DatagramSocket getSocket()** liefert die Nummer des Port zurück auf dem der Socket initialisiert wurde. Diese Methode ist nützlich, wenn der Socket mit initDatagramSocket() initialisiert wurde und man wissen will, welche Nummer der Port hat. Die Methode ruft getLocalPort des bereits initialisierten Sockets auf und gibt das Ergebnis zurück. Ist der Socket noch nicht instanziiert wird der Wert -1 zurückgegeben.

**void stopSocketListener()** beendet den Thread. Da die stop() Methode der Thread Klasse veraltet ist, sollte immer nur diese Methode verwendet werden um einen DatagramSocketListener zu unterbrechen. Diese Methode verursacht, daß die run - Methode des Threads verlassen wird.

---

## Class GeoAddress

Die Klasse GeoAddress kapselt Funktionen zum Einlesen und zur Ausgabe einer logischen geografischen Adresse. Jeder (gültigen) logischen Adresse sollte mittels des GeoLocationService ein geografisches Gebiet, in Form eines Polygons o. ä., zuzuordnen sein.

**GeoAddress(String addressString)** benutzt den übergebenen String zur Initialisierung des Objektes. Im String müssen die Domains durch Punkte voneinander getrennt sein. Beispiel: "Germany.Baden-Wuerttemberg.Stuttgart.Vaihingen.Breitwiesenstr.20-22". Erlaubte Sonderzeichen: \_ , -. Weitere Sonderzeichen (auch Leerzeichen) sind nicht erlaubt.

**static GeoAddress valueOf(String from)** ist eine weitere Methode um ein Objekt von GeoAddress zu erzeugen.

**String toString()** gibt die logische geografische Adresse als String aus. Die Ausgabe entspricht dabei dem geforderten Format für logische Adressen.

---

## Class GeoMessage

Die Klasse GeoMessage repräsentiert ein Nachricht, die aufgrund geografischer Adressinformation verschickt werden kann. GeoMessage stellt dabei Methoden zur Erzeugung, Eingabe und Ausgabe einer Nachricht bereit. Die Form der Nachricht wurde an das Aussehen einer Mime-Message angelehnt. Objekte vom Typ GeoMessage werden von einer Applikation erzeugt und einem GeoClient übergeben.

**GeoMessage()** ist der einzige Konstruktor von GeoMessage und initialisiert interne Variablen mit Defaultwerten.

**void setFrom(InternetAddress fromAddress)** setzt den Wert, der als Absenderadresse benutzt wird. Der Defaultwert wird im Konstruktor gesetzt, wobei versucht wird, aus den Systemvariablen die Emailadresse des Benutzers oder <username>@<localmachine> zu verwenden. fromAddress muss eine gültige Adresse sein.

**InternetAddress getFrom()** liefert die aktuell gesetzte Adresse des Senders zurück.

**void setFromPosition(Coordinate fromPosition)** ermöglicht das Setzen einer geografischen Position von der die Nachricht stammt. Die Position ist vom (abstrakten) Typ nexus.location.Coordinate, wird aber in dieser Methode nach GeodeticCoordinate konvertiert, da die meisten Komponenten nur geodätische Koordinaten unterstützen. Das Setzen der Position des Senders ist wichtig, z.B. um im Notfall die Quelle einer GeoMessage aufzufinden. Der Defaultwert der Senderposition wird im Konstruktor gesetzt, wobei die Position GeodeticCoordinate(48.72247,9.12446,0.0) benutzt wird, welche in etwa der Lage des Gebäudes der Fakultät Informatik (Feb. 2001) entspricht.

**Coordinate getFromPosition()** liefert die aktuell gesetzte Position des Senders. Das Objekt ist ein Subtyp der abstrakten Klasse Coordinate (aus dem Package nexus.location) und hat den Typ GeodeticCoordinate.

**void setSentDate(Date date)** setzt das Datum das als Sendedatum verwendet werden soll. Der GeoClient aktualisiert das Sendedatum bei Auslieferung der Nachricht an eine GeoNode.

**Date getSentDate()** liefert das aktuell gesetzte Sendedatum zurück.

**void setExpirationDate(java.util.Date date)** setzt das Verfallsdatum einer GeoMessage. Solange das Verfallsdatum einer Nachricht nicht überschritten wurde, wird die Nachricht periodisch an alle (möglicherweise mobilen) Empfänger im Zielgebiet ausgeliefert. Die GeoClients sorgen dafür, das Anwender eine Nachricht nur einmal zu Gesicht bekommen indem sie eine mitgesendete Message-ID vergleichen. Wurde kein Verfallsdatum gesetzt, (oder ist es gleich dem Sendedatum) wird die Nachricht einmal an alle Empfänger die sich aktuell im Empfangsgebiet aufhalten, gesendet.

**Date getExpirationDate()** liefert das aktuell gesetzte Verfallsdatum der GeoMessage.

**void setHeader(String headerName, String headerValue)** ermöglicht das Hinzufügen eines zusätzlichen Headers, ähnlich den Headern von Mime-Messages. Damit können einer GeoMessage beispielsweise weitere Kriterien (wie Status, Geschlecht, Beruf, Bewegungsrichtung, Gruppenbezeichnung etc.) mitgegeben werden, die die Empfänger im Zielgebiet adressieren.

