

Prüfer: Prof. Dr. Kurt Rothermel
Betreuer: Dipl.-Inf. Alexander Leonhardi
Dipl.-Geogr. Steffen Volz

begonnen am: 12. April 1999
beendet am: 7. Oktober 1999

CR-Klassifikation: C.2.4, H.3.4, H.3.5, J.2.4

Diplomarbeit Nr. 1768

Entwicklung einer Server- Komponente für ein räumliches Modell in Nexus

Johannes Schützner

Fakultät Informatik
Institut für Parallele und
Verteilte Höchstleistungsrechner
Universität Stuttgart
Breitwiesenstraße 20-22
D-70565 Stuttgart

Kurzfassung

Ein Großteil der Informationen im Internet besitzt einen mehr oder weniger großen Bezug zu einem konkreten physischen Ort. Zur Unterstützung der Entwicklung von Anwendungen, die diesen Bezug ausnutzen, bietet sich eine Plattform an, die räumliche Basisdienste zur Verfügung stellt. Die Zielsetzung des Nexus-Projekts, in das diese Arbeit eingebettet ist, besteht in der Entwicklung einer solchen Plattform.

Um räumliche Basisdienste anbieten zu können, bedarf es eines Modells des physischen Raums. Die Server-Komponente, die in dieser Arbeit zu entwickeln ist, verwaltet räumliche Modelle, die um so genannte virtuelle Objekte ergänzt sind. Virtuelle Objekte sind nur über die Plattform wahrnehmbar. Sie stellen an bestimmten Positionen Informationen bereit. Die Server-Komponente bietet als Dienste unter anderem räumliche Analysen und Navigationsunterstützung an. Weiter ist eine Manipulation der realen Welt über die Plattform möglich.

Die vorliegende Arbeit umfasst die Entwicklung einer prototypischen Server-Komponente, die auf einem Geo-Informationssystem aufbaut. Außerdem definiert sie die Dienste, welche die Server-Komponente anbietet. Schließlich untersucht sie konzeptionell die Architektur eines föderierten Systems von mehreren Server-Komponenten.

Abstract

The vast majority of the information content of the internet is in relation to some physical location. In order to support the development of applications, which exploit this fact, it makes sense to offer a platform that provides fundamental spatial services. The aim of the Nexus-project, which embeds this thesis, is to develop such a platform.

It requires a model of the physical space to be able to make spatial services available. The server-component, which is to be developed as a part of this thesis, manages spatial models that are enriched by so-called virtual objects. Virtual objects can only be perceived via the platform. At augmented locations they provide information. Among others, the server-component offers spatial analyses and navigation support. Also, the real world can be manipulated by the platform.

This thesis encompasses the development of a prototypical server-component that is based on a geographical information system. Moreover, it defines the spatial services of the server-component. Finally, it examines the architecture of a federated system of multiple server-components from a conceptual point of view.

Inhaltsverzeichnis

1 Einführung	1
2 Verwandte Ansätze	6
2.1 Ursprung Internet	6
2.2 Ursprung Kontextbewusstsein	8
2.3 Ursprung GIS	13
2.4 Zusammenfassung	15
3 Geo-Informationssysteme	16
3.1 Grundlagen	16
3.1.1 Raumbezogene Objekte	18
3.1.2 Analysefunktionalität von Geo-Informationssystemen	22
3.1.3 Ausgabe raumbezogener Objekte	23
3.2 Kategorisierung von GIS-Anwendungen im Internet	24
4 Randbedingungen durch die Mobilität	27
4.1 Mobile Rechner	27
4.1.1 Klassifikation mobiler Rechner	27
4.1.2 Probleme mobiler Rechner	30
4.2 Drahtlose Kommunikation	30
4.2.1 Drahtlose LANs	31
4.2.2 Drahtlose WANs	32
4.2.3 Probleme drahtloser Kommunikation	34
5 Anforderungen an die Server-Komponente	36
5.1 Überblick über das Use Case-Modell	37
5.1.1 Vorhandene Aktoren	37
5.1.2 Grobbeschreibung des Gesamtmodells	39
5.2 Die einzelnen Use Cases	41
5.2.1 Use Case des Aktors Ortsdienst	41
5.2.2 Use Cases des Aktors raumbezogener Klient	42
5.3 Ergänzende Anforderungen	47
5.3.1 Physische Anforderungen	47
5.3.2 Entwurfsvorgaben	48
5.4 Zusammenfassung der wichtigsten Anforderungen	48
6 Definition der Schnittstelle	49
6.1 Struktur der Schnittstelle	49
6.2 Elemente der Schnittstelle	50

6.2.1 Basisschnittstelle	50
6.2.2 Benutzer-API	51
6.2.3 Management-API	57
6.2.4 HTML-basierte Schnittstelle	58
6.3 Vergleich mit der Open GIS Service Architecture	59
7 Konzeptionelle Architektur	61
7.1 Architektur des SMS	61
7.2 Integration in Gesamtarchitektur	64
7.3 Föderation	66
7.3.1 Einbettung der Föderation	67
7.3.2 Räumliches Modell	68
7.3.3 Föderierte Realisierung der Dienste	72
7.4 Kommunikationsinfrastruktur	74
8 ArcView-basierte Architektur	76
8.1 Auswahl einer Systemkonfiguration	76
8.1.1 Auswahlkriterien	76
8.1.2 Beurteilung potenzieller Konfigurationen	77
8.2 Beschreibung der Architektur	80
8.2.1 Einführung in Avenue	80
8.2.2 Architektur auf Basis des ArcView-IMS	81
8.2.3 Eingesetzte Skripte	84
8.2.4 Erforderliche Tabellen und globale Variablen	87
8.2.5 Architektur der Benutzer-API	88
9 Implementierungsaspekte	92
9.1 Implementierungsumfang	92
9.2 Implementierung der Avenueskripte	93
9.3 Implementierung der APIs	96
9.4 Beispielapplikation	98
10 Zusammenfassung und Ausblick	101
A Anhang	104
A.1 Glossar des Use Case-Modells	104
A.2 Definition der Benutzer-API	105
A.3 Definition der Management-API	109
A.4 Implementierungsumfang	109
Literatur	113

Produktreferenzen 119

Abbildungsverzeichnis

Abbildung 1.1: Erweiterte Gebiete in Nexus (siehe Hohl et al., 1999).....	2
Abbildung 1.2: Nexus-Komponenten	3
Abbildung 2.1: Architektur des Generic Context Servers.....	12
Abbildung 3.1: Aufbau von Geo-Informationssystemen	17
Abbildung 4.1: Klassen mobiler Rechner	28
Abbildung 4.2: Konzepte drahtloser WANs	32
Abbildung 5.1: Aktoren des SMS	38
Abbildung 5.2: Use Case-Diagramm: Gesamtmodell.....	39
Abbildung 6.1: Struktur der Schnittstelle.....	49
Abbildung 7.1: Komponenten des SMS.....	62
Abbildung 7.2: Arbeitsweise der Home Servers	63
Abbildung 7.3: Integration des SMS in Nexus-Plattform	65
Abbildung 7.4: Ablauf der Föderation	66
Abbildung 7.5: Räumliche Beziehungsverhältnisse von Modellen	70
Abbildung 7.6: Inhaltliche Beziehungsverhältnisse von Modellen	70
Abbildung 8.1: Architektur basierend auf ArcView-Internet Map Server (Teil 1)	82
Abbildung 8.2: Architektur basierend auf ArcView-Internet Map Server (Teil 2)	83
Abbildung 8.3: Aufrufhierarchie der Avenueskripte	85
Abbildung 8.4: Zeichenkette, die das Dispatch-Skript erhält	86
Abbildung 8.5: Architektur der Benutzer-API.....	89
Abbildung 8.6: Klassen für räumliche Objekte.....	90
Abbildung 8.7: Dynamische Betrachtung der Architektur der Benutzer-API	91
Abbildung 9.1: Szenario, indem die Antwort des SMS die API nicht erreicht.....	96
Abbildung 9.2: Benutzungsoberfläche der Beispielapplikation.....	98

Tabellenverzeichnis

Tabelle 3.1: GIS-Internet-Kategorien	25
Tabelle 5.1: Raumbezogene Ereignisse des SMS	45
Tabelle 6.1: Dienst der Basisschnittstelle	50
Tabelle 6.2: Kurzübersicht über die Dienste dieser Klasse.	51
Tabelle 6.3: Kurzübersicht über Dienste dieser Klasse.	53
Tabelle 6.4: Kurzübersicht über die weiteren Dienste dieser Klasse	54
Tabelle 6.5: Kurzübersicht über die Dienste der Klasse Navigation	54
Tabelle 6.6: Kurzübersicht über die Dienste der Klasse Sachdaten	55
Tabelle 6.7: Kurzübersicht über die Dienste der Klasse Manipulation.	55
Tabelle 6.8: Kurzübersicht über die Ereignisse der Klasse Ereignisse.	56
Tabelle 6.9: Kurzübersicht über die Ereignisse der Klasse Grafik	56
Tabelle 6.10: Kurzübersicht über die Dienste der Management-API.	57
Tabelle 7.1: Kombinierbare Beziehungsverhältnisse	71
Tabelle 7.2: Dienste, die eine Föderation erfordern	72

1 Einführung

Motivation

Wissenschaftler von der University of Kent in Canterbury prognostizieren, dass in der nahen Zukunft praktisch jeder Rechner seinen Standort kennen wird (siehe Brown, 1998). Betriebssysteme werden Applikationen diese räumliche Information so gebräuchlich anbieten wie heutzutage das Datum (siehe Imielinski und Navas, 1999).

Das Wissen über den eigenen Standort wird *Location Awareness* oder *Raumbezug* genannt. Raumbezug kann beispielsweise dazu verwendet werden, um Druckaufträge immer zum räumlich nächsten Drucker zu senden. Eine frühe Arbeit, die Informationen mit geografischen Positionen in Verbindung setzte, stammt von Fitzmaurice (1993). Die grundlegende Idee dabei war, dass an einen PDA (Personal Digital Assistant) Informationen über einen Gegenstand übermittelt werden, sobald er in die Nähe des Gegenstands gehalten wird.

Raumbezug kann auch im Internet verwendet werden. Es wird behauptet, dass 80% der Information im Internet einen Bezug zu einer räumlichen Position besitzt. Dieser Sachverhalt lässt sich ausnutzen, um den im Allgemeinen ungeordneten Informationsgehalt des Internets zu strukturieren. Benutzer können bei bestimmten Aufgaben unterstützt werden, indem ihnen der Zugriff auf jene Informationen erleichtert wird, die in Bezug zu ihrem aktuellen Aufenthaltsort oder zu einem vorgegebenen Ort stehen.

Das Nexus-Projekt greift dieses Konzept im Rahmen einer DFG-Forschergruppe unter Beteiligung

- des Instituts für Parallele und Verteilte Höchstleistungsrechner (IPVR),
- des Instituts für Photogrammetrie (IfP) und
- des Instituts für Nachrichtenvermittlung und Datenverarbeitung (IND)

der Universität Stuttgart auf. Ziel ist die Entwicklung einer Plattform, die raumbezogenen Applikationen räumliche Basisdienste zur Verfügung stellt (siehe Hohl et al., 1999). *Raumbezogene Applikationen* haben Kenntnis über ihre eigene Position und die Position von anderen Objekten. Die Plattform soll des Weiteren die Kooperation raumbezogener Applikationen unterstützen.

Aufgrund der Mobilität der Menschen und deren potenziell permanentem Informationsbedarf geht der Nexus-Ansatz davon aus, dass Benutzer mobile Rechner wie Notebooks oder PDAs verwenden. Die Kommunikation mit der Server-Plattform findet dabei gewöhnlich drahtlos über Systeme wie GPS oder später UMTS statt.

Die Plattform von Nexus basiert auf Modellen von so genannten *Erweiterten Gebieten* (*Augmented Areas*). Dabei handelt es sich um geografische Regionen der physischen Welt, die um virtuelle Objekte ergänzt sind und denen jeweils eine Instanz des Nexus-Systems sowie der anzubietende Informationsinhalt zugeordnet werden. Erweiterte Gebiete sind keine isolierten Entitäten, da sie sich zu einer globalen *Erweiterten Welt* (*Augmented World*) integrieren lassen. Sie können sich räumlich umschließen und überlappen (vgl. Abb. 1.1).

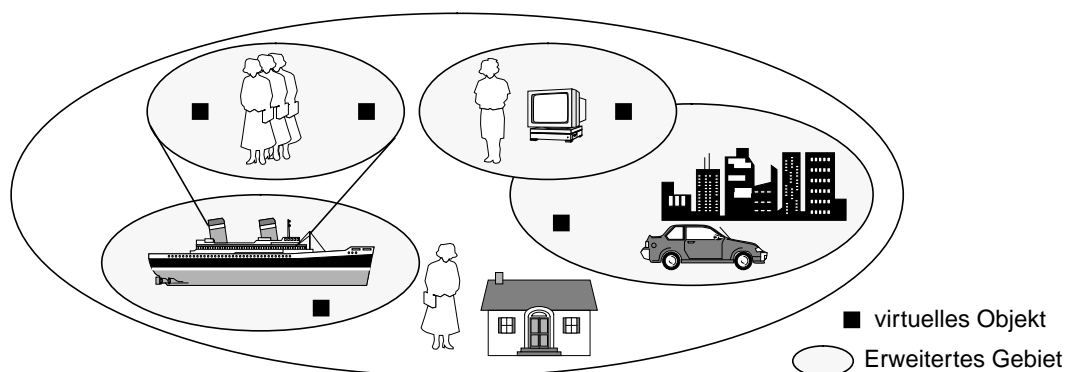


Abbildung 1.1: Erweiterte Gebiete in Nexus (siehe Hohl et al., 1999)

Die Repräsentation eines Erweiterten Gebiets wird als *Modell des Erweiterten Gebiets* bezeichnet. Der raumbezogene Teil dieses Modells ist das *räumliche Modell*. Es verwaltet Position, Form und Attribute von statischen, mobilen und virtuellen Objekten. Im Gegensatz zu statischen Objekten (wie Häuser oder Straßen) und mobilen Objekten (wie Menschen) kommen virtuelle Objekte in der realen Welt nicht vor und werden nur über die Nexus-Plattform wahrgenommen. Sie dienen der Verknüpfung von Information mit räumlichen Positionen. Virtuelle Litfasssäulen beispielsweise stellen strukturierte Informationen an bestimmten Orten zur Verfügung (siehe Fritz, 1999). Ein Benutzer kann auf diese Informationen zugreifen, falls er sich mit seinem (mobilen) Rechner an einem solchen Ort befindet. Das räumliche Modell wird durch Sensorsysteme aktualisiert, die den Zustand der realen Welt bestimmen. Über die Nexus-Plattform sollen sich Objekte auch manipulieren lassen. Haustüren könnten über die Plattform zum Beispiel geöffnet werden.