**String getHeader(String headerName, String defaultValue)** liefert den Wert eines zusätzlich gesetzten Headers mit der Bezeichnung <headerName> zurück. Sollte kein Header mit der angegebenen Bezeichnung gesetzt worden sein wird der Wert der Variablen <defaultValue> zurückgegeben.

**void setArea(nexus.location.Area area)** bietet die Möglichkeit die geografische Zieladresse der GeoMessage zu setzen. Der Typ der Adresse ist abstrakt. Folgende Subtypen des abstrakten Typs Area aus dem nexus.location Paket sind möglich: Circle, Segment, Polygon, GeoPolygon. Sollte ein Subtyp von Area als Argument übergeben werden, der von dieser Methode nicht unterstützt wird, wird NotSupportedException geworfen. Der Defaultwert des Zielgebietes ist null. D.h. ohne Festlegung einer Zieladresse kann die Nachricht nicht gesendet werden.

**Area getArea()** gibt das aktuell gesetzte geografische Zielgebiet der GeoMessage zurück.

**void setSubject(String subject)** ermöglicht das Setzen eines Subjects (Überschrift) ähnlich des Subjects einer email. Wird dem Anwender eine Liste von GeoMessages angezeigt, werden vorzugsweise die Subjects der Nachrichten zur Identifizierung der GeoMessages verwendet.

**String getSubject()** liefert die aktuell gesetzte Überschrift (subject) einer GeoMessage als String zurück.

**Circle getBoundingCircle()** liefert ein Objekt vom Typ nexus.location.Circle zurück, daß einen Kreis repräsentiert, der das gesetzte Zielgebiet umgibt. Diese Funktion wird vom GeoClient verwendet, da das geografische Routingverfahren der GeoNodes nur auf Kreisen oder zum Koordinatensystem achsenparallelen Rechtecken basiert. Einer GeoNode kann dabei die Nachricht mit einem Kreis (bounding circle) oder einem Rechteck (bounding box) übergeben werden. Siehe dazu das Kommunikationsprotokoll zwischen GeoClient und GeoNode.

**Segment getBoundingBox()** liefert ein Objekt vom Typ nexus.location.Segment zurück, daß ein Rechteck mit zum Koordinatensystem achsenparallelen Kanten repräsentiert. Siehe Erklärung zu getBoundingCircle().

**void setContent(String content)** setzt den Inhalt (Text) der Nachricht. Obwohl im Normalfall der Inhalt einer Nachricht aus einem Text bestehen kann, kann auch ein Bild oder ein anderes Objekt gesetzt werden. Bei der Übermittlung der Nachricht

zwischen GeoClient und GeoNode wird die GeoMessage in das sog. Base64-Format encodiert und bei Empfang vom GeoClient wieder decodiert. Damit bleiben Steuer-codes, die sonst vom XML-Parser ignoriert werden würden, erhalten.

**String getContent()** liefert den Inhalt einer GeoMessage als String zurück. Es bleibt der Applikation überlassen, wie der String behandelt wird.

**String toString()** liefert eine String-Repäsentation der GeoMessage. Diese Funktion wird vom GeoClient benutzt um die Nachricht in ein Datagrammpaket zu verpacken.

**static GeoMessage valueOf(String str)** erzeugt eine Instanz von GeoMessage und entnimmt dem String alle Werte der Nachricht. Diese Funktion wird vom GeoClient verwendet, um aus einem empfangenen Datagrammpaket eine GeoMessage zu erzeugen.

---

## GeoApplication

Das Paket GeoApplication beinhaltet Klassen, die den Anwendungsentwickler unterstützen, Applikationen zu erstellen, die die Fähigkeit haben, GeoMessages senden.

### Class GeoApplication

Die Klasse GeoApplication implementiert die Funktionalität, die eine Anwendung benötigt, um GeoMessages senden zu können. Dazu gehört die Implementierung des Kommunikationsprotokolls zwischen einer Anwendung und einem GeoClient. Anwendungsentwickler müssen somit nichts über das Protokoll wissen. Sie können diese Klasse sehr einfach benutzen um eine Anwendung zu erstellen indem sie GeoApplication als Superklasse oder als internes Objekt einsetzen.

**GeoApplication()** ist der Konstruktor dieser Klasse. In dieser Methode werden Anfangswerte interner Variablen des Objektes gesetzt. Beispielsweise wird der Port auf dem mit einem GeoClient kommuniziert werden kann eingelesen. Es wird angenommen, daß sich ein betriebsbereiter GeoClient auf dem lokalen Rechner befindet. Kann kein Port aus Konfigurationsfiles gelesen werden wird Port 4001 als Defaultwert angenommen.

**final void detectGeoClient()** implementiert das Protokoll mit dem eine Anwendung Kontakt zu einem GeoClient findet. Kann kein GeoClient kontaktiert werden, wird die Exception NoConnectionToGeoClientException geworfen. Diese Methode muß erfolgreich ausgeführt werden bevor die Methoden registerApplication, unregisterApplication oder sendMessage aufgerufen werden. Wird eine der genannten Methoden vor detectGeoClient() ausgeführt, wird diese Methode automatisch aufgerufen.

Nach erfolgreicher Ausführung dieser Methode befindet sich die Anwendung in einem Zustand in welchem sie GeoMessages senden kann.

**final void registerApplication(String name, int port)** registriert die Anwendung beim GeoClient unter einem Namen. Dem GeoClient wird damit mitgeteilt, daß eine Anwendung diesen Namens auf Port <port> auf Nachrichten wartet. Ankommende GeoMessages, die als Zielanwendung diesen Namen gesetzt haben, werden an diesen Port gesendet.