Einen groben Überblick über die gesamte Nexus-Architektur liefert Abbildung 1.2 (siehe Leonhardi, 1999). Der *Information Service* stellt den Informationsgehalt virtueller Objekte bereit. Dabei kann er auch externe Informationsräume wie das *World Wide Web* einbinden. Die *Föderation* (engl. *Federation*) bietet eine einheitliche Schnittstelle nach außen an und verteilt Anfragen auf die *Spatial Model Server*, welche die erforderlichen räumlichen Teilmodelle besitzen. Aufgabe eines *Spatial Model Servers* ist die Verwaltung von räumlichen Modellen. Weiter erlaubt er die räumliche Analyse und die Navigation innerhalb eines verwalteten Modells. Das *Area Service Register* gibt die *Spatial Model Server*-Instanzen an, die für ein spezifiziertes Gebiet zuständig sind. Falls bei einer Anfrage mehr als ein Server betroffen ist, greift die Föderation auf die zuständigen *Spatial Model Server* zu und integriert deren Teilergebnisse. Zur Abstraktion von Details verschiedenartiger Sensorsysteme dient der *Location Service* (siehe Wünsche, 1999)

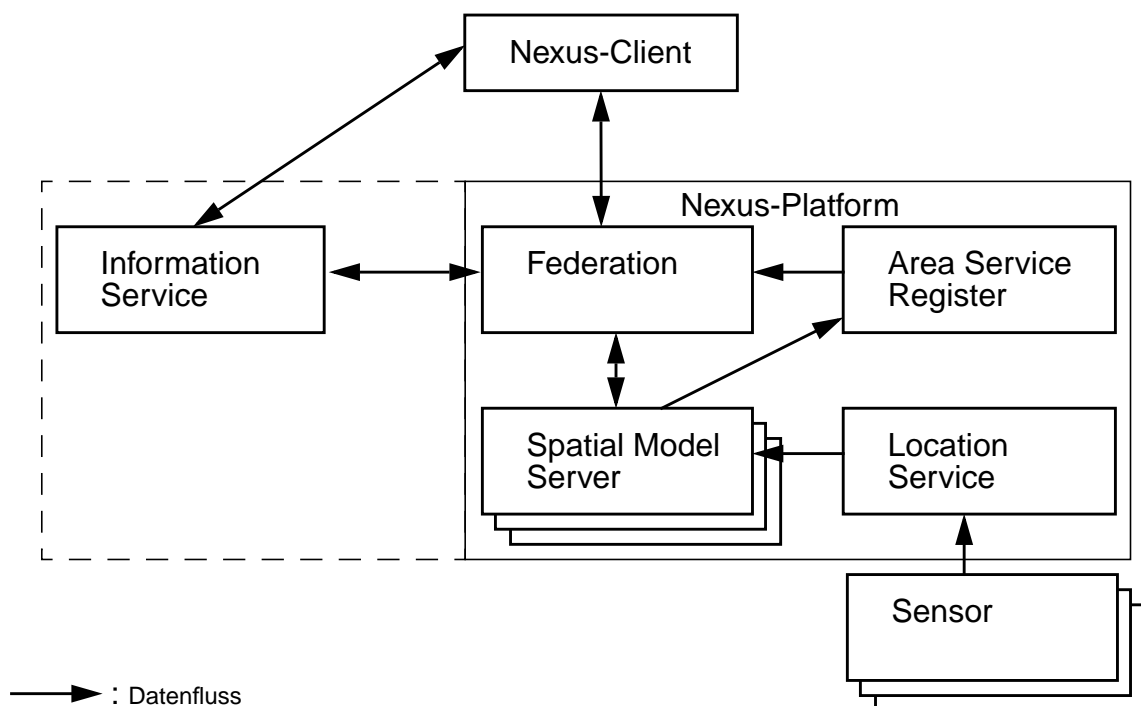


Abbildung 1.2: Nexus-Komponenten

Die *Nexus-Plattform* (engl. *Nexus-Plattform*) beinhaltet die Komponenten wie in Abbildung 1.2 skizziert. Da der *Information Service* auch externe Informationsquellen einbinden kann, ist er nur teilweise in der Plattform enthalten.

Die vorliegende Arbeit ist im Ansbubprojekt *Nexus: Mobiler und ortsabhängiger Zugriff auf Informationsdienste im Internet* angesiedelt. Dort werden unterschiedliche Ideen von Nexus erprobt und ein Prototyp der Nexus-Plattform entwickelt. Die Arbeit wird betreut von der Abteilung Verteilte Systeme des IPVR und vom IfP.

Ziele

Primäres Ziel dieser Arbeit ist die Untersuchung und Entwicklung des Spatial Model Servers. Im Folgenden wird er mit *SMS* abgekürzt. Da diese Arbeit in dem oben beschriebenen Ansbubprojekt angesiedelt ist, besitzt der SMS prototypischen Charakter.

Da ein SMS ähnliche Funktionalität wie ein Geo-Informationssystem anbieten muss, soll er auf einem solchen aufsetzen und dessen Funktionalität nutzen. *Geo-Informationssysteme* (GIS) erfassen, modellieren, verwalten, analysieren und präsentieren räumliche Daten (vgl. Kapitel 3). Verschiedene GIS-Konfigurationen sollen zunächst auf ihre Eignung als Basis des SMS untersucht werden. Die geeignetste Konfiguration wird dann ausgewählt.

Darüber hinaus ist die Schnittstelle zu definieren, die der SMS als Teil der Nexus-Plattform Applikationen zur Verfügung stellt und die den übrigen Server-Komponenten von Nexus angeboten wird. Außerdem sind verschiedene Kommunikationsmechanismen wie RPC oder CORBA zu untersuchen. Der Mechanismus, der die Anforderungen des SMS am besten unterstützt, ist zu bestimmen.

Anschließend soll die Client/Server-Architektur des SMS entworfen werden, wobei die generellen Anforderungen von Nexus zu berücksichtigen sind. Modelle sollen dabei zur Leistungssteigerung und besseren Skalierbarkeit auf mehrere Rechner verteilt werden können. Ein Konzept zum Zugriff auf mehrere Teilmodelle, oder anders ausgedrückt die Föderation von Anfragen, ist in diesem Zusammenhang zu untersuchen. Schließlich soll ein repräsentativer Teil der Funktionalität des SMS implementiert werden.

Überblick

Die Arbeit ist wie folgt strukturiert. Die Kapitel 2 bis 4 legen die Grundlagen der Arbeit dar. Kapitel 2 stellt Ansätze vor, die mit dem hier untersuchten inhaltlich verwandt sind. Kapitel 3 legt die für das Verständnis dieser Arbeit notwendigen Grundlagen der Geo-Informationssysteme dar. Das folgende Kapitel 4 formuliert Randbedingungen des SMS, die sich aus der Mobilität der verwendeten Klienten und der drahtlosen Kommunikation zwischen dem SMS und den Klienten ergeben.

Die Kapitel 5 bis 7 erörtern den prinzipiellen Lösungsansatz der Arbeit. Kapitel 5 definiert dazu zunächst die Anforderungen an den SMS. Die Anforderungen werden dabei als Use Case-Modell spezifiziert. In Kapitel 6 werden die anzubietenden Dienste der Schnittstelle festgelegt. Anschließend beschreibt Kapitel 7 die Architektur des SMS von einem konzeptionellen Standpunkt aus.

Die Kapitel 8 und 9 legen die konkrete Umsetzung des Lösungsansatzes auf Basis eines GIS dar. Kapitel 8 wählt eine GIS-Konfiguration aus und erklärt den Entwurf des SMS auf Basis dieser Konfiguration. Kapitel 9 erläutert einige wichtige Aspekte der Implementierung und stellt eine Beispielapplikation vor bevor Kapitel 10 schließlich die Arbeit zusammenfasst und einen Ausblick gibt.

2 Verwandte Ansätze

Dieses Kapitel stellt Ansätze vor, die konzeptionell ähnlich sind mit dem in dieser Arbeit untersuchten Server räumlicher Modelle (SMS). Die Betrachtung verwandter Ansätze erlaubt es, dort erworbene Erkenntnisse in diese Arbeit einfließen zu lassen. Außerdem kann auf diese Art der hier untersuchte Ansatz von anderen Ansätzen abgegrenzt werden. Nachfolgend werden Systeme und Projekte aus den Bereichen Internet, Kontextbewusstsein und GIS vorgestellt

2.1 Ursprung Internet

Dieser Abschnitt stellt Ansätze vor, deren Ursprünge aus dem Internet stammen.

WorldBoard

Das WorldBoard-Projekt stammt von der Indiana University in Bloomington (siehe Spohrer, 1998). Hintergrund von *WorldBoard* ist die Idee, Information mit jedem Punkt der Erde assoziierbar zu machen. Damit sollen Informationen, wie sie heutzutage das WWW anbietet, räumlichen Positionen zugeordnet werden können.

Anstatt Webservern verwendet der WorldBoard-Ansatz so genannte WorldBoard-Server, die Informationen in Abhängigkeit vom Ort des Benutzers, der Form des Informationsraumes um den Benutzer und der Blickrichtung des Benutzers anbieten. Der Ort des Benutzers setzt sich aus dem Längen- und Breitengrad sowie der Höhe zusammen. WorldBoard nutzt kein räumliches Modell, d.h. Informationen werden nicht raumbezogenen Objekten zugeordnet, sondern anonymen Koordinaten der Erde.

Die Blickrichtung von Benutzern erfasst das WorldBoard-System, indem es Kopf und Augen der Benutzer nachverfolgt. Zur Darstellung der Information wird die Einblendung in Brillengläser ins Auge gefasst.

Die Zielsetzung des WorldBoard-Projekts ist ähnlich zu Nexus. Im Gegensatz zum SMS ist es in Ermangelung eines räumlichen Modells bei WorldBoard jedoch nicht möglich, räumliche Analysen durchzuführen, Navigationshilfe zu erhalten oder räumliche Ereignisse zu nutzen.

Geografisches Routing

Innerhalb des *DataMan*-Projekts an der Rutgers University in New Jersey wird *geografisches Routing* untersucht (siehe Imielinski und Navas, 1999). Dieser Forschungsansatz ist Teil des *GloMo* (*Global Mobile Information System*)-Programms, das von der DARPA gesponsert wird.

Die Grundannahme ist, dass aufgrund der wachsenden Popularität von *GPS* (*Global Positioning System*) Ortsdienste in Zukunft genauso verbreitet sein werden wie Datendienste. Daher wird die räumliche Entfernung ein eminent wichtiger Suchparameter im WWW darstellen. Um dieser Entwicklung Rechnung zu tragen, entwickelt das Projekt raumbezogene Routing-Methoden. Diese Routing-Methoden erlauben es auf der Vermittlungsschicht, bestimmte räumliche Bereiche zu adressieren. Terminologisch neu eingeführt werden dazu die Begriffe Geocast und Geo-Multicast. *Geocast* bezeichnet einen Broadcast an einen bestimmten räumlichen Bereich. Ein *Geo-Multicast* ist ein Multicast an einen räumlichen Bereich.

Die räumlichen Bereiche werden mittels Punkten, Kreisen und geschlossenen Polygonen spezifiziert. Längen- und Breitengradangaben geben die Positionen einzelner (Eck- bzw. Mittel-) Punkte an. Benutzer sollen aber nicht direkt mit diesen Koordinaten arbeiten, sondern vielmehr mit Applikationen interagieren, die Karten auf grafischen Benutzungsoberflächen darstellen und die Benutzeraktionen in Koordinaten transformieren.

Die vorgeschlagenen Routing-Methoden reichen von einer Methode, welche den räumlichen Zielbereich direkt in den Header von Nachrichten integriert und die raumbezogene Router, so genannte *GeoRouter*, einsetzt bis zu einer Lösung, die auf dem *Domain Name System* (*DNS*) basiert. Letztere fügt räumliche Information zu DNS-Servern hinzu und führt neue Domänen ein, wie zum Beispiel *.geo* auf der obersten Ebene.

Geografisches Routing setzt auf einer tieferen Schicht als Nexus an. Daher sind zum Beispiel keine räumlichen Analysefunktionen vorhanden. Geografisches Routing könnte ein grundlegender Mechanismus sein, auf dem Nexus aufbaut oder der in Nexus integriert ist.

2.2 Ursprung Kontextbewusstsein

Kontextbewusste Applikationen sind Applikationen, die sich automatisch an den Kontext anpassen, in dem sie ausgeführt werden (siehe Ryan et al., 1999). Ein Beispiel für eine kontextbewusste Applikation ist ein Web-Browser, der in einem mobilen Gerät ausgeführt wird, und der die Schriftgröße vergrößert, falls der Benutzer spazieren geht (siehe Gellersen, 1999). Dem Benutzer soll damit das Lesen der Information erleichtert werden. Ein weiteres Beispiel ist eine Druckroutine, die Druckaufträge immer zum räumlich nächsten Drucker sendet.

Kontext (oder Umgebungsinformation) beinhaltet unter anderem Ort, Identität, Aktivität und Zustand von Personen, Gruppen und Objekten (siehe Salber et al., 1999). Temperatur und Zeit sowie die Software- und Hardware-Umgebung gehören ebenfalls zum Kontext einer Applikation (siehe Brown, 1996).

Für diese Arbeit ist insbesondere interessant, auf welche Art und Weise kontextbewusste Systeme räumliche Information, die ja Teil des Kontexts ist, verwalten und verarbeiten.

Cyberguide

Cyberguide ist ein Projekt der *Future Computing Environments Group* des Georgia Institute of Technology (siehe Abowd et al., 1997). Im *Cyberguide*-Projekt werden Prototypen mobiler, kontextbewusster Tourenführer entwickelt. Langfristiges Ziel des Projekts ist, dass eine Applikation weiß, wo der Tourist sich befindet, wohin er schaut, dass die Applikation Fragen des Touristen voraussehen und beantworten kann und schließlich es dem Touristen ermöglicht, mit anderen Personen sowie der Umgebung zu interagieren. Zu Beginn konzentriert sich das Projekt nur auf einen Ausschnitt des Kontexts, nämlich auf den Ort und die räumliche Orientierung des Benutzers.

Die generelle Architektur von *Cyberguide* besteht aus den folgenden Komponenten:

- *Kartograf*: Der Kartograf besitzt genaue Kenntnis über die räumliche Umgebung des Benutzers. Er wird durch Karten realisiert.
- *Bibliothekar*: Diese Komponente stellt inhaltliche Informationen über die Sehenswürdigkeiten bereit, auf die ein Tourist treffen kann.

- *Navigator*: Der Navigator positioniert den Ort des Benutzers innerhalb seiner räumlichen Umgebung. Er wird durch ein Modul realisiert, das genaue Positions- und Orientierungsinformation liefert.
- *Kommunikator*: Der Kommunikator erlaubt das Senden und Empfangen von Nachrichten, wobei Nachrichten auch rundgesendet werden können. Diese Komponente kann des Weiteren genutzt werden, um den aktuellen Ort des Benutzers einer zentralen Stelle mitzuteilen, die interessierte Benutzer dann über diesen Ort informiert.

Im Zusammenhang mit dem SMS ist die Realisierung des Kartografen und des Kommunikators relevant. Der Kartograf wird im Cyberguide-Projekt durch ein selbstentwickeltes System realisiert, das eine Karte verwaltet und darstellt. Die Karte, die Vektordarstellung verwendet (vgl. Kapitel 3.1.1), kann vergrößert und verkleinert sowie verschoben werden. Die Position des Benutzers wird durch ein Symbol auf der Karte dargestellt.

Über die Analyse- und Navigationsfähigkeiten eines GIS, auf das der SMS ja aufbaut, verfügt der Kartograf nicht. Suchfunktionalitäten sind nur rudimentär verfügbar. Darüber hinaus unterstützt es weder virtuelle Objekte noch Ereignisse. Ein weiterer Unterschied ist, dass die räumlichen Daten lokal auf dem mobilen Rechner gehalten werden. Perspektivisch ist jedoch angedacht, räumliche Daten von entfernten Servern zu besorgen.

Der Kommunikator war ursprünglich über verdrahtete Netzwerke realisiert worden und verwendete die TCP/IP-Protokolle zur Übertragung. Inzwischen wurde ein Infrarot-Netzwerk aufgebaut, das innerhalb von Gebäuden verwendet wird. Darauf aufbauend wurde ein einfaches Protokoll definiert, das drei Nachrichtentypen kennt. Es erlaubt das Versenden von E-Mails, das Rundsenden von Nachrichten an alle mobilen Rechner und das Aktualisieren von Positionsinformation.

Stick-e Notes

Stick-e Notes ist ein Ansatz der University of Kent (siehe Brown, 1996; Pascoe, 1997). In Analogie zu Post-It Notes, d.h. kleinen Notizzetteln, die an Büromöbeln, Wänden, Türen, etc. befestigt werden können, lassen sich Stick-e Notes mit bestimmten Kontexten in Beziehung setzen. Sobald ein Benutzer einen spezifizierten Kontext betritt, wird die virtuelle Notiz aufgerufen und dem

Benutzer zur Verfügung gestellt, normalerweise auf dessen PDA. Um die Erzeugung kontextbewusster Applikationen zu vereinfachen, soll die Definition von Stick-e Notes vergleichbar einfach wie das Erschaffen von Webseiten sein.

Ein besonderes Kontextelement bei Stick-e Notes sind die so genannten *Geister*. Mit Geist ist der imaginäre Begleiter eines Benutzers gemeint, der vom Benutzer frei angegeben werden kann. Der Nutzen der Geister liegt darin, dass bestimmte Kontexte vorgetäuscht werden können.

Eine Stick-e Note wird in SGML-Format beschrieben. Sie besteht einerseits aus den Marken, die den Kontext der Stick-e Note spezifizieren, wie `<with>` oder `<at>`, und damit die Triggerbedingung festlegen. Andererseits besteht sie aus dem Inhalt der Stick-e Note, der Text, Anweisungen oder ein vollständiges Programm sein kann.

Die Architektur des Stick-e Notes-Ansatzes besteht aus folgenden Komponenten:

- *SePrepare* erlaubt die Erzeugung von Stick-e Notes.
- *SeManage* verwaltet die Notizen, die angeboten werden.
- *SeTrigger* bestimmt, welche der angebotenen Stick-e Notes aktiviert werden.
- *SeShow* speichert die Notizen, die aktiviert wurden und präsentiert sie dem Benutzer.

Der Raumbezug der Stick-e Notes besteht aus Koordinaten, die sich auch in Bezug zu Karten setzen lassen. Räumliche Analysefähigkeiten, die vergleichbar mit denen des SMS sind, sieht der Ansatz jedoch nicht vor. Zwar können Triggerbedingungen wie „Bei Zusammenkunft von x und y“ spezifiziert werden, doch sind komplexe räumliche Beziehungen, wie sie GIS erlauben, nicht möglich.