Konnte der GeoClient nicht erreicht werden wird `NoConnectionToGeoClientException` geworfen.

Meldet der GeoClient, daß bereits eine Anwendung unter diesem Namen angemeldet ist, wird `GeoApplicationAlreadyRegisteredException` geworfen.

Meldet der GeoClient, daß eine andere Anwendung bereits auf diesem Port auf GeoMessages wartet, wird `GeoApplicationAlreadyRegisteredException` geworfen.

**final void unregisterApplication()** macht eine Registrierung rückgängig. Der GeoClient wird kontaktiert und darüber informiert, daß die Anwendung nicht mehr für ankommende GeoMessages zu Verfügung steht. Dabei wird der Name, der bei Anwendung angegeben wurde, verwendet. Konnte der GeoClient nicht erreicht werden wird `NoConnectionToGeoClientException` geworfen.

Meldet der GeoClient, daß keine Anwendung unter diesem Namen registriert wurde, wird `GeoApplicationNotRegisteredException` geworfen.

**final void unregisterApplication(String name)** bietet die gleiche Funktionalität wie `unregisterApplication()`. Zusätzlich kann hier ein Name einer Anwendung angegeben werden. Es kann möglich sein, daß ein Anwendung eine andere beim GeoClient abmelden soll, da sie diese ersetzt (z.B. da es sich um eine neuere Version handelt). Diese Methode kann die gleichen Exceptions werfen wie `unregisterApplication()`.

**final void sendMessage(GeoMessage geomessage)** ist die Methode, mit der die GeoApplication eine GeoMessage sendet. Dabei muss die korrekt erzeugte GeoMessage als Argument übergeben werden. Der GeoClient wird kontaktiert und die Nachricht wird protokollgerecht übergeben.

Konnte kein GeoClient erreicht werden wird `NoConnectionToGeoClientException` geworfen.

**final void sendReceiveACK(InetAddress address, int port)** muss aufgerufen werden, wenn eine GeoMessage vom GeoClient empfangen wurde. Wie eine Anwendung ein Nachricht empfängt, bleibt dem Anwendungsprogrammierer überlassen. Empfohlen wird die Verwendung eines `DatagramSocketListeners` aus dem `Communication` Paket von `GeoCast`. Der Empfang einer GeoMessage muss dem GeoClient mit dieser Methode quittiert werden, da der GeoClient sonst vermutet, daß die Nachricht verloren ging und weiter versucht, die GeoMessage an die Applikation auszuliefern.

Scheitern mehrere Versuche, nimmt der GeoClient an, daß die Anwendung nicht mehr läuft und entfernt sie aus seiner Liste der empfangsbereiten GeoApplications. Die Argumente <address> und <port> müssen dem empfangenen DatagramPacket entnommen werden, da der Socket auf dem der GeoClient die GeoMessage sendete und nun auf ein ReceiveACK wartet, nicht mit dem bekannten Service Port übereinstimmt.

---

## Exceptions

In diesem Package sind Exceptions definiert, die von Methoden der GeoApplication geworfen werden können. Diese Exceptions sind:

**GeoApplicationAlreadyRegisteredException** wird geworfen, wenn ein GeoClient bei einer Registrierung meldet, daß eine Anwendung mit dem mitgelieferten Namen schon registriert wurde.

**GeoApplicationNotRegisteredException** wird geworfen, wenn ein GeoClient bei einem Unregister-Anfrage meldet, daß keine Anwendung unter dem gesendeten Namen registriert wurde.

**PortAlreadyInUseException** wird geworfen, wenn ein GeoClient meldet bei einer Registrierung, daß dieser Port schon durch eine, auf GeoMessages wartende, Anwendung belegt ist.

---

## Class DummyGeoApplication

Die Klasse DummyGeoApplication demonstriert die einfache Verwendung von GeoApplication als Superklasse und implementiert das DatagramSocketListenerUser - interface um GeoMessages zu Empfangen. DummyGeoApplication arbeitet folgende Schritte ab:

1. Initialisierung und Start eines DatagramSocketListeners
2. Registrierung der Anwendung unter dem als erstes Argument angegebenen Namen und dem als zweites Argument angegebenen Port
3. Erzeugung einer GeoMessage mit dem als drittes Argument angegebenen Text
4. Senden der GeoMessage an den lokalen GeoClient
5. DummyGeoApplication wartet einige Sekunden auf eventuell eintreffende GeoMessages
6. Für den Fall, daß GeoMessages eintreffen werden diese ausgegeben.

## 7. Abmelden der Anwendung beim GeoClient

---

### GeoClient

Das Paket GeoClient beinhaltet alle Klassen die für den Betrieb eines GeoClients notwendig sind. Der eigentliche GeoClient setzt sich aus mehreren Klassen zusammen. Diese Teilklassen sind eigenständige Threads, die dafür sorgen, daß Anfragen, nebenläufig zum Hauptprozeß des GeoClient, bearbeitet werden können.

Folgendes Bild stellt den Aufbau und Zusammenhang dieser Teilklassen dar.

Abbildung 17: Aufbau und Zusammenhang der GeoClient-Komponente

Der GeoClient besteht aus zwei Threads, die Nachrichten von anderen Komponenten des Systems entgegennehmen.