Context Toolkit

Das *Context Toolkit* zielt darauf ab, die Entwicklung kontextbewusster Applikationen durch Bereitstellung einer Bibliothek zu erleichtern, die kontextbewusste Bausteine enthält (siehe Salber et al. 1999). Die Bausteine werden *Context Widgets* genannt und nehmen das Konzept der grafischen Baukästen und Bibliotheken zur Entwicklung grafischer Benutzeroberflächen auf. Sie vermitteln zwischen der Anwendung und der Umgebung, in der die Anwendung ausgeführt wird. Das Konzept des Context Toolkits stammt vom *Graphics, Visualization and Usability (GVU)-Center* am Georgia Institute of Technology.

Ein Hindernis bei der Entwicklung kontextbewusster Applikationen ist bisher, dass Kontextinformation von einer Vielzahl von verschiedenartigen Sensoren beschafft wird. Darüber hinaus erfordern Anwendungen, dass von kontextbezogenen Rohdaten abstrahiert wird, um mit der Kontextinformation sinnvoll arbeiten zu können. Eine weitere Schwierigkeit ergibt sich daraus, dass Kontextinformation von verteilten und heterogenen Quellen besorgt wird. Außerdem müssen Änderungen der Umgebung in Echtzeit festgestellt und Applikationen darüber informiert werden.

Context Widgets mindern diese Probleme, indem sie von Komplexität und Details der verwendeten Sensoren abstrahieren. Sie stellen weiter Kontextinformation auf einem Niveau zur Verfügung, das den Anforderungen von Applikationen entspricht. Sie ermöglichen auch die Wiederverwendung von Komponenten, die Kontextinformation erfassen.

Über die Realisierung eines konkreten Context Widgets, das raumbezogene Informationen verarbeitet, macht das Context Toolkit keine Aussagen. Der SMS eignet sich als ein Context Widget, das raumbezogene Informationen liefert und Teil eines Context Toolkits ist. Dies bestätigt den Nutzen und die vielfältigen Einsatzmöglichkeiten des SMS.

Generic Context Server

Die Zielsetzung bei der Entwicklung des *Generic Context Servers* liegt darin, eine generische Infrastruktur zu realisieren, welche Kontextinformation speichert, bereitstellt und archiviert (siehe Salber und Abowd, 1998). Die Infrastruktur setzt sich dabei aus mehreren Kontextservern zusammen. Der Schwerpunkt des Ansatzes besteht darin, von Kontextsensoren und Anforderungen einzelner Applikationen zu abstrahieren. Der Generic Context Server ist keine alternative Lösung für die Verwaltung eines räumlichen Modells. Er stellt vielmehr eine Architektur dar, in die der SMS integrierbar ist. Der Generic Context Server wurde wie das Context Toolkit vom GVU-Center am Georgia Institute of Technology konzipiert.

Die folgenden Punkte stellen die Entwurfsziele des Generic Context Servers dar:

- Ermöglichung des lokalen und entfernten Zugangs zu Kontextdaten auch in heterogenen Umgebungen
- Unterstützung einer Vielzahl von Applikationen, Sensoren und Operationen auf Kontextdaten
- Aufbewahrung der Geschichte von Kontextdaten

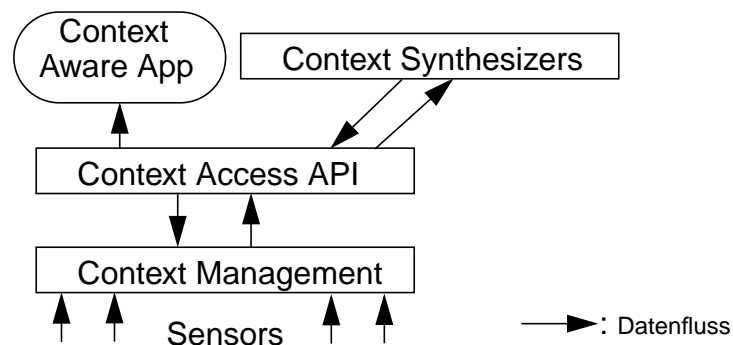


Abbildung 2.1: Architektur des Generic Context Servers

Die Architektur des Generic Context Servers besteht aus den Modulen Context Management, Context Access API und Context Synthesizern (vgl. Abb. 2.1). Das Modul *Context Management* kümmert sich um Kontexterwerb und -speicherung. Die *Context Access API* stellt sowohl lokalen als auch entfernten Zugang zu Kontextdaten bereit. Sie realisiert die Übertragung mittels HTTP und formuliert Anfragen und Antworten in XML. *Context Synthesizer* schließlich sind Module, die in die Architektur bei Bedarf eingefügt werden können. Sie abstrahieren Kontextdaten in höhere Abstraktionsschichten. Sie entdecken zum Beispiel räumliche Beziehungen, die in den Kontextdaten nicht explizit angegeben sind.

Der SMS stellt einen möglichen Context Synthesizer dar, der in die Infrastruktur des Generic Context Servers integriert werden kann. Die allgemeine Einsatzfähigkeit des SMS wird dadurch erneut bekräftigt.

Smart Badge

Smart Badge ist in einem gemeinschaftlichen Projekt der Wollongong University in Australien und der KTH in Schweden entstanden (siehe Beadle et al., 1997). Es ist ein mobiles Gerät, das Personen oder Rechnern Kontextinformation und dabei insbesondere raumbezogene Information zur Verfügung stellt. Das Gerät wurde entwickelt mit dem Ziel, eine skalierbare und raumbezogene Infrastruktur zu erschaffen und durch Experimente mit der Infrastruktur neue Erkenntnisse zu erwerben. Der Schwerpunkt der Untersuchungen liegt jedoch auf der Sensorik. Raumbezug soll durch den Smart Badge zwar hergestellt werden, doch wird nicht näher darauf eingegangen, wie der Raumbezug konkret realisiert wird.

HIPS

HIPS (Hyper-Interaction within Physical Spaces) ist ein Projekt eines europäischen Konsortiums, dem unter anderem die Università degli Studi di Siena in Italien, das GMD, das University College of Dublin sowie Alcatel Italia angehören (siehe Benelli et al., 1997). Das Hauptziel von HIPS ist die Entwicklung eines neuen Interaktionsparadigmas zur Navigation realer Räume. Als Untersuchungsgegenstand wird ein elektronischer Reiseführer entwickelt, der es erlaubt, sowohl im physischen Raum als auch im dazu in Beziehung stehenden Informationsraum simultan zu navigieren.

Der Schwerpunkt des Projekts liegt in der Gestaltung der Benutzungsoberfläche, die hauptsächlich sprachbasiert ist. Benutzer des Systems sollen gewünschte Informationen möglichst einfach erhalten können. Sie sollen ihre Erfahrungen zudem austauschen können. HIPS muss daher bidirektional arbeiten.

Benutzer erhalten kontext- und insbesondere raumbezogene Informationen (siehe Not et al., 1998). Zu stationären Objekten lassen sich Hypertextseiten assoziieren, welche Audio-, Video- oder Textinformation enthalten. Räumliche Nähe kann dabei als Querverweis zwischen Seiten dienen.

Zur Bestimmung räumlicher Positionen verwendet HIPS GPS. Es wird behauptet, dass die Navigations- und räumlichen Analysefähigkeiten von HIPS weit über die konventioneller GIS-Applikationen hinaus gehen. Konkrete Ansätze in diesem Bereich werden aber nicht erläutert.

Wie Nexus möchte auch HIPS ein räumliches Modell nutzen und unterstützt Navigation und räumliche Analyse. Es betrachtet jedoch keine Skalierbarkeits- und Lastverteilungsaspekte. Außerdem handelt es sich bei HIPS nicht um eine Plattform, die räumliche Applikationen bedient. Virtuelle Objekte sind des Weiteren nicht vorgesehen.

2.3 Ursprung GIS

Dieser Abschnitt von Kapitel 2 präsentiert Ansätze, welche GIS insofern erweitern, dass sie mobil einsetzbar sind.

FieldNote

FieldNote ist ein System, das im Rahmen des Projekts *Mobile Computing in a Fieldwork Environment* an der University of Kent entstanden ist (siehe Ryan et al., 1998). Mit Stick-e Notes von derselben Universität hat es einzig gemeinsam, das es kontextbewusst ist. Es nutzt GIS-Funktionalität und dient Forschern und Studenten im freien Feld, Daten zu speichern und Informationen gemäß den Bedürfnissen der Anwender bereitzustellen. Ein GPS-Empfänger bestimmt dabei die aktuelle Position. Beispielsweise kann *FieldNote* von Archäologen bei Ausgrabungen vor Ort eingesetzt werden.

Die Architektur von *FieldNote* beinhaltet eine statische Komponente sowie eine mobile Komponente, die auf einem PDA aufsetzt. Die statische Komponente verwaltet die im Feld gesammelten Daten und stellt der mobilen Komponente relevante raumbezogene Informationen bereit. Sie basiert auf einem selbstentwickelten GIS, das ein Datenbank-Managementsystem (DBMS) einsetzt.

Die mobile *FieldNote*-Komponente verwendet die GIS-Funktionalität der statischen Serverkomponente, indem sie eine Benutzeroberfläche anbietet und Anfragen des Benutzers an den Server weiterleitet. Die Antworten des Servers werden in Textform, als Bild oder in Form von Vektordaten zurückgeliefert. Der Server verwendet zur Generierung der Antworten Active Server Pages (ASP) oder den CGI-Mechanismus mit Perl als Skriptsprache. Es wird mit dem Gedanken gespielt, Java und JDBC zur Datenbankanbindung zu verwenden.

Wie der SMS verwendet auch *FieldNote* ein GIS zur Verwaltung raumbezogener Daten. Dies ermöglicht *FieldNote* die Analyse der raumbezogenen Daten. *FieldNote* kennt im Gegensatz zum SMS aber keine virtuellen Objekte. Es berücksichtigt weiter nicht die Randbedingungen mobiler Rechner und drahtloser Kommunikation. Die Verteilung der Arbeitslast ist darüber hinaus starr und lässt sich nicht anpassen. Skalierbarkeitsgesichtspunkte werden auch nicht untersucht. Es ist außerdem ein konkretes System, das die Verarbeitung raumbezogener Informationen im freien Feld ermöglicht. Der SMS ist dagegen Teil einer Infrastruktur, die beliebige Applikationen bedient.

Deep Map (GIS)

Deep Map ist ein Forschungsprojekt am *European Media Lab (EML)* in Heidelberg (siehe Malaka, 1999). Im Rahmen von *Deep Map* wird der Prototyp eines digitalen, persönlichen und mobilen Reiseführers entwickelt.

Ein Teilprojekt von *Deep Map* ist *Deep Map (GIS)*. Das Ziel dieses Teilprojekts ist die Entwicklung eines mehrdimensionalen GIS, das vom mobilen Reiseführer des *Deep Map*-Projekts eingesetzt werden soll. Im Gegensatz zu traditionellen GIS unterstützt es die dritte und vierte Dimension. *Deep Map (GIS)* wird mobilen Reiseführern über das WWW bereitgestellt.

Die Ansätze von SMS und *Deep Map (GIS)* sind zu einem gewissen Grad komplementär. Während der in dieser Arbeit zu entwickelnde SMS auf ein bestehendes GIS aufbaut, untersucht *Deep Map (GIS)* Geo-Informationssysteme als solche und strebt an, Konzepte zu entwickeln, welche die Zielsetzung von *Deep Map* unterstützen. Auf der anderen Seite verwendet *Deep Map (GIS)* zur Verbreitung der räumlichen Information konventionelle Internet-Techniken, während der SMS die Kommunikation auf die Bedürfnisse mobiler Rechner und drahtloser Kommunikation zu optimieren versucht. Außerdem untersucht der SMS im Gegensatz zu *Deep Map (GIS)*, wie die Arbeitslast möglichst optimal zwischen Klient und Server verteilt werden kann. Unterschiedlich ist außerdem, dass *Deep Map (GIS)* keine virtuellen Objekte kennt.

2.4 Zusammenfassung

Keiner der vorgestellten Ansätze erfüllt die Gesamtheit der Anforderungen des SMS. Manche besitzen keine räumlichen Analyse- und Navigationsfähigkeiten wie *Stick-e Notes*, andere wiederum entwickeln konkrete Anwendungen und somit keine Plattform wie beispielsweise *Field-Note*. Gesichtspunkte wie Arbeitslastverteilung und Skalierbarkeit spielen nur bei wenigen der vorgestellten Ansätzen eine Rolle.

3 Geo-Informationssysteme

Dieses Kapitel stellt die grundlegenden Konzepte von GIS vor und definiert Begriffe, die für das Verständnis dieser Arbeit notwendig sind. Darüber hinaus erläutert dieser Abschnitt, auf welche Weise ein GIS das Internet nutzen kann, um seine Funktionalität entfernten Anwendern zur Verfügung zu stellen.

3.1 Grundlagen

Begriffsklärung

Um die Definition von Geo-Informationssystemen zu ermöglichen, wird zunächst geklärt, wodurch Informationssysteme allgemein charakterisiert sind. *Informationssysteme* zielen darauf ab, Menschen bei der Arbeit mit Informationen zu unterstützen (siehe Worboys, 1995). Sie zeichnen sich dadurch aus, daß sie die Aufnahme, Verwaltung, Analyse und Präsentation von Informationen ermöglichen (siehe Bill und Fritsch, 1997). Informationssysteme unterstützen damit sowohl die Erfassung von Daten als auch deren Analyse und Nutzung. Die Datenverwaltung umfasst die Datenmodellierung, Datenstrukturierung und Datenspeicherung. Es bietet sich an, die Daten eines Informationssystems intern in einer Datenbank zu halten.

Als weitere Vorbereitung auf die Definition von Geo-Informationssystemen wird der Begriff Raumbezug konkretisiert. *Raumbezug* bedeutet, dass Daten in Verhältnis zum Bezugsobjekt Erde stehen (siehe Bill und Fritsch, 1997). Er lässt sich über zwei- oder dreidimensionale Koordinaten beispielsweise herstellen. Wenn ein exaktes Bezugssystem wie ein Koordinatensystem existiert, so liegt eine *primäre Metrik* vor. Falls der Raumbezug über ungenauere Angaben, wie zum Beispiel Straßennamen, geschieht, so ist die Metrik *sekundär*.

Geo-Informationssysteme sind Informationssysteme, die raumbezogene Daten verarbeiten (siehe Bill und Fritsch, 1997). Sie umfassen somit neben Daten auch Modellierungs- und Analysefunktionalitäten. Die Präsentation der Daten kann sowohl textuell als auch grafisch geschehen.

Die Abbildung 3.1 veranschaulicht den prinzipiellen Aufbau von GIS (siehe Worboys, 1995). Die Geo-Datenbank enthält die raumbezogenen Daten, auf die das GIS aufbaut. Sie muss nicht zentralisiert sein, sondern kann auch externe Datenbanken miteinbeziehen. Das GIS stellt Funktionen bereit, welche die Analyse der Daten direkt unterstützt. Es bietet eine Schnittstelle an, welche die Erfassung und Modellierung von Daten sowie deren Präsentation erlaubt, und Benutzer direkt mit den raumbezogenen Daten arbeiten lässt. Im Rahmen dieser Arbeit wird die Erfassung raumbezogener Daten nicht behandelt. Außerdem verfügt die Schnittstelle über Kommunikationsprimitive, die den Zugriff auf entfernte Datenbestände ermöglichen, aber auch die Kommunikation mit Applikationen realisieren können.

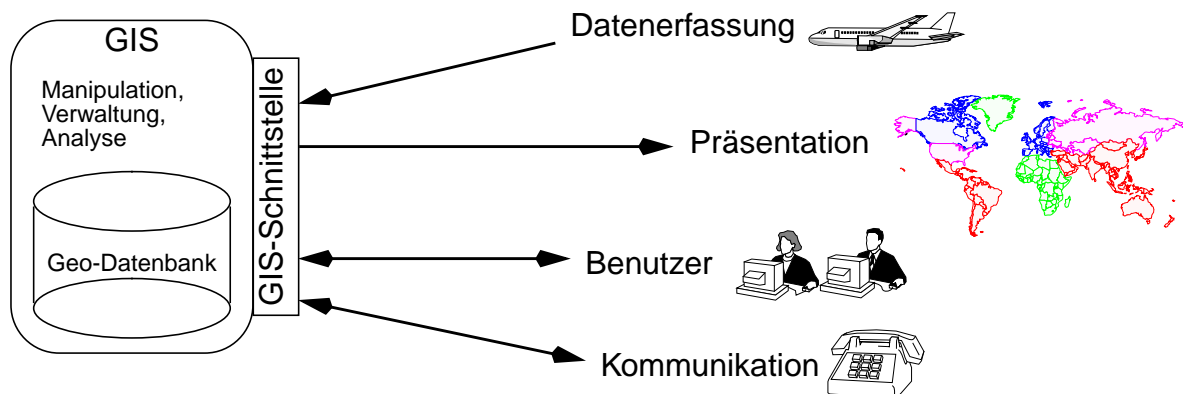


Abbildung 3.1: Aufbau von Geo-Informationssystemen

Allgemeine Anforderungen an Geo-Informationssysteme

Es gibt eine Reihe von Anforderungen, die an GIS gestellt werden. Diese Anforderungen sind im Einzelnen (siehe Bill und Fritsch, 1997):

- Die Systeme sollten eine große Menge räumlich indizierter Punkte verwalten können.
- Die zugrundeliegende Datenbank sollte es erlauben, raumbezogene Objekte hinsichtlich Existenz, Position und Eigenschaften abzufragen.
- Abfragen lassen sich aus Subabfragen rekursiv zusammensetzen.
- GIS sollten sich den unterschiedlichen Anforderungen verschiedener Benutzergruppen anpassen können.

- Sie sollten über die verwalteten raumbezogenen Objekte während der Nutzung lernen können, um zum Beispiel zuvor angewandte Regeln in neuen Anfragen anzuwenden. Dieser Aspekt spielt für diese Arbeit eine untergeordnete Rolle.

Nach dieser allgemeinen Beschreibung von Geo-Informationssystemen stellt der nächste Abschnitt die Objekte vor, welche von GIS verwaltet werden.

3.1.1 Raumbezogene Objekte

Die Entitäten eines GIS sind physisch, geometrisch oder begrifflich unterscheidbare Einheiten der Natur, die eine Identität besitzen. Man nennt sie *raumbezogene* (oder *räumliche*) *Objekte*, die — analog zur objektorientierten Programmierung — Objektklassen zugeordnet sind.

Raumbezogene Objekte bestehen aus den folgenden Charakteristika, die anschließend näher erläutert werden (siehe Bill und Fritsch, 1997):

- Geometriedaten, die den Raumbezug herstellen
- topologische Beziehungen, die Nachbarschaftsverhältnisse definieren
- thematischen Sachdaten oder Attribute
- Identifikatoren

Geometrie

Die *Geometrie* eines raumbezogenen Objekts lässt sich durch die relative Lage von Punkten sowie durch die Form des Objekts vollständig spezifizieren. Üblicherweise werden kartesische Koordinaten relativ zu einem Bezugssystem verwendet. Es ist jedoch prinzipiell auch möglich, zur Spezifikation von Objekten Entfernungen und Winkel einzusetzen. Man unterscheidet zwischen Vektor- und Rasterdarstellung.

Die *Vektordarstellung* beruht auf den Grundstrukturen Punkt und Linie, wobei sich die fundamentale Betrachtungsweise an Linien orientiert. Flächen werden als geschlossener Linienzug dargestellt. Vektorielle Darstellungen von Objekten können auch als Graphen aufgefasst werden. Dabei entsprechen die Objektpunkte den Knoten eines Graphen und die Linien den Kanten. Vorteilhaft

bei der Vektordarstellung ist die geringe Datenmenge, die anfällt, und die sich daraus ergebenden kurzen Rechenzeiten. Der logische Objektbezug sowie die logische Datenstrukturierung werden außerdem gut unterstützt.

Die *Rasterdarstellung* bezieht sich direkt auf Flächen und besitzt als einzige Grundstruktur das Pixel (Picture Element). Ihre Vorteile liegen in der besseren Handhabung von Flächen und in der schnelleren und einfacheren Datenerfassung. Nachteile sind die großen Datenmengen und Rechenzeiten, die diese Darstellungsform mit sich bringt, sowie Einschränkungen beim logischen Objektbezug und bei der logischen Datenstrukturierung. Die letzten beiden Punkte haben zur Folge, dass die Anfragemöglichkeiten an geometrische Objekte in Rasterdarstellung schlecht sind.

Raumbezogene Objekte können mehrere geometrische Dimensionen besitzen. Die Geometrie eines Objekts ist:

- *Zweidimensional (2D)*, falls sie sich ausschließlich auf die Planimetrie (x- und y-Koordinaten) beziehen und keine Höhenangaben verwendet werden.
- *Zwei-plus-eindimensional (2D+1D)*, falls die Planimetrie durch ein digitales Geländemodell komplementiert wird, das nicht mit der Lagegeometrie des Objekts verknüpft ist. Das bedeutet, dass unabhängig von räumlichen Objekten Höhenlinien in der Abbildung eingetragen sind.
- *Zweieinhalbdimensional (2,5D)*, falls zusätzlich zur Lagegeometrie die Höhe raumbezogener Objekte vorhanden ist. Diese räumliche Beschreibung ist daher abhängig von der Dichte der Lagegeometrie.
- *Dreidimensional (3D)*, falls x-, y- und z-Koordinaten für das gesamte betrachtete Gebiet in hinreichender Dichte vorliegen. Es existieren drei unterschiedliche 3D-Modelle. *3D-Linienmodelle* entstehen durch Überlagerung der Lagegeometrie mit Höhenlinien. *3D-Flächenmodelle* verwenden zur Beschreibung von Objekten meist ebene Vielecke. Am aufwendigsten sind *3D-Volumenmodelle*, die Objekte durch die Teilkörper definiert, aus denen es aufgebaut ist.
- *Vierdimensional (4D)*, falls neben den x-, y-, und z-Koordinaten auch der Zeitparameter t Berücksichtigung findet. Die vorgestellten 3D-Modelle besitzen auch hier Gültigkeit.

Topologie

Um ausdrucksstarke Datenmodelle entwickeln zu können, die imstande sind, eine große Vielfalt von Anfragen effizient zu beantworten, müssen explizit *topologische Beziehungen* zwischen Objekten spezifiziert werden. Diese Nachbarschaftsbeziehungen geben beispielsweise an, welche Eckpunkte eines Quaders durch Linien miteinander verbunden sind. Die topologischen Basiselemente sind Knoten, Kanten und Flächen.

Bei der topologischen Betrachtungsweise interpretiert man Objekte als Graphen. Zwei Schlüsselkonzepte dabei sind Inzidenz und Adjazenz. Die Elemente eines Graphen sind *inzident*, wenn sie verschiedenartig sind und direkt aneinander grenzen. Die Knoten an den Enden einer Kante sind zu dieser beispielsweise inzident. Knoten sind *adjazent*, wenn sie über eine Kante verbunden sind. Umgekehrt sind Kanten adjazent, wenn sie in den selben Knoten münden.

Mittels der Topologie läßt sich beispielsweise leicht die Frage nach den Straßen beantworten, die in einem bestimmten Ort enden. Straßen werden dabei als Kanten und der Ort als Knoten aufgefasst.

Topologische Dimensionen werden in Bezug auf die flächenhafte Ausdehnung von Objekten festgelegt. Man unterscheidet:

- *Nullzellen*, die Knoten repräsentieren
- *Einszellen*, die Linien repräsentieren
- *Zweizellen*, die geschlossene Linienpolygone umfassen und Flächen definieren
- *Dreizellen*, die dreidimensionale Elemente darstellen aus denen sich dreidimensionale Objekte aufbauen lassen

Thematik

Ein Charakteristikum von GIS ist die gemeinsame Verwaltung von Geometrie- und Sachdaten. Andere Begriffe für *Sachdaten* sind *thematische Daten* und *Attribute*. Sachdaten stellen alle nicht-geometrischen Daten von raumbezogenen Objekten dar, wie deren Namen und Eigenschaften. Ein Beispiel für ein Sachdatum ist der Name einer Straße, deren Geometrie als Linie repräsentiert wird. Eine Menge von thematisch in Beziehung stehenden Sachdaten fasst man zu einer *Thematik* zusammen. Ein geometrisches Objekt kann zu beliebig vielen Thematiken gehören.

Der Geometrie des Landes Baden-Württemberg lassen sich zum Beispiel die Thematiken Einwohner und Regierung zuordnen. Die Thematik Einwohner umfasst dann beispielsweise die Sachdaten Einwohnergesamtzahl, Anzahl der weiblichen Einwohner und Anzahl der Einwohner in ländlichen Gebieten. Die Thematik Regierung besteht demgegenüber nur aus dem Attribut Regierungspartei.

Thematiken ermöglichen auch die selektive Darstellung von Objektgruppen. Um obiges Szenario weiterzuführen, kann beispielsweise die Anforderung bestehen, dass alle Objekte dargestellt werden, die eine Einwohnerzahl besitzen. In diesem Fall wird das Land Baden-Württemberg dargestellt, wohingegen Objekte wie Flüsse oder Fabriken ignoriert werden.

Thematisches Modellieren bezeichnet die Definition, Bearbeitung und Speicherung der Thematik, die einem raumbezogenen Objekt zugrundeliegt. Die prinzipiellen Ansätze zum thematischen Modellieren sind das Ebenenprinzip und das Objektklassenprinzip.

Beim *Ebenenprinzip* werden Geometriedaten von verschiedenen Thematiken in unterschiedlichen Ebenen abgespeichert. Die Gesamtdarstellung entsteht durch Überlagerung der Ebenen. Die Ebenen werden durch den Raumbezug, d.h. durch ihre Position, verknüpft.

Das *Objektklassenprinzip* ist der modernere Ansatz. Es kann mit thematischen Zusammenhängen wesentlich flexibler umgehen. Dieses Prinzip ordnet thematische Mengen hierarchisch an, wobei auch netzwerkartige Verbindungen erlaubt sind. In einer *thematischen Menge* sind semantisch zusammengehörende Thematiken aggregiert. Die für eine Problemstellung relevanten Objekte erhält man durch Navigation durch das Netzwerk.

Analog zu Geometrie und Topologie gibt es auch bei der Thematik den Dimensionsbegriff. Im Gegensatz zum Dimensionsbegriff bei Geometrie und Topologie bezieht er sich jedoch nicht auf einzelne Objekte, sondern auf GIS, da diese die Gesamtheit der Thematiken verwalten. GIS sind *n-dimensional*, falls n verschiedene Thematiken vorliegen. Rein geometrische Darstellungen ohne zugeordnete Thematiken sind konsequenterweise *thematisch dimensionslos*.

Identifikatoren

Objektidentifikatoren, die auch *Objektschlüssel* genannt werden, bezeichnen raumbezogene Objekte eindeutig. Sie dienen auch der Zuordnung von Thematiken zu Objekten.

3.1.2 Analysefunktionalität von Geo-Informationssystemen

Nachdem die wesentlichen Prinzipien von GIS dargelegt wurden, stellt dieser Unterabschnitt Funktionalitäten von GIS vor, die der Analyse raumbezogener Daten dienen. Die Erfassung dieser Funktionalitäten ist für diese Arbeit relevant, da sie Aufschluss darüber geben, welche Aufgaben von einem eingesetzten GIS direkt übernommen werden können.

Datenabfragen

Unter dem Oberbegriff Datenabfragen fasst man geometrische, topologische und thematische Anfragen zusammen (siehe Bill und Fritsch, 1997). Zu den geometrischen Anfragen gehört beispielsweise die Frage nach Objekten innerhalb eines vorgegebenen Polygons. Eine topologische Anfrage ist die Frage nach Objekten, die auf einer Seite an einen Weg grenzen. Thematische Anfragen werden in Worboys (1995) *mengenorientierte Anfragen* genannt. Mit ihnen können Objektmengen selektiert werden, deren Thematiken bestimmte Bedingungen erfüllen. Außerdem können die Attribute eines identifizierten Objekts abgefragt werden (siehe Tscheuschner, 1997). Die vorgestellten Anfragen sind mit booleschen Operatoren kombinierbar.

Computed Geometry (COGO)

Funktionen dieser Gruppe führen Messungen und Berechnungen durch. Darunter fallen Abstände, Flächen und Volumen, aber auch Anzahl und Anteil von Objekten mit bestimmten räumlichen oder thematischen Eigenschaften.

Zonen- oder Puffergenerierung

Unter Zonen- oder Puffergenerierung versteht man die Erzeugung von Quadrat- oder Kreispuffern bzw. Quader- oder Kugelpuffern um punkt-, linien- oder flächenförmige Objekte. Zielsetzung ist die Bestimmung der Objekte, die sich innerhalb des Einflussbereichs von Objekten befinden.

Flächenverschneidung

Durch geometrische Überlagerung von Objekten erzeugt die Flächenverschneidung neue Objekte. Sie ermöglicht die Beantwortung von Fragen nach punktförmigen Objekten, die sich innerhalb eines anderen Objekts befinden. Außerdem kann sie die Anteile von linien- oder flächenförmigen Objekten ermitteln, die beim Schnitt mit einer Fläche entstehen.

Interpolation und Abstraktion

Ebene und räumliche Interpolation sowie Abstraktion ermöglichen die Abbildung von Daten in eine andere Form. Zum Beispiel können Punkthaufen als Höhenlinien dargestellt werden.

Netzwerkanalysen

Ein Netzwerk im Sinne von GIS ist ein gerichteter Graph, der aus den topologischen Informationen der raumbezogenen Objekte gebildet wird. Netzwerkanalysen führen beispielsweise Suchen nach nächsten Nachbarn und dem kürzesten Weg zwischen zwei Objekten durch. Sie sind darüber hinaus in der Lage, Netzwerke vollständig zu analysieren und simulieren.

3.1.3 Ausgabe raumbezogener Objekte

Neben den Analysefähigkeiten von GIS ist im Kontext dieser Arbeit interessant, auf welche Art und Weise raumbezogene Objekte dargestellt werden. Dies ist für den SMS relevant, da er von GIS erzeugte Bilder und Darstellungen eventuell direkt an den Klienten liefert. Die Ausgabe umfasst gewöhnlicherweise die im Folgenden beschriebenen Funktionalitäten (siehe Bill und Fritsch, 1997).

Displaymanagement

Displaymanagement beinhaltet Grundfunktionen bei der Interaktion mit der Grafik. Dazu gehören Anzeigen, Verschieben, Vergrößern/Verkleinern und Fenstertechnik.

Gestaltungsfunktionen

Üblicherweise steht eine Bibliothek von Mustern zur Verfügung, die grafische Symbole, Linienarten und Flächendarstellungen anbietet.

Grafische und textuelle Ausgabe

Neben der Karte gehören zur grafischen Ausgabe eine Großzahl anderer Darstellungen wie Längsschnitte oder Perspektiven. Die textuelle Ausgabe ermöglicht die Generierung von Listen, in denen Aspekte raumbezogener Objekte enthalten sind.

Generalisierung

Die Generalisierung befasst sich mit Problemen der Darstellung von Daten in unterschiedlichen Maßstäben (Aggregationsebenen). Vor allem bei sehr verschiedenen Maßstäben besteht das Problem der Darstellung detaillierter Daten in einem größeren Maßstab.

Bilderzeugung

GIS können aus dargestellten Karten Bilder erzeugen, die sich in andere Dokumente einbauen oder ausdrucken lassen. Es werden beispielsweise Bilder in EPS- oder JPEG-Format generiert.

Hiermit wird die grundlegende Vorstellung von GIS abgeschlossen. Der folgende Abschnitt kategorisiert die möglichen Nutzungsszenarien des Internets durch GIS.

3.2 Kategorisierung von GIS-Anwendungen im Internet

Seit einigen Jahren bestehen Bestrebungen, Internet-Dienste in GIS zu nutzen. GIS-Anwendungen im Internet können nach dem Umfang der über das Internet angebotenen Funktionalität kategorisiert werden. Die prinzipiellen Kategorien stellt dieser Abschnitt vor.

Geo-Datenserver

Der *Geo-Datenserver* übermittelt geografische Daten als Datei (siehe Fitzke et al., 1997). Er führt keinerlei Abfrage- oder Analyseaufgaben durch. Zur Verarbeitung der Daten müssen Klienten daher über Software verfügen, welche die räumlichen Daten weiterverarbeitet.

Map-Server

Map-Server liefern Karten zur Online-Visualisierung (siehe Stahl, 1997). Sie bieten jedoch nicht räumliche Daten an, die von Applikationen weiterverarbeitet werden können.

Im Fall von *statischen Map-Servern* werden ausschließlich vorgefertigte Karten angeboten. Die Klienten-Applikation steuert lediglich die Auswahl der Karten.

Interaktive Map-Server erlauben es Klienten, die Darstellungsart der Karte zu manipulieren (siehe Fitzke et al., 1997). Die Karte wird vom Server dann individuell nach den Wünschen der Klienten-Applikation erstellt und übertragen.

Online-Auskunftssystem

Online-Auskunftssysteme sind dadurch charakterisiert, dass sie vorgefertigte oder interaktiv erstellte Karten darstellen, sowie thematische und einfache geometrische und topologische Abfragen erlauben (siehe Fitze et al., 1997). Auch sie stellen keine räumlichen Daten zur Verfügung.