Für Requests von Applikationen (GeoApplications) ist der ApplicationRequestsHandler zuständig. Diese Klasse startet zwei weitere Klassen von Threads um Requests nebenläufig zu bearbeiten. Bei Empfang eines *GeoApplicationRegisterRequests* startet der ApplicationRequestsHandler einen ProcessRegisterRequestThread. Trifft ein *GeoApplicationSendRequest* ein, wird ein ProcessSendRequestThread gestartet.

Für Nachrichten von GeoNodes existiert der zweite Hauptthread des GeoClients. Der GeoNodeMessagesHandler empfängt Nachrichten auf einem Multicastkanal und startet ProcessReceiveGeoMessagesThreads um sie zu verarbeiten.

### Class GeoClient

Die Klasse GeoClient hat drei wichtige interne Klassen. Diese sind: ApplicationRequestsHandler, GeoNodeMessagesHandler und GeoClientToGeoNodeImAliveInformer. Diese drei Klassen sind Threads die nebenläufig Anfragen von Applikationen und GeoNodes bearbeiten.

Zwei der Klassen und ihre Aufgaben wurden schon grob beschreiben. Eine detailliertere Beschreibung findet sich in den Unterkapiteln zu diesen Objekten.

**GeoClient()** initialisiert ein Objekt des Typs `GeoClient`. Der Konstruktor versucht die aktuelle Position des `GeoClient`s zu ermitteln. Ist kein GPS-Sensor verfügbar, wird versucht die Position aus einer Konfigurationsdatei zu lesen. Schlägt auch diese Methode fehl, wird eine Defaultposition (`GeodeticCoordinate(48.72247,9.12446,0.0)`) angenommen, welche in etwa dem Gebäude der Fakultät Informatik der Universität Stuttgart entspricht (Stand Feb. 2001).

Da für das Senden von `GeoMessages` `GeoNodes` notwendig sind, wird versucht eine `GeoNode` zu kontaktieren. Dazu wird der Port auf dem `GeoNodes` erreichbar sein sollten aus einem Konfigurationsfile gelesen. Sollte dies nicht möglich sein, wird Port 4002 als Defaultwert angenommen. Als Internetadresse der `GeoNode` wird die Adresse des lokalen Subnetzes verwendet.

Empfängt der `GeoClient` kein Acknowledge auf einen *GeoClientToGeoNodeRequest* wird automatisch eine `GeoNode` gestartet.

Der Konstruktor initialisiert und startet einen `ApplicationRequestsHandler`, einen `GeoNodeMessagesHandler` und einen `GeoClientToGeoNodeImAliveInformer`. Alle drei Objekte sind `Threads`.

**void startupGeoNode()** ist die Methode, die eine `GeoNode` startet. Diese Methode wird vom Konstruktor aufgerufen. Da das Starten einer `GeoNode` Zeit aufgrund des Austauschs von `Request` und `Acknowledge`-Nachrichten Zeit kostet, wird zu diesem Zweck ein weiterer `Thread`, `GeoNodeStartupThread`, gestartet.

**boolean sendGeoClient\_to\_GeoNode\_Request()** implementiert den Teil des *GeoClientToGeoNodeRequest* des spezifizierten Protokolls zwischen `GeoClient` und `GeoNode`. Wenn ein *GeoNodeToGeoClientACK* empfangen wurde, gibt die Methode `true` zurück, wird ein Timeout von 10 Sekunden erreicht ist das Ergebnis `false`.

**InetAddress getGeoNodeAddress()** liefert die Adresse der erreichten `GeoNode`. Diese Methode wird vom `ProcessSendRequestThread` aufgerufen, wenn `GeoMessages` an die `GeoNode` übergeben werden.

**int getGeoNodePort()** gibt die Nummer des Ports auf dem die `GeoNode` auf Nachrichten wartet zurück. Auch diese Methode wird vom `ProcessSendRequestThread` aufgerufen. Da `GeoClients` und `GeoNodes` mobil sein können, kann sich die Adresse und der Port der, für den Aufenthaltsort des `GeoClients`, in Frage kommenden `GeoNode` ändern. Eine Aktualisierung vor dem Senden an die `GeoNode` ist also notwendig.

**GeodeticCoordinate getPosition()** liefert die aktuelle Position des `GeoClient`s als geodätische Koordinate. Diese Methode wird vom `ProcessReceiveGeoMessageThread` aufgerufen. Da `GeoClients` mobil sein können, kann sich die Position ständig ändern.

Bei Eintreffen einer GeoMessage muss die aktuelle Position abgefragt werden, um sie mit dem Zielgebiet der GeoMessage zu vergleichen.

**boolean unregister(String applicationName)** ist die Methode des GeoClients, die eine Anwendung aus der Liste der bereiten GeoApplications entfernt. Diese Methode wird vom ApplicationRequestsHandler aufgerufen. Sollte sich herausstellen, daß diese Methode zu viel Zeit kostet und damit den ApplicationRequestsHandler zu lange blockiert, kann ein ProcessUnregisterThread entworfen werden, der diese Funktion erfüllt. Wurde keine Applikation mit dem Namen <applicationName> in der Liste gefunden, wird false zurückgegeben, sonst true.

---

### Class ApplicationRequestsHandler

Die Aufgabe des ApplicationRequestsHandler wurde schon grob beschrieben. Das Interface DatagramSocketListenerUser wird von dieser Klasse implementiert. Ein DatagramSocketListener hört auf einem bekannten ServicePort auf Requests von Applikationen. Entsprechend den Requests werden Threads gestartet, die diese Requests verarbeiten und den Applikationen Antworten schicken.