GIS-Funktionsserver

Der *GIS-Funktionsserver* ermöglicht den Zugang zu den Analysefähigkeiten eines GIS. Die zu verarbeitenden Daten werden vom Klienten bereitgestellt. Der GIS-Funktionsserver übermittelt die Ergebnisse aber normalerweise ohne Visualisierungsmöglichkeiten als Datei (siehe Fitzke et al., 1997).

Online-GIS

Ein *Online-GIS* bietet die gesamte GIS-Funktionalität über das Internet an (siehe Stahl, 1997). Explizit stellt es räumliche Daten zur Weiterverarbeitung in Applikationen nicht zur Verfügung.

Kategorie	Erfassung	Präsentation	Abfrage	Analyse	Lieferung
Geo-Datenserver	x	-	-	-	Daten
Map-Server	x	x	-	-	Grafik
Online-Auskunftssystem	x	x	x	-	Grafik
GIS-Funktionsserver	-	(x)	x	x	Funktionen, Grafik
Online-GIS	x	x	x	x	Grafik

Tabelle 3.1: GIS-Internet-Kategorien

Die Tabelle 3.1 fasst die Funktionalitäten der einzelnen Kategorien zusammen. Die Spalte *Erfassung* gibt an, ob eine Kategorie räumliche Daten als Datei entgegennehmen und in das Modell einbringen kann. Die Spalte *Lieferung* spezifiziert, in welcher Form Anfragen beantwortet werden. Der Wert *Funktionen* bedeutet dabei, dass Daten verarbeitet werden, die der Klient bereitstellt.

Der SMS lässt sich nicht direkt in dieses Schema der GIS-Anwendungen einordnen. Er umfasst die Funktionalität eines Online-GIS, da neben der Analyse auch die Präsentation räumlicher Daten angeboten wird. Darüber hinaus ist er jedoch in der Lage, Daten zurückzuliefern, die in räumli-

chen Applikationen genutzt werden. Darauf ist ein Online-GIS nicht ausgelegt. Außerdem ist die Fähigkeit von Online-GIS zur Erfassung räumlicher Daten nicht ausreichend, da auch einzelne räumliche Objekte manipuliert und neue virtuelle Objekte in ein räumliches Modell eingebracht werden sollen.

Damit endet die Betrachtung von GIS. Das folgende Kapitel stellt Randbedingungen an den SMS vor, die sich aus der Mobilität der Benutzer ergeben.

4 Randbedingungen durch die Mobilität

Die Applikationen, die der SMS bedient, setzen meistens auf mobilen Rechnern auf und kommunizieren mit dem Server drahtlos. Um die Dienste des Servers entsprechend diesen Randbedingungen gestalten zu können, untersucht dieser Abschnitt Eigenschaften mobiler Rechner und drahtloser Kommunikation. Diese Charakteristika werden später herangezogen, um Anforderungen an den Server zu spezifizieren.

4.1 Mobile Rechner

Mobile Rechner unterscheiden sich von stationären Rechnern durch ihre mobile Einsatzfähigkeit. Um Mobilität gewährleisten zu können, besitzen sie eine eigene Energiequelle wie einen Akku. Auch mobile Rechner lassen sich in Rechnernetze einbinden, wobei drahtlose Kommunikation benötigt wird. Dieser Abschnitt beschreibt verschiedene Klassen von mobilen Rechnern und hält allgemeine Probleme mobiler Rechner fest.

4.1.1 Klassifikation mobiler Rechner

Die Abbildung 4.1 kategorisiert mobile Rechner in drei Klassen und gibt die Hauptunterscheidungsmerkmale an. Die Einteilung darf jedoch nicht als strenge Taxonomie mit disjunkten Klassen aufgefasst werden, da permanent neue Geräte erfunden werden und die Grenzen zwischen den Rechnerklassen verwischen. Die angegebenen Klassen dienen vielmehr der Verdeutlichung der Vielzahl möglicher Geräte, die es gegenwärtig gibt und die potenzielle Applikationsplattformen darstellen.

Notebook

Notebooks sind vergleichbar mit vollwertigen stationären PCs, d.h. sie besitzen ein nur geringfügig kleiner dimensioniertes Display als PCs, eine nur wenig langsamere Taktfrequenz des Prozessors, sowie Hauptspeicher, Sekundärspeicher und externe Speicher (Diskette, CD-ROM, DVD,

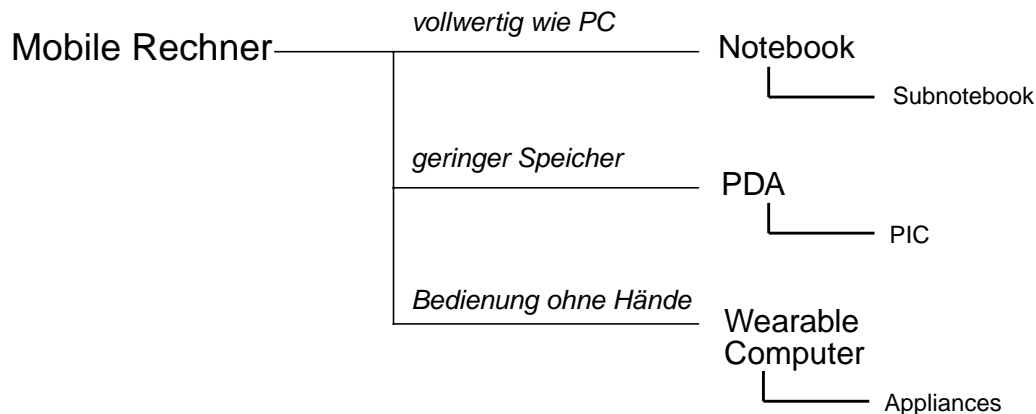


Abbildung 4.1: Klassen mobiler Rechner

etc.), dessen Leistungsabstriche zu PCs vernachlässigbar sind. Sie verwenden außerdem die gleichen Betriebssysteme wie PCs. Ihre Ausmaße sind die eines DIN-A4-Blatts in der Länge und Breite und ca. 5 Zentimeter in der Höhe.

Subnotebooks besitzen eine etwas geringere Leistungsfähigkeit als Notebooks und unterscheiden sich von diesen hauptsächlich in der Größe. Sie sind in der Länge und Breite ungefähr so groß wie ein DIN-A5-Blatt und ca. 3 Zentimeter hoch. Subnotebooks verwenden wie Notebooks die Betriebssysteme der PCs.

Personal Digital Assistant (PDA)

Der *PDA* zeichnet sich dadurch aus, dass er derzeit etwas größer als eine menschliche Hand ist und sein Leistungsvermögen nicht mit dem stationärer PCs mithalten kann. Gegenwärtig besitzen PDAs keinen Sekundärspeicher. Um Daten persistent speichern zu können, verwendet er einen nicht-transienten Hauptspeicher. Aufgrund seiner geringen Größe sind externe Speichermöglichkeiten äußerst schwierig. Es können einerseits Flash-Cards verwendet werden. Ein anderer Ansatz, der zur Zeit untersucht wird, ist der Einsatz von kleinen Festplatten, welche die PCMCIA-II-Schnittstelle ansprechen. PDAs verwenden spezialisierte Betriebssysteme wie zum Beispiel Windows CE (siehe Microsoft, 1999) oder PalmOS (siehe 3Com, 1999).

PDAs sind nicht unbedingt mit einer Tastatur ausgestattet. Als Eingabemedium wird oft ein drucksensitives Display in Kombination mit einem Stift eingesetzt. Das Display ist nur etwa handflächengroß, weshalb PDAs auch *Palmtops* genannt werden. Manche PDAs unterstützen auch die Spracheingabe.

Personal Intelligent Communicator (PIC) stellt eine Unterklasse von PDA dar. Ein PIC besitzt sowohl die Funktionalität eines PDAs als auch die eines Mobiltelefons. Der Haupteinsatzzweck besteht in Kommunikationsaufgaben (E-Mail, WWW, etc.). Da er in einem recht kleinen Gerät sowohl PDA als auch Mobiltelefon integriert, ist er meist ressourcenschwach.

Wearable Computer

Die Vision des *Wearable Computing* ist ein vollwertiger Rechner, der ständig am Körper getragen wird und der permanent verfügbar ist, um Menschen bei ihren Tätigkeiten zu unterstützen (siehe Mann, 1998). Wearable Computer schränken weder Mobilität oder Aktivitäten des Benutzers ein, noch kapseln sie ihn von seiner Umgebung ab (siehe Gellersen, 1999). Bei älteren Modellen befanden sich Prozessor und Speicher des Wearable Computers typischerweise auf dem Rücken des Benutzers. Sie werden jedoch zunehmend kleiner und unauffälliger. Die Eingabe geschieht natürlichsprachlich und die Ausgabe erfolgt über ein *Head-Mounted-Display*. Die Ausgabe kann auch in Sprachform realisiert werden. Ein *Head-Mounted-Display* hat die Form einer Brille; in sie wird die Ausgabe eingeblendet. Speziell für Aufgaben, die den Einsatz von Händen benötigen, ist der Wearable Computer interessant, da die Benutzerschnittstelle des Wearable Computers keine manuellen Handlungen erfordert.

Wearable Computers im weiteren Sinne sind kleine spezialisierte Geräte, die man mit sich führt und die sehr ressourcenschwach sein können. Diese Rechner werden je nach Aufgabe auch *Information Appliances* oder *Communication Appliances* genannt (siehe Gellersen, 1999). Ein Beispiel für letztere Gruppe ist Metronaut, der an der Carnegie Mellon University entwickelt wurde (siehe Smailagic und Martin, 1997). Metronaut bestimmt die aktuelle Position mit einem Strichcodelesegerät und gibt Navigationshilfen. Sein einziges Speichermedium ist ein 2 MB großer Hauptspeicher. Ein weiteres Beispiel für *Communication Appliances* stellen *Pager* dar, die Textnachrichten empfangen und darstellen können (siehe Tanenbaum, 1996).

Im Anschluss an diese Beschreibung mobiler Rechnerarten erörtert der nächste Abschnitt generelle Probleme mobiler Rechner.

4.1.2 Probleme mobiler Rechner

Die folgenden Aussagen beschreiben Eigenschaften mobiler Rechner. Sie sind allgemeingültig und werden auch in Zukunft ihre Gültigkeit behalten.

- Mobile Rechner sind ressourcenschwach im Vergleich zu statischen Rechnern, d.h. die Speicherkapazität ist gering und die Energiequelle endlich (siehe Satyanarayanan, 1996; Forman und Zahorjan, 1994). Trotz technologischen Fortschritts werden mobile Rechner im Vergleich zu stationären Rechnern immer ressourcenschwach bleiben.
- Mobile Rechner besitzen eine kleine Benutzeroberfläche (siehe Forman und Zahorjan, 1994). Diese Tatsache wäre zwar technologisch leicht zu ändern, größeren Geräten würde jedoch die Benutzerakzeptanz fehlen.
- Mobile Rechner sind inhärent gefährdet (siehe Satyanarayanan, 1996). Im Vergleich zu stationären Rechnern, die in Räumen verschlossen sind, ist das Diebstahlrisiko bei mobilen Rechnern um ein Vielfaches größer. Im Zusammenhang mit dieser Arbeit ist dieser Punkt nicht bedeutsam.

Der folgende Abschnitt stellt die drahtlose Kommunikation vor.

4.2 Drahtlose Kommunikation

Mobilkommunikation hat zum Ziel, dass Benutzer von einem beliebigen Ort aus und während sie sich bewegen kommunizieren können (siehe Baumann, 1998). Daher ist es nicht akzeptabel, wenn Benutzer nur über ein Festnetz Zugang zu Netzwerken erhalten. Es bedarf vielmehr drahtloser Kommunikation. Im Rahmen dieser Arbeit sind einerseits *drahtlose lokale Netze* (LANs) relevant, da sie innerhalb kleiner Gebiete Mobilkommunikation ermöglichen, die relativ hohe Übertragungskapazitäten bereitstellt. Andererseits sind in Nexus auch *drahtlose Weitverkehrsnetze* (WANs) wichtig, da über diese flächendeckende Dienste angeboten werden können.

Im Folgenden werden Technologien und Standards drahtloser LANs und WANs vorgestellt, sowie deren generelle Probleme aufgezeigt. Dabei wird des Öfteren auf IEEE 802.11 verwiesen. IEEE 802.11 definiert verschiedene drahtlose *MAC (Medium Access Control)*-Standards der Sicherungsschicht von Rechnernetzen (siehe Halsall, 1996).

4.2.1 Drahtlose LANs

Drahtlose LANs haben im Vergleich zu Festnetzen geringere Übertragungskapazitäten (siehe Forman und Zahorjan, 1994). Erstrebenswert ist, dass sie störungsfrei arbeiten, keine Lizenz erfordern sowie ein möglichst großes Gebiet abdecken. Sie können mit Infrarot- oder Funktechnologie sowie als digitale schnurlose Nebenstellenanlage realisiert werden.

Infrarot-Systeme (IR-LAN)

Der Betrieb von Infrarot-Systemen erfordert keine Lizenz (siehe Baumann, 1998). Da Infrarot-Wellen keine Mauern durchdringen können, ist die Reichweite von IR-LANs innerhalb Gebäuden auf ein Zimmer beschränkt. IR-LANs arbeiten entweder im Punkt-zu-Punkt- oder im diffusen Modus (siehe Halsall, 1996). Letzterer erlaubt die Kommunikation zwischen mehreren Punkten.

Für die Kommunikation mittels Infrarot existieren IEEE 802.11-Standards für 1 und 2 Mb/s, sowie komplexere Standards, die 4 bzw. 10 Mb/s Übertragungsrate erlauben (siehe Halsall, 1996). Letztere erfordern jedoch eine kompliziertere Empfängerelektronik. Zur Zeit entwickelt die *Infrared Data Association* (IrDA) den AIR-Standard, der eine Parametrisierung der Leistung von 250 Kb/s bei 8 Meter Reichweite bis zu 4 Mb/s Übertragungsrate bei einer Reichweite von 4 Metern erlaubt und im diffusen Modus arbeitet (siehe IrDA, 1998).

Funksysteme (Radio-LAN)

Funksysteme haben im optimalen Fall eine Reichweite von 100 bis 200 Metern und eine Datenrate von bis zu 3 Mb/s (siehe Breezecom, 1998; Lucent, 1999). Je nach Gebäudeart können diese Werte auch wesentlich kleiner sein. Im Mikrowellenbereich erreichen sie jedoch 10 bis 100 Mb/s (siehe Baumann, 1998). Radio-LANs durchdringen Mauern, verfügen aber nur über einen begrenzten Frequenzbereich. Funksysteme müssen teilweise lizenziert werden.

Für die Funkübertragung spezifiziert IEEE 802.11 Standards für 1 und 2 Mb/s Übertragungskapazität (siehe Halsall, 1996). Diese Standards werden in Bälde ergänzt durch einen Standard, der 10 Mb/s Datenrate erlaubt. Der HIPERLAN-Standard (*High PERFORMANCE Radio Local Area Network*) für Funkssysteme erlaubt im diffusen Modus Datenraten von bis zu 20 Mb/s. Die Reichweite beträgt hierbei 50 Meter.

Digitale schnurlose Nebenstellenanlagen (DECT)

Digitale schnurlose Nebenstellenanlagen (DECT: Digital Enhanced Cordless Telecommunication) basieren auf der Technologie schnurloser Telefone (siehe Baumann, 1998). Ihre Reichweite beträgt innerhalb Gebäuden 50 Meter und im Freien 300 Meter (siehe Hascher, 1999). Die Übertragungskapazität ist zur Zeit auf wenige Hundert Kb/s beschränkt. Es wird aber an Verfahren gearbeitet, die bis zu 2 Mb/s Datenrate besitzen.

4.2.2 Drahtlose WANs

Drahtlose WANs ermöglichen Funkverbindungen innerhalb einer größeren geografischen Region. Sie sind bisher hauptsächlich auf die Telefonie ausgerichtet (siehe Tanenbaum, 1996). Im Gegensatz zu drahtlosen LANs, die einen Kanal gemeinsam nutzen, verfügen drahtlose WANs über mehrere Kanäle (siehe Baumann, 1998). Drahtlose WANs sind zellulare Netzwerke, dessen Zellen Basisstationen und Funkvermittlungsstellen umfassen.