**ApplicationRequestsHandler(GeoClient myGeoClient, PrintStream log)** ist der Konstruktor dieser Klasse. Als erstes Argument wird eine Referenz auf ein existierendes GeoClient-Objekt übergeben. Diese Referenz ist notwendig, da Methoden zur Abfrage der aktuellen Position des GeoClients und der vom GeoClient kontaktierten GeoNode aufgerufen werden müssen, da diese Informationen dynamisch sind. Soll dieses Objekt Eintragungen in ein Logfile machen, kann ein PrintStream als zweiter Parameter übergeben werden.

**void receiveDatagram(DatagramPacket packet)** implementiert die Methode des DatagramSocketListenerUser - Interface. Empfangene Datagramme werden auf ihren Typ hin untersucht. Handelt es sich beim empfangenen Request um einen *GeoApplicationDetectionRequest* wird sofort ein *DetectionRequestACK* an den Sender des Pakets geschickt. Handelt es sich um einen *GeoApplicationRegisterRequest* wird ein *ProcessRegisterRequestThread* gestartet und ihm das Datagramm übergeben. Ist es ein *GeoApplicationUnregisterRequest*, wird versucht, den Eintrag der Applikation aus der Lister der empfangsbereiten Anwendungen des GeoClients zu entfernen, Je nach Ergebnis dieser Operation wird ein Acknowledge oder eine Abweisung zurückgesendet. Wird ein *GeoApplicationSendRequest* empfangen, startet diese Methode einen *ProcessSendRequest* und übergibt das Datagrammpaket zur Weiterverarbeitung.

---

### Class **ProcessRegisterRequestThread**

Dieser Thread bekommt den empfangenen `GeoApplicationRegisterRequest` und versucht die Anwendung in die Liste der empfangsbereiten `GeoApplications` des `GeoClient` einzutragen.

**ProcessRegisterRequestThread(GeoClient myGeoClient, DatagramPacket packet)**  
wird eine Referenz auf den `GeoClient` und das Datagrammpaket übergeben.

**run()** ist die implementierte Methode der abstrakten Superklasse `Thread`. In dieser Methode wird die Hauptarbeit geleistet. Es wird nachgeschaut, ob sich schon eine Anwendung mit dem angegebenen Namen und Port beim `GeoClient` eingetragen hat. Wenn nicht, wird sie eingetragen und ein *GeoClientRegisterACK* an die Applikation gesendet. Bei Belegung des Anwendungsnamens wird die Nachricht *ApplicationAlreadyRegistered*, bei Belegung des Ports *PortAlreadyInUse* zurück gesendet.

---

### Class **ProcessSendRequestThread**

Dieser Thread bekommt den empfangenen `GeoApplicationSendRequest`. Die `GeoMessage` dieses Requests wird an die `GeoNode` weitergeleitet.

**ProcessSendRequestThread(GeoClient myGeoClient, DatagramPacket packet, InetAddress)**  
wird neben der Referenz des `GeoClients` und des Datagrammpakets auch die Adresse und der Port der aktuell zuständigen `GeoNode` übergeben.

**run()** ist auch hier die Methode, in der die Hauptarbeit geleistet wird. Der Algorithmus durchläuft folgende Schritte:

1. Die `GeoMessage` wird aus dem empfangenen Datagramm extrahiert
  2. Das Zielgebiet der `GeoMessage` wird decodiert
  3. Ein umschliessendes Rechteck oder der Umkreis des Zielgebietes werden errechnet
  4. Die Nachricht an die `GeoNode` wird erzeugt und gesendet
  5. Thread wartet auf eine Antwort der `GeoNode`. Wurde nach Ablauf eines Timeout keine Antwort erhalten wird die Nachricht erneut gesendet
  6. Traf ein *GeoNodeToGeoClientACK* ein, wird ein *GeoClientSendACK* an die Applikation gesendet.
-

## Class `GeoNodeMessagesHandler`

Der `GeoNodeMessagesHandler` wartet auf Nachrichten von `GeoNodes`. Da der `GeoClient` nicht blockiert werden soll, wenn ein `receive` auf den `MulticastSocket` aufgerufen wird, ist diese Klasse ein eigenständiger Thread.

**`GeoNodeMessagesHandler(GeoClient myGeoClient)`** implementiert den Konstruktor der Klasse. Als einziges Argument erhält der Thread eine Referenz auf seinen `GeoClient`. Adresse und Port des Multicastkanals auf dem `GeoClient`s auf Nachrichten von `GeoNodes` hören wird aus einem Konfigurationsfile gelesen. Ist dies nicht möglich werden die Defaultwerte 230.0.0.1 für die Multicastgruppe und Port 4000 angenommen. Ein `DatagramSocketListener` wird erzeugt und gestartet.

**`void receiveDatagram(DatagramPacket packet)`** implementiert die Methode des Interface `DatagramSocketListenerUser`. Wird eine Nachricht von einer `GeoNode` empfangen, startet diese Methode einen `ProcessReceiveGeoMessageThread` und übergibt das empfangene Datagramm.

---

## Class `ProcessReceiveGeoMessageThread`

Dieser Thread verarbeitet ankommende `GeoMessages`. Da in kurzen Absänden `GeoMessages` von `GeoNodes` eintreffen können wurde die Verarbeitung auf diesen eigenständigen Thread ausgelagert. Damit soll eine Blockierung des `GeoNodeMessagesHandler` verhindert werden, da dieser sonst Nachrichten von `GeoNodes` "überhören" könnte.