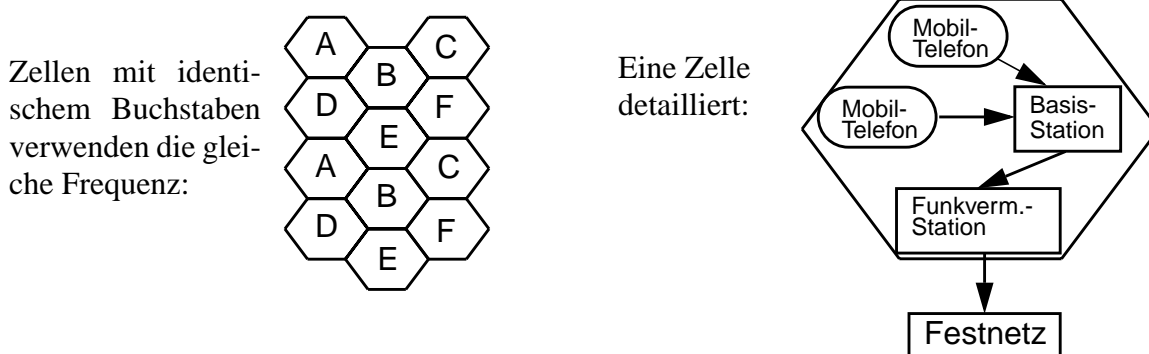


Abbildung 4.2: Konzepte drahtloser WANs

Die Aufteilung von Gebieten in kreisförmige Funkzellen bietet die Vorteile, dass mobile Endgeräte mit geringeren Sendeleistungen auskommen und durch Wiederverwendung von Frequenzen höhere Kapazitäten erreicht werden (vgl. Abb. 4.2).

Alle Mobiltelefone einer Zelle übertragen an die Basisstation, die aus einem Rechner, Empfänger- und Sendestation besteht. Sie ist mit einer Funkvermittlungsstelle verbunden, die Gespräche vermittelt und Verbindung zum Festnetz bietet.

Nachfolgend werden verschiedene drahtlose WAN-Ansätze vorgestellt.

AMPS

Das nordamerikanische System für drahtlose WANs nennt sich *Advanced Mobile Phone System* (AMPS) und ist analog (siehe Tanenbaum, 1996). Von den 832 Vollduplexkanälen, die jede Zelle dort besitzt, sind zur Übertragung von Nutzdaten nur rund 45 verfügbar. Mobile Geräte können Verbindungen aufbauen, indem sie auf einem designierten Kanal eine Verbindung beantragen. Dabei übermitteln sie auch ihre Identifikations- und Rufnummer. Die Funkvermittlungsstelle weist dann einen freien Kanal zu. Die Nettodatenrate eines Kanals beträgt 9,6 Kb/s. AMPS wird genauso wie GSM der zweiten Generation drahtloser WANs zugerechnet.

GSM

GSM steht für *Global System for Mobile Communication* und stammt aus Europa (siehe Baumann, 1998). Es überträgt Daten digital und ist grundsätzlich leitungsvermittelt. Pro Zelle sind ca. 200 Duplexkanäle verfügbar. Jeder Kanal verfügt über eine Nettodatenrate von 9,6 Kb/s. Für die Übertragung von Daten ist nachteilig, dass bei der Übergabe von Gesprächen zwischen Zellen (Handover) Verluste von etwa 0,3 Sekunden entstehen (siehe Tanenbaum, 1996). Dieses Problem wird von CDPD gelöst.

CDPD

CDPD (*Cellular Digital Packet Data*) ist ein digitaler Datagrammdienst, der auf AMPS aufsetzt (siehe Tanenbaum, 1996). Er folgt eng dem OSI-Modell. Die Nettodatenrate pro Kanal ist wiederum 9,6 Kb/s.

UMTS

Gemeinsamer Nachfolger der drahtlosen WANs der zweiten Generation ist *UMTS (Universal Mobile Telecommunications System)*. Das Ziel von UMTS ist die Realisierung einer Telekommunikationstechnologie, die verdrahtete und mobile Dienste anbietet, welche nahtlos kombinierbar sind (siehe Swain, 1994). Daher erfordert UMTS, Dienste in drahtlosen und verdrahteten Umgebungen auf die gleiche Art zu präsentieren und vielfältige Dienste und Applikationen zu unterstützen. Es ermöglicht Übertragungsraten von bis zu 2 Mb/s, wobei die tatsächlich benötigte Übertragungsrate flexibel und schnell zugewiesen wird. Raten von annähernd 2 Mb/s werden jedoch nicht flächendeckend, sondern nur innerhalb von Gebäuden und an so genannten Hot Spots angeboten. *Hot Spots* sind Gebiete mit sehr hohen Anforderungen an die Übertragungskapazität. Die minimale Datenrate, die im Freien zumindest nutzbar ist, wurde auf 144 Kb/s definiert. UMTS wird frühestens im Jahre 2002 einsetzbar sein.

Technisch wird UMTS als Kombination von CDMA und TDMA implementiert. *TDMA (Time Division Multiple Access)* ordnet jeder Station fest definierte Zeitscheiben zur Übertragung zu.

CDMA (Collision Detection Multiple Avoidance) ermöglicht jeder Station einer Zelle, über das gesamte Frequenzspektrum zu verfügen (siehe Tanenbaum, 1996). Dies wird durch einen eindeutigen m-Bit-Code erreicht, der jeder Station zugeteilt wird. Der m-Bit-Code repräsentiert für die zugeordnete Station eine Eins; die Null wird durch das Einser-Komplement des Codes dargestellt. Alle anderen Muster sind für die Station unzulässig und werden daher ignoriert. Zur Realisierung von CDMA bedarf es einer komplexen Technik. Dafür steht jeder Station bei beispielsweise 1 MHz Frequenz 1 Mb/s Datenrate zur Verfügung.

Nach der Vorstellung drahtloser LAN- und WAN-Technologien handelt der folgende Unterabschnitt von generellen Problemen bei drahtloser Kommunikation.

4.2.3 Probleme drahtloser Kommunikation

Drahtlose Kommunikation hat mit mehr Schwierigkeiten zu kämpfen als Kommunikation über Festnetze. Die Ursache liegt in der Tatsache begründet, dass die Umgebung mit den übertragenen Signalen interagiert, Signale blockiert und Signalrauschen sowie Echos erzeugt (siehe Forman und Zahorjan, 1994). Daraus ergeben sich:

- eine geringere Verbindungsqualität, d.h. kleinere Bandbreite, höhere Fehlerraten und kurzzeitige Unterbrechungen der Verbindung
- größere Latenzzeiten aufgrund wiederholter Übertragungen und anderen Gründen

Weitere Probleme sind:

- große Varianz in Leistung und Zuverlässigkeit (siehe Satyanarayanan, 1996)
- heterogene Netzwerke mit stark unterschiedlicher Netzwerkqualität (siehe Forman und Zahorjan, 1994)
- Sicherheitsrisiken, da Informationen relativ leicht in falsche Hände gelangen können (siehe Forman und Zahorjan, 1994)

Die angesprochenen Eigenschaften und Problematiken mobiler Rechner sowie drahtloser Kommunikation werden im nächsten Kapitel bei der Formulierung von Anforderungen an das zu entwickelnde System Berücksichtigung finden.

5 Anforderungen an die Server-Komponente

Dieses Kapitel beschreibt die prinzipiellen Anforderungen an den SMS aus Sicht der Benutzer. Um das Verständnis zu erleichtern, wird der SMS vorab kurz charakterisiert. Ein SMS verwaltet ein oder mehrere räumliche Modelle. Ein von einem SMS verwaltetes räumliche Modell repräsentiert mit einer bestimmten Genauigkeit einen Ausschnitt der realen Welt, der um virtuelle Objekte ergänzt ist. Wenn nicht anders angegeben, beziehen sich die folgenden Aussagen auf ein einzelnes Modell. Der SMS stellt einen Server im weiteren Sinne dar. Er muss seine Arbeit nicht jederzeit auf dem Server-Rechner verrichten, sondern kann Teile der Arbeitslast auch zum Klienten auf den Klienten-Rechner verlagern. Ziel dieser Verlagerung ist die optimale Nutzung der Ressourcen Rechenleistung und Netzwerkkapazität.

Die Föderation von Anfragen wird hier außer Acht gelassen, da die Aufgabe einer Föderationskomponente die Verteilung von Anfragen auf geeignete SMS ist. Dies hat keinen Einfluss auf die Definition der allgemeinen Anforderungen an einen SMS.

Die generellen Anforderungen ergeben sich vor allem aus den Projektzielen von Nexus (siehe Hohl et al., 1999; Hohl, 1999), aus den Randbedingungen der Mobilität (vgl. Kapitel 4) sowie aus einer vorherigen Arbeit (siehe Fritz, 1999). Anregungen ergaben sich auch aus der Betrachtung von GIS (vgl. Kapitel 3).

Die Anforderungen werden in Form von Use Cases definiert, da sich diese in letzter Zeit als Darstellungsform für allgemeine Anforderungen immer mehr durchgesetzt haben und sich insbesondere in der objektorientierten Welt einer großen Beliebtheit erfreuen.

Ein *Use Case* spezifiziert eine Abfolge von Aktionen, die das zu entwickelnde System ausführen kann und die einem bestimmten Benutzer des Systems ein überprüfbares wertschöpfendes Ergebnis liefert (siehe Jacobson et al., 1999). Die Abfolge kann dabei Varianten beinhalten. Die Sichtweise ist immer die von Benutzern. Zwischen Use Cases kann eine *Generalisierungsbeziehung* bestehen. Das bedeutet, dass der generalisierte Use Case im übergeordneten Use Case als Black-

Box verwendet wird. Die Generalisierung ist vergleichbar mit einem Prozeduraufruf in Programmiersprachen. Use Cases sind *abstrakt*, falls sie nur existieren, um von anderen Uses Cases wiederverwendet zu werden und nicht selber instanziiert werden können. Ansonsten sind sie *konkret*.

Die Benutzer des Systems werden im Use Case-Modell *Aktoren* genannt und müssen nicht unbedingt menschlicher Natur sein. Applikationen, die mit dem System interagieren, sind ebenfalls Aktoren. Für mehr Informationen über Use Cases siehe Jacobson et al. (1999).

Dieses Kapitel ist wie folgt strukturiert: Zunächst wird ein Überblick über das komplette Use Case-Modell gegeben. Dabei werden die einzelnen Use Cases identifiziert und die Aktoren des Systems vorgestellt. Im Anschluß daran werden die Use Cases genau spezifiziert. Den Abschluss dieses Kapitels bilden ergänzende Anforderungen, die sich nicht einzelnen Use Cases zuordnen lassen, sondern das gesamte System betreffen. Diese ergänzenden Anforderungen sind hauptsächlich nichtfunktionaler Art.

5.1 Überblick über das Use Case-Modell

Dieser Abschnitt stellt die Aktoren des SMS vor und gibt einen groben Überblick über die Use Cases. Während der Beschreibung der Aktoren und Use Cases werden relevante Begriffe definiert. Diese Begriffe sind im Glossar in Kapitel A.1 zusammengefasst.

5.1.1 Vorhandene Aktoren

Aktoren stellen Klassen von Benutzern mit identischen Anforderungen an das System dar. Der SMS besitzt folgende Aktoren: die raumbezogene Applikation, den Information Service, den Location Service und den SMS-Manager. Abbildung 5.1 stellt dies dar.

Raumbezogene Applikation

Die raumbezogene Applikation ist eine Applikation, die hauptsächlich auf mobilen Rechnern läuft und ihren Raumbezug über die Nexus-Plattform herstellt. Normalerweise kommuniziert sie mit dem SMS über drahtlose Verbindungen. Wie in Kapitel 4 dargestellt, besitzen mobile Rechner und

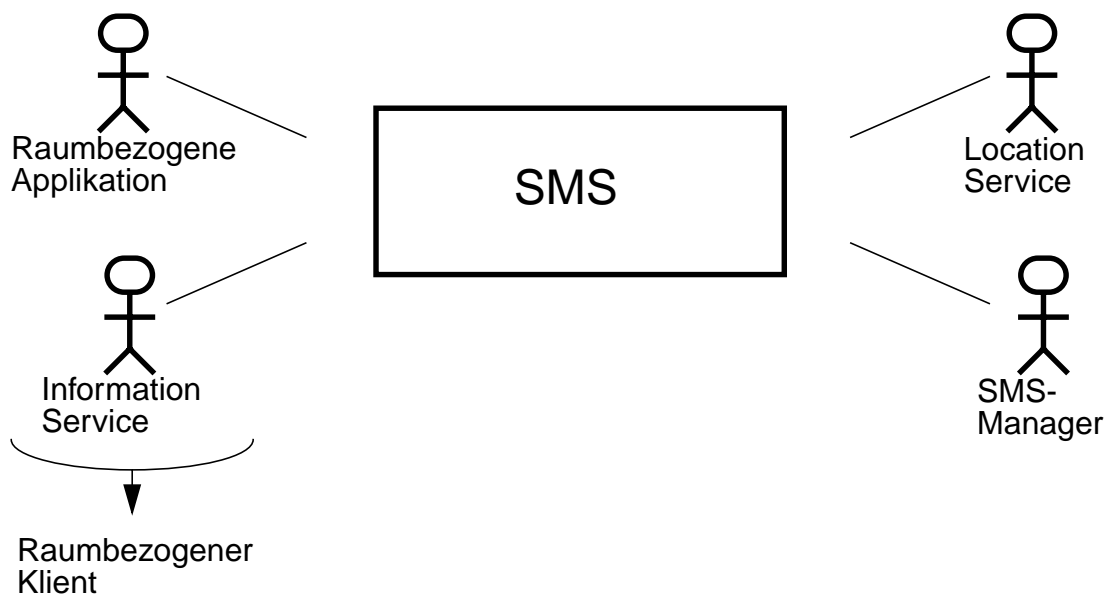


Abbildung 5.1: Aktoren des SMS

drahtlose Kommunikation potenziell stark unterschiedliche Leistungsvermögen. Daher sollen die angebotenen Dienste des SMS eine grafisch-interaktive als auch eine textuelle Darstellung der räumlichen Information ermöglichen.

Information Service

Der Information Service stellt Informationen bereit, die über den SMS einen Raumbezug erhalten. Er verwaltet den Informationsgehalt von virtuellen Litfasssäulen sowie Post-It-Notizen (siehe Fritz, 1999). Um zum Beispiel mobile Objekte mit virtuellen Objekten in Beziehung setzen zu können, konsultiert er den SMS, der ihm Informationen über das räumliche Modell liefert. Im Gegensatz zur raumbezogenen Applikation braucht bei der Kommunikation mit dem Information Service keine Rücksicht auf schlechte Übertragungsraten bzw. rechenschwache Server genommen werden.

Raumbezogener Klient

Der Information Service soll die Dienste des SMS auf die gleiche Weise wie die raumbezogene Applikation nutzen. Daher verfügt er über dieselben Use Cases wie diese. Im Folgenden wird raumbezogener Klient als Überbegriff für die raumbezogene Applikation und den Information Service verwendet, da beide Aktoren Klienten des SMS darstellen.

Location Service

Der Location Service bestimmt den Aufenthaltsort mobiler Objekte und abstrahiert von Details konkreter Positionierungssysteme. Logisch geht die Initiative zur Kooperation von SMS und Location Service normalerweise vom SMS aus.

SMS-Manager

Die Aufgabe des SMS-Managers ist die Verwaltung der räumlichen Modelle eines SMS. Er fügt zum Beispiel neue statische Objekte hinzu oder definiert den räumlichen Bereich eines Modells.

5.1.2 Grobbeschreibung des Gesamtmodells

Die Use Cases des Gesamtmodells sind in Abbildung 5.2 dargestellt. Sie sind konkret. Sie werden bis auf Aktualisiere Ort, der vom Location Service gestartet wird, alle vom raumbezogenen Klienten initiiert. Der SMS-Manager besitzt keine Use Cases, die ja wertschöpfend für Benutzer sind, da er nur administrative Aufgaben erfüllt.

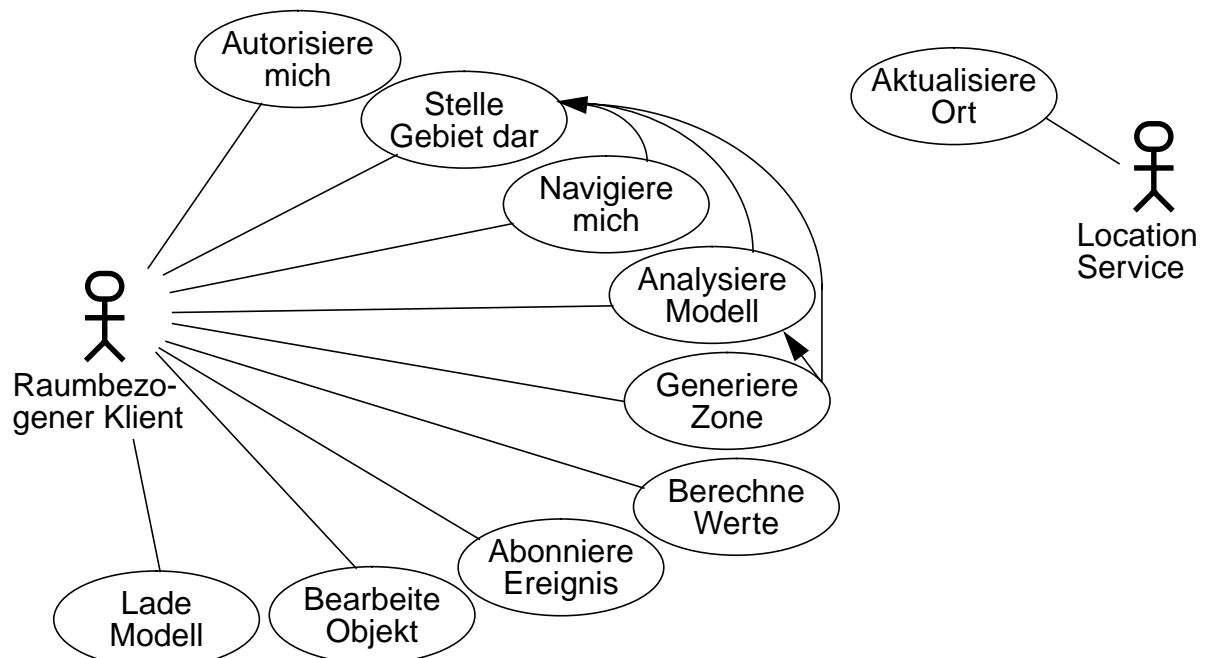


Abbildung 5.2: Use Case-Diagramm: Gesamtmodell

Der Use Case **Autorisiere mich** autorisiert den raumbezogenen Klienten, bestimmte Informationen des SMS abzurufen und eventuell Manipulationen am räumlichen Modell vorzunehmen.

Stelle Gebiet dar erlaubt dem raumbezogenen Klienten, Benutzern einen Überblick über die räumlichen Begebenheiten der gegenwärtigen Umgebung zu präsentieren. Die Umgebung kann zwei- oder dreidimensional dargestellt werden. **Stelle Gebiet dar** wird von den Use Cases **Navigiere mich**, **Analysiere Modell** und **Generiere Zone** generalisiert. Er wird von ihnen also in mindestens einem Zweig verwendet. In Abbildung 5.2 wird die Generalisierung durch gerichtete Kanten zwischen Use Cases repräsentiert.

Die Aktualisierung der Position eines mobilen Objekts ist die einzige Handlung des Use Cases **Aktualisiere Ort**. Damit der Location Service die Position eines mobilen Objekts dem SMS meldet, muss ein entsprechendes Ereignis zuvor beim Location Service registriert worden sein.

Zur Beantwortung von Anfragen nach Objekten mit bestimmten räumlichen oder attributiven Eigenschaften dient der Use Case **Analysiere Modell**.

Generiere Zone ist ein Use Case, der zwei- bzw. dreidimensionale Bereiche um Objekte bestimmt. Damit lassen sich zum Beispiel alle Banken innerhalb eines bestimmten Umkreises um Bushaltestellen ermitteln.

Um Messungen und Berechnungen im räumlichen Modell vornehmen zu können, wird der Use Case **Berechne Werte** definiert.

Der Use Case **Navigiere mich** bestimmt optimale Wege zu vorgegeben Zielpunkten und beschreibt gefundene Wege.

Raumbezogene Klienten sollen sich für räumliche Ereignisse registrieren können. Dies wird durch den Use Case **Abonniere Ereignis** realisiert. Klienten können sich etwa für das Ereignis registrieren, dass sich ein mobiles Objekt mit bestimmten Eigenschaften innerhalb eines spezifizierten Gebiets befindet.

Der Use Case **Bearbeite Objekt** umfasst zum Einen die Erzeugung und das Löschen virtueller Objekte. Mobile Objekte können außerdem registriert und abgemeldet werden. Außerdem umfasst er die Änderung von Attributen räumlicher Objekte sowie Zustandsänderungen statischer Objekte. Eine solche Zustandsänderung wäre beispielsweise das Öffnen der Haustür eines Wohnhauses.

Der letzte Use Case ist **Lade Modell**. Er erlaubt es dem Klienten, spezifizierte Teilmodelle zu laden, die vom Klienten selbst verarbeitet werden müssen.

Der folgende Abschnitt stellt die Use Cases ausführlicher vor.

5.2 Die einzelnen Use Cases

Nach Initiatoren geordnet, erläutert dieser Abschnitt die einzelnen Use Cases im Detail. Für jeden Use Case wird kurz die Motivation, der Standardpfad sowie alternative Pfade der Abarbeitung erklärt. Falls erforderlich, werden zusätzlich jeweils nichtfunktionale Anforderungen und Vor- und Nachbedingungen spezifiziert.

5.2.1 Use Case des Aktors Ortsdienst

Use Case 1: Aktualisiere Ort

Damit sich mobile Objekte jederzeit in Bezug zu anderen Objekten setzen können, soll es dem SMS ermöglicht werden, sein räumliches Modell immer auf dem aktuellen Stand zu halten.

- **VORBEDINGUNG:**

Das betroffene mobile Objekt hat sich beim SMS registriert. Weiter wurde beim Location Service das Ereignis registriert, dass eine veränderte Position des mobilen Objekts dem SMS gemeldet wird.

- **STANDARDPFAD:**

- 1) Der Ortsdienst teilt dem SMS die neue Position des mobilen Objekts mit. Die Position wird in WGS-84-, Gauss-Krueger- oder einem lokal definierten Format spezifiziert (siehe Dana, 1998).
- 2) Der SMS rechnet die Koordinaten in sein intern verwendetes Format um.
- 3) Der SMS aktualisiert die Position des Objekts im räumlichen Modell.
- 4) Der Vorgang endet.

- **NACHBEDINGUNG:**

Die tatsächliche Position des mobilen Objekts entspricht der Position im räumlichen Modell.

5.2.2 Use Cases des Aktors raumbezogener Klient

Die Use Cases des raumbezogenen Klienten werden in der Reihenfolge von Abbildung 5.2 erklärt.

Use Case 2: Autorisiere mich

Sicherheitsaspekte spielen bei Ortsdiensten eine große Rolle (siehe Leonhardt, 1998). Daher besteht die Anforderung, dass die Klienten sich authentifizieren. Den Klienten können nach dem Feststellen ihrer Identität verschiedene Rechte zugewiesen werden.

- **STANDARDPFAD:**
 - 1) Der Klient überträgt seinen Bezeichner sowie seine aktuelle Position an den SMS.
 - 2) Der SMS speichert den Bezeichner und ermittelt die Rechte des Klienten. Die Rechte werden gespeichert.
 - 3) Der Klient wird an der übertragenen Position in das räumliche Modell eingetragen.
 - 4) Der Vorgang terminiert.
- **NACHBEDINGUNG:**

Der Klient ist autorisiert, bestimmte räumliche Informationen vom SMS zu erhalten bzw. zu manipulieren.
- **ANMERKUNG:**

Dieser Use Case wird im Rahmen dieser Arbeit nur konzeptionell angedacht.

Use Case 3: Stelle Gebiet dar

Damit sich Benutzer von raumbezogenen Klienten ein Bild eines räumlichen Gebiets machen können, soll der SMS in der Lage sein, sowohl zwei- als auch dreidimensionale Gebiete in Form von Bildern oder VRML-Modellen darzustellen (siehe Carey und Bill, 1997). Über Änderung der Gebietsgrenzen kann die Applikation zum Beispiel Zoomfunktionalität realisieren.

- **VORBEDINGUNG:**

Der raumbezogene Klient ist autorisiert, räumliche Informationen über das spezifizierte Gebiet zu erhalten.
- **STANDARDPFAD:**
 - 1) Der Klient spezifiziert das Gebiet durch die geometrischen Ausmaße, den Detaillierungsgrad und die Objektklassen, die zu präsentieren sind. Er gibt ebenfalls an, ob er ein Bild oder ein

VRML-Modell anfordert. Der Detaillierungsgrad gibt an, wieviele Details räumlicher Objekte dargestellt werden. Bei dreidimensionalen Gebieten wird auch ein Blickpunkt angegeben.

2) Der SMS erzeugt ein Bild in GIF- oder JPEG-Format bzw. ein VRML-Modell des spezifizierten Modells. Falls der Klient räumliche Objekte des spezifizierten Gebiets zuvor selektiert hat, so werden diese Objekte gesondert dargestellt.

3) Der SMS übermittelt dem Klienten das erzeugte Bild bzw. Modell.

5) Der Vorgang terminiert.

- NICHTFUNKTIONALE ANFORDERUNGEN:

Das Datenvolumen, dass zur Darstellung übertragen wird, soll minimal gehalten werden.

Use Case 4: Navigiere mich

Mobilen Objekten soll die Navigation zu Zielorten erleichtert werden. Daher besteht die Anforderung, dass der SMS optimale Pfade bestimmt und beschreibt.

- VORBEDINGUNG:

Der raumbezogene Klient ist autorisiert, räumliche Informationen zu erhalten.

- STANDARDPFAD:

1) Der Klient gibt einen Zielort und eventuell weitere Orte vor, die auf dem Weg zum Zielort besucht werden sollen. Er kann außerdem Zwangsbedingungen für den Weg vorgeben. Solche Bedingungen sind beispielsweise *Nur auf Gehwegen* oder *Nur auf Autobahnen*.

2) Der SMS bestimmt einen optimalen Pfad von der aktuellen Position des Klienten über die Zwischenorte zum Zielort unter Berücksichtigung der Zwangsbedingungen.

3) Er beschreibt den optimalen Pfad unter Angabe der aktuellen Himmelsrichtung oder als Polylinie in der grafischen Darstellung der Umgebung (Use Case 3: Stelle Gebiet dar). Die Angabe der Himmelsrichtung ist insbesondere im Wearable Computing-Umfeld nützlich.

4) Nach jeder Fortbewegung an einen neuen Ort wird eine aktualisierte Beschreibung des Pfades geliefert, d.h. entweder als neue Himmelsrichtung oder als geänderte Polylinie in der grafischen Darstellung.

5) Der Vorgang endet.

- ALTERNATIVER PFAD:

statt 3) und 4): Der SMS liefert einmalig eine textuelle Beschreibung des bestimmten Pfades.

Use Case 5: Analysiere Modell

Für Anwender ist es höchst hilfreich, die Umgebung nach vielfältigen Kriterien analysieren zu können. Auf diese Weise ist es ihnen zum Beispiel möglich, Parkhäuser zu lokalisieren, die an einer bestimmten Straße angrenzen.

- **VORBEDINGUNG:**
Der raumbezogene Klient ist autorisiert, räumliche Informationen zu erhalten.
- **STANDARDPFAD:**
 - 1) Der Klient fragt räumliche Objekte nach *geometrischen, thematischen* oder *topologischen* Gesichtspunkten ab (vgl. Kapitel 3.1.2). Verschiedene Bedingungen können sich dabei kombinieren lassen. Die einzelnen Dienste werden in Kapitel 6 bei der Definition der Schnittstelle festgelegt. Zu der Gruppe der Anfragen gehört auch die Suche nach einem bestimmten Objekt, die Anfrage nach Attributwerten eines Objekts sowie die Bestimmung des nächstgelegenen Objekts.
 - 2) Der SMS führt die Anfrage aus.
 - 3) Er sendet das Ergebnis der Anfrage zurück an den Klienten.
 - 4) Der Vorgang endet.
- **ALTERNATIVER PFAD:**
statt 3): Der Klient fordert ein Bild oder Modell an, in welches das Ergebnis der Anfrage grafisch eingezeichnet ist (Use Case 3).

Use Case 6: Generiere Zone

Um die räumlichen Objekte innerhalb bestimmter Bereiche um zuvor selektierte Objekte auswählen zu können, ist es vorteilhaft, Zonen zu bilden. Eine Zone setzt sich aus Gebieten um die Menge von zuvor selektierten Objekte zusammen.

- **VORBEDINGUNG:**
Der raumbezogene Klient ist autorisiert, räumliche Informationen zu erhalten.
- **STANDARDPFAD:**
 - 1) Der Klient spezifiziert die räumlichen Objekte, um welche die Zone gelegt wird (Use Case 5), sowie Zonenform und -größe.

2) Der SMS übermittelt dem Klienten die Bezeichner der räumlichen Objekte innerhalb der Zone.

3) Der Vorgang endet.

- ALTERNATIVER PFAD:

statt 2): Der Klient fordert ein Bild oder Modell an, in welches die selektierten räumlichen Objekte grafisch eingezeichnet sind (Use Case 3).

Use Case 7: Berechne Werte

Anwender sind daran interessiert, Abstände messen zu können, beispielsweise zwischen Orten. Daher soll der SMS imstande sein, Abstände zwischen Objekten und Anzahl bzw. Anteil bestimmter räumlicher Objekte zu berechnen.

- VORBEDINGUNG:

Der raumbezogene Klient ist autorisiert, räumliche Informationen zu erhalten.

- STANDARDPFAD:

1) Der Klient übermittelt Anfragen nach Abständen, sowie Anzahl oder Anteil räumlicher Objekte mit bestimmten Eigenschaften.

2) Der SMS berechnet die Werte und übermittelt sie dem Klienten.

3) Der Vorgang endet.

Use Case 8: Abonniere Ereignis

Falls der Raumbezogene Klient auf das Eintreten eines raumbezogenen Ereignisses wartet, kann er solange beim SMS anfragen, bis das Ereignis eintritt. Ökonomischer ist es jedoch, wenn sich Klienten für Ereignisse registrieren können und bei Eintritt des Ereignisses informiert werden.

Eintritt: Objekt mit bestimmten Eigenschaften betritt Bereich.
Austritt: Objekt mit bestimmten Eigenschaften verlässt Bereich.
Treffen: Ein gegebenes Objekt betritt den Umkreis eines Objekts mit bestimmten Eigenschaften.
Gruppentreffen: Mehrere Objekte mit bestimmten Eigenschaften kommen innerhalb eines spezifizierten Gebiets zusammen.

Tabelle 5.1: Raumbezogene Ereignisse des SMS

- **VORBEDINGUNG:**
Der raumbezogene Klient ist autorisiert, räumliche Informationen zu erhalten.
- **STANDARDPFAD:**
 - 1) Der Klient abonniert ein Ereignis. Die möglichen Ereignisse sind logische Ausdrücke der Basisereignisse aus Tabelle 5.1 (siehe Leonhardi und Kubach, 1999).
 - 2) Jedesmal wenn das abonnierte Ereignis eintritt, wird der Klient vom SMS darüber informiert.
 - 3) Der Eintritt des Ereignisses wird nicht mehr gemeldet, wenn der Klient das Abonnement beendet.
- **ALTERNATIVER PFAD:**
statt 3): Der Eintritt des Ereignisses wird für eine spezifizierte Anzahl oder Zeit gemeldet.

Use Case 9: Bearbeite Objekt

Eine Idee des Nexus-Ansatzes ist es, dass räumliche Objekte nicht nur präsentiert, sondern mittels der Nexus-Plattform auch manipuliert werden können.

- **VORBEDINGUNG:**
Der raumbezogene Klient ist autorisiert, räumliche Objekte zu manipulieren.
- **STANDARDPFAD:**
 - 1) Der raumbezogene Klient versucht, die Attribute eines räumlichen Objekts zu ändern.
 - 2) Der SMS prüft, ob der Klient autorisiert ist, das Objekt zu manipulieren.
 - 3) Falls die Prüfung positiv ausfällt, propagiert der SMS die Änderung an das Objekt. Die Änderungen an dem Objekt sind auch für andere Klienten sichtbar.
 - 4) Der Vorgang endet.
- **ALTERNATIVE PFADE:**
statt 1): Der raumbezogene Klient registriert ein mobiles Objekt beim SMS oder meldet es ab.
statt 1): Der raumbezogene Klient erzeugt oder löscht ein virtuelles Objekt. Beim Erzeugen eines Objekts müssen neben den Attributen auch die Klasse des Objekts angegeben werden. Zwischen Klassen räumlicher Objekte sollen Vererbungsbeziehungen möglich sein.

Use Case 10: Lade Modell

Falls der Klient Teilmodelle des räumlichen Modells selbst verarbeiten möchte — zur Leistungssteigerung beispielsweise —, so sollte es ihm möglich sein, streng abgegrenzte Teilmodelle vom SMS zu erhalten.

- **VORBEDINGUNG:**
Der raumbezogene Klient ist autorisiert, räumliche Informationen zu erhalten.
- **STANDARDPFAD:**
 - 1) Der raumbezogene Klient spezifiziert das Teilmodell durch den geometrischen Bereich und die gewünschten Thematiken.
 - 2) Der SMS sendet das Teilmodell zum Klienten.
 - 3) Der Vorgang endet.

Damit wird die Spezifikation der Use Cases abgeschlossen. Der nächste Abschnitt definiert weitere Anforderungen, die hauptsächlich nichtfunktionaler Natur sind und die sich nicht einzelnen Use Cases zuordnen lassen.