**`ProcessReceiveGeoMessageThread(GeoClient myGeoClient, String geom)`** initialisiert den Thread. Neben der Referenz auf den `GeoClient` wird dem Konstruktor die Nachricht der `GeoNode` als String übergeben. Aus diesem String wird die eigentliche `GeoMessage` extrahiert. Vom `GeoClient` wird die Liste der aktiven Anwendungen kopiert.

**`void run()`** ist die Methode in der die meiste Arbeit getan wird.

1. Zuerst wird das Zielgebiet der `GeoMessage` mit der aktuellen geografischen Position des `GeoClient`s verglichen. Liegt die Position des `GeoClient`s ausserhalb der geografischen Adresse kann die Nachricht verworfen werden.
2. Es wird geprüft, ob die Anwendung, für die diese `GeoMessage` bestimmt ist, beim `GeoClient` registriert wurde. Handelt es sich bei der `GeoMessage` um eine email (spezielle Zielanwendung), kann diese an den Benutzer gesendet werden.
3. Die Nachricht wird an den, für die Zielapplikation gespeicherten Port gesendet.
4. Der Thread wartet auf ein `ReceiveACK` von der Applikation. Verstreicht ein Timeout, wird ein zweites Mal versucht die Nachricht zuzustellen.

## GeoMessenger

Die Komponente GeoMessenger ist eine Beispielanwendung für den praktischen Einsatz von GeoCast. Der Hauptteil des Codes implementiert die grafische Benutzeroberfläche. Da die Gestaltung und Realisierung von Benutzerschnittstellen nicht das Thema dieser Studienarbeit ist, wird diese Komponente hier nur sehr grob beschreiben. Für weitere Informationen kann die Dokumentation des Codes (mit javadoc generierte Seiten) hinzugezogen werden.

Da auf eine Trennung der Implementierung von Funktionalität und grafischer Oberfläche geachtet wurde, ist die Anwendung GeoMessenger in zwei Teile untergliedert. Die Klasse GeoMessenger\_Impl enthält die Implementierung der Funktionalität der Anwendung. Die grafische Schnittstelle zum Benutzer (GUI) ist in der Klasse GeoM\_GUI implementiert. Für die Benutzerschnittstelle existieren noch weitere Klassen, die Komponenten für die Interaktion mit dem Anwender realisieren.

### Class GeoMessenger

GeoMessenger initialisiert ein Objekt vom Typ GeoMessenger\_Impl und ein Objekt vom Typ GeoM\_GUI. Da beide Objekte jeweils gegenseitig Referenzen aufeinander benötigen, werden diese in der main - methode von GeoMessenger zugewiesen. Das Hauptfenster der GUI wird angezeigt.

### Class GeoMessenger\_Impl

Die Klasse GeoMessenger\_Impl erweitert die Klasse GeoApplication. Damit besitzt sie sämtliche Methoden, die für das Kontaktieren eines GeoClients, das Registrieren einer Anwendung beim GeoClient und das Senden von GeoMessages notwendig sind. Erläuterungen zu diesen Funktionen siehe Kapitel 8.1 GeoApplication auf Seite 42.

GeoMessenger\_Impl implementiert das Interface DatagramSocketListenerUser. Ein DatagramSocketListener wird dazu benutzt auf Nachrichten vom GeoClient zu warten.

**GeoMessenger\_Impl()** initialisiert das Objekt und seine Variablen. Ein DatagramSocketListener wird instanziiert und gestartet. Als nächstes versucht der Konstruktor den GeoMessenger beim lokalen GeoClient unter dem Namen "GeoMessenger" und dem Port des DatagramSocketListeners zu registrieren. Sollte der Name oder der Port beim GeoClient schon belegt sein, terminiert das Programm sofort mit einer Fehlermeldung. Wurde kein GeoClient gefunden, startet das Programm ohne die Fähigkeit GeoMessages zu senden. Der Konstruktor liest aus einer Konfigurationsdatei Daten des Users die für das Anzeigen seiner Emails notwendig sind. Der dort angegebene Mailserver wird kontaktiert und die aktuellen Emails geladen.

## **GeoNode**

Dieses Paket wird von Gerald Grau im Rahmen seiner Diplomarbeit [Grau00] entworfen und implementiert. Für Dokumentation siehe [Grau00].

## 8.2 Kommunikationsprotokoll zwischen GeoApplication und Geo-Client

**DetectionRequest**

## **RegisterRequest**

## SendMessageRequest

### **8.3 Kommunikationsprotokoll zwischen GeoClient und GeoNode**

Für Nachrichten zwischen GeoClient und GeoNode wurden Schnittstellen vereinbart, die auf XML-Notation basieren. Die konkreten Document Type Definitions (DTD) befinden sich im Anhang und werden in der folgenden Darstellung der Kommunikationsprotokolle nur durch ihre Namen repräsentiert.

## **GeoClientToGeoNodeRequest**

**GeoClientToGeoNodeMessage**

**GeoNodeToGeoClientMessage**

## 9 Fazit - Ausblick

In dieser Studienarbeit wurden Konzepte für geografisches Routing, das Senden, Weiterleiten und Empfangen von Nachrichten aufgrund geografischer Zieladressen, vorgestellt. Bereits existierende Systeme ([Navas97], [Finn87]) sind prototypische Realisierungen eines GeoCast Systems. Im Hinblick auf mögliche Anwendungsszenarien zeigt sich, daß einige Konzepte erweiterungsfähig sind. Es ist wünschenswert, neben zweidimensionalen Gebieten auf der Erdoberfläche auch die dritte Dimension, die Position der Grundfläche und die Höhe des Zielraumes, in das Adressierungskonzept zu integrieren. In Zusammenarbeit mit Gerald Grau, dessen Diplomarbeitsthema es ist, sog. GeoNodes, Komponenten die dafür sorgen, daß eine Nachricht in das adressierte Zielgebiet geleitet wird, zu entwickeln, wurde ein in das Projekt Nexus eingebettetes GeoCast System realisiert.

Dabei wurden einige Aspekte ausgelassen, die in Zukunft für Erweiterungen dieses GeoCast Systems interessant sind. Die Möglichkeit mobile Gebiete, Busse, Bahnen, Schiffe oder PKWs zu adressieren wird von unserem System noch nicht unterstützt.

Ein weiteres Problem, das Heute schon beim Email-Service vorhanden ist, ist das Spamming. Darunter versteht man das ungewollte Empfangen von Nachrichten, mit wertlosem Inhalt oder Werbung. Ein Berechtigungskonzept für das Senden von GeoMessages kann eine der Gegenmaßnahmen zur Bekämpfung von Spam sein. Beispielsweise könnte die Größe und Lage des Gebietes eingeschränkt werden, in das eine Person senden kann. Oder man darf nur an die nahe Umgebung senden. Lösungen für dieses Problem müssen bei Bedarf in GeoCast eingebaut werden.

Weitere Verbesserungen sind auf der Seite des Clients zu erwarten. Die Auswahl eines dreidimensionalen Gebietes, Gebäude, Räume in Gebäuden, könnte durch bessere Benutzerschnittstellen unterstützt werden. Dienste, die Karten mit logischen Informationen und in drei Dimensionen visualisieren, sind im Entstehen und haben gute Anwendungsaussichten in GeoCast.

## Abbildungsverzeichnis

1	Verteilung ortsbedingter Informationen an die relevante Umgebung . . . . .	6
2	Adressierung zur Steuerung des Verkehrs . . . . .	7
3	Warnungen werden an die unmittelbare Umgebung des Gefahrenpunktes gesendet . . . . .	7
4	Beispiel: Medizinischer Hilferuf . . . . .	8
5	Eine mögliche Oberfläche eines Wetterservices für Handydisplays . . . . .	9
6	Vergleich des VIT-Konzepts mit GeoCast . . . . .	9
7	Berechnung der Position und Zeit mit 4 Satellitensignalen . . . . .	13
8	Meridianstreifen des Gauß-Krüger-Systems . . . . .	16
9	Umkreis (geografisches Objekt, Radius) : geografisches Objekt . . . . .	18
10	Polygon mit Höhe und Abstand zu NN . . . . .	19
11	Adressierung von Objekten die sich in eine Richtung bewegen . . . . .	21

---

12	Adressierung von Objekten die sich auf ein Objekt zu bewegen . . . . .	21
13	Polygon(Hauptbahnhof, Neues_Schloss, Rathaus, Punkt(L49.545,B9.841), Keplerstr) . . . . .	22
14	Verwendung logischer Operatoren . . . . .	23
15	Überblick über die GeoCast Komponenten . . . . .	28
16	Punkte P2 und P5 liegen innerhalb eines Polygons . . . . .	31
17	Aufbau und Zusammenhang der GeoClient-Komponente . . . . .	45

## Tabellenverzeichnis

## A XML Messages - Schnittstelle zwischen GeoClient und GeoNode

Die folgende Definition der Nachrichten zwischen GeoClient und GeoNode wurde in Zusammenarbeit mit Gerald Grau erstellt.

### GeoClientToGeoNodeMessage

**Verwendung:** Diese Nachricht wird vom GeoClient an die GeoNode gesendet, wenn von einer Applikation eine Nachricht an den GeoClient übergeben wurde.

**Wird beantwortet mit** GeoNodeToGeoClientACK oder GeoNodeToGeoClientNACK

```
<?xml version="1.0" standalone="yes"?>
<!--
Lifetime:
    in seconds
Mode:
    BoundingBox
    BoundingBoxCircle
GeoPoint:
    longitude (-180 (west), +180 (east))
    latitude (-90 (south), +90 (north))
    (no height)
    First Point is lower left or center point,
    second is upper right or point on the circle with same longitude
Message:
    message that will be delivered
-->
```

```
<!DOCTYPE GeoClientToGeoNodeMessage [
  <!ELEMENT GeoClientToGeoNodeMessage
(Lifetime,Mode,GeoPoint,GeoPoint,Message)>
  <!ELEMENT Lifetime (#PCDATA)>
  <!ELEMENT Mode (#PCDATA)>
  <!ELEMENT GeoPoint (Longitude,Latitude)>
  <!ELEMENT Longitude (#PCDATA)>
  <!ELEMENT Latitude (#PCDATA)>
  <!ELEMENT Message (#PCDATA)>
]>
```

```
<GeoClientToGeoNodeMessage>
  <Lifetime>10</Lifetime>
```

```

    <Mode>BoundingCircle</Mode>
    <GeoPoint>
        <Longitude>34.12345</Longitude>
        <Latitude>-23.12333</Latitude>
    </GeoPoint>
    <GeoPoint>
        <Longitude>-2.23</Longitude>
        <Latitude>1.23</Latitude>
    </GeoPoint>
    <Message>This is the message as a long string</Message>
</GeoClientToGeoNodeMessage>

```

## GeoClientToGeoNodeRequest

- 1. Verwendung:** Diese Nachricht wird vom GeoClient gesendet um herauszubekommen, ob eine GeoNode erreichbar ist.
- 2. Verwendung:** Während der GeoClient läuft, sendet er regelmässig einen Request, um der GeoNode seine aktuelle dynamische Position mitzuteilen.

**Wird beantwortet mit** GeoNodeToGeoClientACK

```

<?xml version="1.0" standalone="yes"?>
<!--
Lifetime:
    seconds
GeoPoint:
    longitude (-180 (west), +180 (east))
    latitude (-90 (south), +90 (north))
    (no height)
    First Point is lower left or center point,
    second is upper right or point on the circle with same longitude
-->

```

```

<!DOCTYPE GeoClientToGeoNodeRequest [
    <!ELEMENT GeoClientToGeoNodeRequest (Lifetime,GeoPoint)>
    <!ELEMENT Lifetime (#PCDATA)>
    <!ELEMENT GeoPoint (Longitude,Latitude)>
    <!ELEMENT Longitude (#PCDATA)>
    <!ELEMENT Latitude (#PCDATA)>
]>

```

```

<GeoClientToGeoNodeRequest>
    <Lifetime>10</Lifetime>

```

```
<GeoPoint>
  <Longitude>34.12345</Longitude>
  <Latitude>-23.12333</Latitude>
</GeoPoint>
</GeoClientToGeoNodeRequest>
```

## GeoNodeToGeoClientMessage

**Verwendung:** Diese Nachricht wird von der GeoNode an einen bestimmten Multicast-Kanal gesendet, an dem alle GeoClients in der Service Area dieser Node hören. Diese Nachricht wird von den GeoClients nicht beantwortet, da es für die GeoNode transparent ist, welche und wieviele GeoClients sich in ihrer Service Area aufhalten.

```
<?xml version="1.0" standalone="yes"?>
<!--
ID:
    unique (time_counter)
Message:
    message that will be delivered
-->

<!DOCTYPE GeoNodeToGeoClientMessage [
  <!ELEMENT GeoNodeToGeoClientMessage (ID,Message)>
  <!ELEMENT ID (#PCDATA)>
  <!ELEMENT Message (#PCDATA)>
]>

<GeoNodeToGeoClientMessage>
  <ID>17-0112-23</ID>
  <Message>content of the message</Message>
</GeoNodeToGeoClientMessage>
```

## GeoNodeToGeoClientACK

```
<?xml version="1.0" standalone="yes"?>
<GeoNodeToGeoClientACK></GeoNodeToGeoClientACK>
```

## GeoNodeToGeoClientNACK

```
<?xml version="1.0" standalone="yes"?>
<GeoNodeToGeoClientNACK>RegisterServer is not
available</GeoNodeToGeoClientNACK>
```

## Literatur

- [Imielinski99] Tomasz Imielinski, Julio C. Navas  
*GPS-Based Geographic Addressing, Routing and Resource Discovery*,  
Communications of the ACM, April 1999/Vol. 42 Nr. 4
- [Navas97] Julio C. Navas, Tomasz Imielinski  
*GeoCast - Geographic Addressing and Routing*,  
Computer Science Department, Rutgers, the State University, MOBICOM 97.
- [Tanenbaum96] Andrew S. Tanenbaum  
*Computer Networks*  
Vrije Universiteit Amsterdam, Prentice Hall
- [GPS] Peter H. Dana *Global Positioning System Overview*  
[http://www.colorado.edu/geography/gcraft/notes/gps/gps\\_f.html](http://www.colorado.edu/geography/gcraft/notes/gps/gps_f.html)
- [Nexus1] Dieter Fritsch, Darko Klinec, Steffen Volz  
*Positioning and Data Management Concepts for Location Aware Applications*  
Institute for Photogrammetry, University of Stuttgart
- [Finn87] Gregory G. Finn  
*Routing and Addressing Problems in Large Metropolitan-scale Internetworks*  
ISI Research Report, University of Southern California, 1987
- [IPv6RFC1883] S. Deering, R. Hinden  
*Internet Protocol Version 6, Specification*  
RFC 1883 (+RFC 1884) Ipsilon Networks, Xerox Parc, 1995
- [ActiveBadge] Igor Bokun, Krzysztof Zielinski *Active Badges - The Next Generation*  
Institute of Computer Science, University of Mining and Metallurgy (AGH), Al.  
Mickiewicza 30, 30-059 Krakow, Poland  
<http://iris.ics.agh.edu.pl/ABng/>
- [Map] Sinn Kim, Jong-Hwan Kim, Ik-Hwan Hyun *Development of a Map Matching  
Algorithm for Car Navigation Systems using Fuzzy Q-factor Algorithm*  
EE, Dept. KAIST, Taejon Korea und Technical Center, Daewoo Motor Co., Inchon,  
Korea
- [Grau00] Gerald Grau *Diplomarbeit: "Entwicklung effizienter Methoden zur geographischen  
Nachrichtenweiterleitung"*  
Universität Stuttgart - Fakultät Informatik