5.3 Ergänzende Anforderungen

Die ergänzenden Anforderungen an den SMS lassen sich in physische und nichtfunktionale untergliedern. Die folgenden Anforderungen sind durchnummeriert (*EA: Ergänzende Anforderung*), um sie leichter referenzieren zu können.

5.3.1 Physische Anforderungen

EA1: Plattformunabhängige Klienten. Klienten des SMS sollten auf beliebigen Plattformen aufsetzen können.

EA2: Ressourcenschwache Klienten. Da mobile Rechner tendenziell eher weniger leisten als stationäre Rechner (vgl. Kapitel 4.1), besteht die Anforderung, dass auch ressourcenschwache Klienten die Dienste des SMS nutzen können sollen. Trotzdem sind vollwertige PCs ebenfalls adäquat zu unterstützen.

EA3: Erforderliche Übertragungskapazität minimieren. Aufgrund der geringen Bandbreite drahtloser Kommunikation ist eine Minimierung der erforderlichen Übertragungskapazität erforderlich (vgl. Kapitel 4.2).

5.3.2 Entwurfsvorgaben

EA4: Skalierbarkeit. Die Nexus-Plattform soll in der Anzahl der Applikationen, im geografischen Bereich und in der Anzahl der räumlichen Objekte skalierbar sein (siehe Leonhardt, 1998).

EA5: Verteilung der Arbeitslast. Die Arbeitslast des SMS soll möglichst optimal zwischen Server-Rechner und Klient aufteilbar sein.

EA6: Verwaltung mehrerer räumlicher Modelle. Ein SMS kann mehrere räumliche Modelle verwalten.

EA7: Verwendung von Java für die Klassenbibliothek. Die Klassenbibliothek, welche Anwendungsentwicklern zur Verfügung gestellt wird, soll in der Programmiersprache Java implementiert sein.

5.4 Zusammenfassung der wichtigsten Anforderungen

Der SMS verwaltet räumliche Modelle, die virtuelle, mobile und statische Objekte umfassen und Modelle der realen Welt — ergänzt um virtuelle Objekte — darstellen. Die Objekte besitzen einen Zustand und lassen sich manipulieren. Ein räumliches Modell kann auf eine Vielzahl von Möglichkeiten analysiert sowie in ihm navigiert werden. Außerdem können sich Klienten für räumliche Ereignisse registrieren.

Der SMS unterstützt sowohl ressourcenschwache Klienten als auch Klienten, die rechenstark sind und über ein leistungsstarkes Netzwerk kommunizieren. Er ist skalierbar und erlaubt die Verteilung der Arbeitslast entsprechend dem Zustand des Klienten zwischen Server und Klient. Die Darstellung der räumlichen Information ist entweder grafisch oder textuell.

Das folgende Kapitel konkretisiert die in diesem Kapitel formulierten Anforderungen, indem es die Dienste der Schnittstelle festlegt, welche der SMS anbietet.

6 Definition der Schnittstelle

In diesem Kapitel wird die Schnittstelle des SMS definiert. Sie ist in mehrere Teile untergliedert, um der Tatsache Rechnung zu tragen, dass höchst unterschiedliche Klienten bedient werden und um sie übersichtlicher zu gestalten. Daher wird in diesem Kapitel zunächst die Struktur der Schnittstelle erläutert. Anschließend werden die einzelnen Teile der Schnittstelle vorgestellt. Den Abschluss dieses Kapitels bildet ein Vergleich mit der *Open GIS Service Architecture*, die Dienste von GIS zu normieren beabsichtigt.

Zu beachten ist, dass ein SMS mehrere räumliche Modelle verwalten kann. Die Dienste, die in diesem Kapitel vorgestellt werden, beziehen sich immer auf ein räumliches Modell.

6.1 Struktur der Schnittstelle

Die Schnittstelle lässt sich in vier Bereiche einteilen. Abbildung 6.1 veranschaulicht die Einteilung und gibt die intendierten Klienten an. Die *Basisschnittstelle* bietet Teilmodelle des räumlichen Modells in „Rohform“ an. Sie richtet sich an mächtige Klienten, die auf Notebooks aufsetzen. Diese Klienten sind fähig, Teilmodelle selbständig verarbeiten. Sie könnten lokal zum Beispiel einen SMS einsetzen.

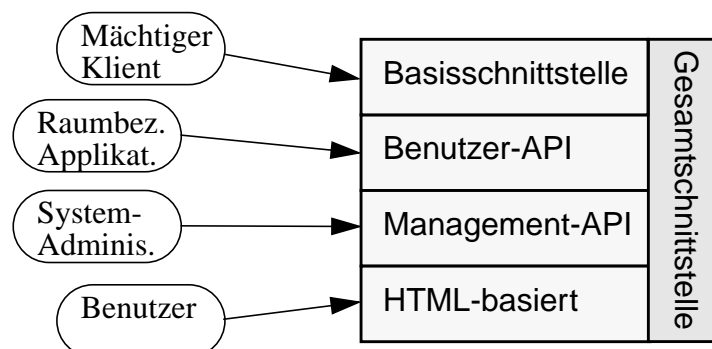


Abbildung 6.1: Struktur der Schnittstelle

Die *Benutzer-API* (Application Programming Interface) ist das Herz der Schnittstelle. Abgesehen vom Laden eines Teilmodells bietet sie die Dienste an, welche die Use Cases aus Kapitel 5 erfordern. In Form einer Bibliothek stellt sie Klassen bereit, die direkt die Anwendungsentwicklung unterstützen. Die Klassen sind in Java implementiert (vgl. Kapitel 5.3.2). Die Benutzer-API wird von raumbezogenen Applikationen eingebunden, die Notebooks genauso wie leistungsstarke PDAs als Plattform nutzen können.

Zur Verwaltung des räumlichen Modells eines SMS dient die *Management-API*. Sie bietet Dienste an, die jegliche räumliche Objekte inspizieren und manipulieren. Sie kann beispielsweise vom Anbieter eines Modells herangezogen werden, um Änderungen am Modell zu automatisieren. Die Management-API bietet Dienste an, die in Kapitel 5 nicht spezifiziert werden, da dort die Benutzersicht dargestellt wird.

Die *HTML-basierte* Schnittstelle zielt auf leistungsschwache Klienten wie einfache PDAs oder PICs ab. Sie bietet eine Schnittstelle, die einzig einen Webbrowser zur Nutzung des SMS erfordert. Benutzer verwenden diese Schnittstelle daher unmittelbar.

6.2 Elemente der Schnittstelle

Dieser Abschnitt stellt die soeben angesprochenen Elemente der Schnittstelle detaillierter vor. Er gibt repräsentative Operationen von wichtigen Funktionsklassen an. Die Ausarbeitung der kompletten Schnittstelle ist eine Aufgabe für die Zukunft. Die Dienste werden in Pseudocode definiert, der an der Javasyntax orientiert ist.

6.2.1 Basisschnittstelle

Die Basisschnittstelle kann nur von Klienten verwendet werden, die befähigt sind, räumliche Modelle des SMS selbständig zu verarbeiten. Sie bietet nur einen Dienst an. Dieser ist definiert wie in Tabelle 6.1 angegeben.

File getModel(area, objectClasses[])

Tabelle 6.1: Dienst der Basisschnittstelle

Der erste Parameter von `getModel` spezifiziert das räumliche Gebiet des Modells. Der zweite Parameter gibt die Objektklassen an, die verlangt werden. Vererbungsbeziehungen zwischen Klassen werden vom SMS dabei berücksichtigt. Der Dienst liefert eine Datei zurück, die das angeforderte räumliche Teilmodell enthält.

6.2.2 Benutzer-API

Die Benutzer-API stellt Dienste bereit, die die Umsetzung der Use Cases aus Kapitel 5 erlauben. Dabei wird angestrebt, eine minimale Menge von Diensten anzubieten, deren Kombination beliebige räumliche Operationen ermöglicht. Um die endgültigen Dienste festlegen zu können, müssen jedoch erst die Erfordernisse der Klienten untersucht werden.

Darüber hinaus wird Wert darauf gelegt, dass Anwendungsentwickler mit den Diensten der API vernünftig arbeiten können. Zu beachten ist, dass der SMS zustandsorientiert arbeitet und daher Dienste aufeinander aufbauen können.

Zur Gliederung werden die Dienste in Funktionsklassen gruppiert. Diese Klassen werden im Folgenden vorgestellt. Die exakte Syntax der Dienste der Benutzer-API ist in Anhang A.2 definiert.

Hinzufügen und Entfernen räumlicher Objekte

Diese Klasse enthält Dienste zum Erzeugen und Löschen virtueller Objekte sowie zum Registrieren und Abmelden mobiler Objekte. Statische Objekte dürfen über die Benutzer-API weder hinzugefügt noch entfernt werden. Eine kurze Übersicht über diese Klasse bietet Tabelle 6.2

Bei der Registrierung mobiler Objekte werden neben dem Bezeichner des Objekts auch der Location Service sowie der Home Server des Objekts spezifiziert. Durch den Location Service kann der SMS bei Bedarf die aktuelle Position des Objekts in Erfahrung bringen. Der *Home Server* speichert Attribute des mobilen Objekts, welche der SMS dort abfragen kann. Er wird ausführlicher in Kapitel 7 vorgestellt. Bei der Registrierung wird auch festgelegt, ob das mobile Objekt von anderen Nutzern des SMS wahrgenommen werden kann. Diese Maßnahme dient der Wahrung der Privatsphäre des Objekts.

```
void registerMobileObject(objectId, locServ, homeServ, isAccessible)
```

Tabelle 6.2: Kurzübersicht über die Dienste dieser Klasse

<code>void unregisterMobileObject(objectId)</code>
<code>ObjectId createVirtualObject(creator, objClass, attribute[], geometry)</code>
<code>void destroyVirtualObject(remover, objectId)</code>

Tabelle 6.2: Kurzübersicht über die Dienste dieser Klasse

Mobile Objekte werden unter Angabe ihrer Bezeichner abgemeldet. Die Erzeugung virtueller Objekte erfordert die Angabe des Erzeugers, der Klasse des Objekts, dessen sachlichen Attribute sowie dessen Geometrie. Der Erzeuger muss angegeben werden, um überprüfen zu können, ob er das spezifizierte Objekt in das räumliche Modell einbringen darf. Der Dienst liefert den neu erzeugten eindeutigen Bezeichner des Objekts zurück.

Zum Entfernen virtueller Objekte wird neben dem Bezeichner des Objekts auch dessen Entferner zur Rechteprüfung angegeben.

Räumliche Analyse

Diese Funktionsklasse bietet räumliche Analyseoperationen an. Die Operationen lassen sich kombinieren. Dadurch können auch komplexe Beziehungen ausgedrückt werden.

Zunächst werden Dienste vorgestellt, die Datenabfragen erlauben (vgl. Tabelle 6.3). `objectsInArea` bestimmt alle Objekte, die sich innerhalb eines spezifizierten Gebiets befinden. Dabei ist auch ein Inklusionstyp anzugeben. Der *Inklusionstyp* legt fest, ob Objekte in die Ergebnismenge aufgenommen werden, falls

- sie vollständig im spezifizierten Gebiet liegen,
- ihr Zentrum sich im Gebiet befindet oder
- sie das Gebiet beliebig schneiden.

`objectsInArea` erfordert außerdem die Angabe einer Integrationsstrategie. Die *Integrationsstrategie* legt fest, auf welche Weise das Ergebnis der aktuellen Operation mit vorherigen räumlichen Analyseanfragen kombiniert wird. Zu den *räumlichen Analyseanfragen* gehören neben `objectsInArea` noch `objectsSatCond`, `neighborsFrom`, `nearestObjects` und `objectsAroundSelected`. Die möglichen Integrationsstrategien dieser Operationen sind wie folgt:

- *NewSet* bildet eine neue Selektionsmenge.

- *SelectFromSet* wählt aus der Selektionsmenge, die die vorherigen Analyseanfragen bestimmt haben, die Objekte aus, welche auch die aktuelle Anfragebedingung erfüllen. Diese formen die neue Selektionsmenge.
- *AddToSet* fügt zur bestehenden Selektionsmenge diejenigen Objekte hinzu, welche die aktuelle Anfragebedingung erfüllen.

objectsSatCond (für *objects that satisfy a condition*) ermittelt die Objekte, die eine Bedingung auf den Sachdaten der Objekte erfüllen. Auch dieser Dienst erfordert die Angabe der Integrationsstrategie. *neighborsFrom* bestimmt die Objekte, die im topologischen Sinn Nachbarn eines vorgegeben Objekts sind. Seine Integrationsstrategie ist wie die von *nearestObjects* immer *SelectFromSet*. *nearestObjects* ermittelt die n nächstgelegenen Objekte in Bezug auf eine Referenzposition.

ObjectId[] objectsInArea(area, inclusionType, integrStrategie)
ObjectId[] objectsSatCond(attrCond, integrStrategie)
ObjectId[] neighborsFrom()
ObjectId[] nearestObjects(point, number)
Point locationOfObject(objectId)
Object geometryOfObject(objectId)
void setCoordSystem(coordSystem)
ObjectId objectInDirection(point, line)

Tabelle 6.3: Kurzübersicht über Dienste dieser Klasse

Zu den geometrischen Diensten gehört auch *locationOfObject*. Es bestimmt eine repräsentative Position räumlicher Objekte. Als repräsentative Position wird der Schwerpunkt der Geometrie des Objekts zurückgeliefert. Um die Geometrie des Objekts an sich zu erhalten, wird *geometryOfObject* angeboten. *setCoordSystem* hat eher administrativen Charakter. Es legt das Koordinatensystem fest, das innerhalb der Schnittstelle verwendet werden soll. Der Dienst *objectInDirection* bestimmt schließlich das Objekt, das von einer Geraden geschnitten wird und am nächsten zu einem vorgegeben Punkt liegt.

Aus den Bereichen Computed Geometry und Zonengenerierung stammen die Dienste aus Tabelle 6.4. `distanceTo` berechnet die Distanz zweier Punkte in Metern. `sizeOfObject` ermittelt die Größe eines räumlichen Objekts. Die Operation `objectsAroundSelected` bestimmt alle Objekte, welche sich innerhalb eines spezifizierten Abstands um zuvor selektierte Objekte befinden. Sie verwendet die Integrationsstrategie `SelectFromSet`. `quantityOfSelected` gibt schließlich die Anzahl der räumlichen Objekte an, die in der aktuellen Selektionsmenge enthalten sind.

<code>long distanceTo(pointA, pointB)</code>
<code>long sizeOfObject(objectId)</code>
<code>ObjectId[] objectsAroundSelected(distance)</code>
<code>int quantityOfSelected()</code>

Tabelle 6.4: Kurzübersicht über die weiteren Dienste dieser Klasse

Navigation

Die Navigation erfordert funktional nur einen Dienst. Da jedoch verschiedene Arten der Wegbeschreibung nötig sind, besteht diese Funktionsklasse aus mehreren Diensten (vgl. Kapitel 5).

Alle Dienste ermitteln den besten Pfad von einem Start- zu einem Zielpunkt, wobei dazwischen eine (möglicherweise leere) Menge von Punkten besucht wird (vgl. Tabelle 6.5). Wenn der Parameter `bestOrder` wahr ist, dann wird der kürzeste Pfad ohne Berücksichtigung der Reihenfolge der Punkte bestimmt. Ansonsten bleibt die Reihenfolge gewahrt. Mittels des Parameters `constraints` können Zwangsbedingungen angegeben werden, die der zu ermittelnde Pfad erfüllen muss; zum Beispiel: `interstateHighway` or `scenicDrive`.

Der gefundene Pfad wird in textueller Form als Wegbeschreibung, unter Angabe von Eckpunkten oder grafisch als Bild dargestellt. Ermittelte Eckpunkte eines Weges können verwendet werden, um die aktuelle Himmelsrichtung des Pfades zu bestimmen.

<code>String findPathExpl(points[], bestOrder, constraints)</code>
<code>Point[] findPathCorners(points[], bestOrder, constraints)</code>
<code>Image findPathImage(points[], bestOrder, constraints)</code>

Tabelle 6.5: Kurzübersicht über die Dienste der Klasse Navigation

Abfragen von Sachdaten

Aufgabe der Dienste aus Tabelle 6.6 ist die Bereitstellung von Sachdaten. `getAttribute` liefert ein Attribut eines räumlichen Objekts zurück. `attributeNames` gibt die Namen aller Attribute eines Objekts an. Damit lässt sich herausfinden, welche Attribute ein Objekt besitzt. Der Dienst `classOfObject` benennt die Klasse von Objekten. Auf diese Weise soll es möglich sein, auf mögliche semantische Unterschiede zwischen syntaktisch identischen Attributen aufmerksam gemacht zu werden. Erforderlich dabei ist, dass Klassen global eindeutige Namen tragen.

<code>Object getAttribute(objectId, attrName)</code>
<code>String[] attributeNames(objectId)</code>
<code>ObjClass classOfObject(objectId)</code>

Tabelle 6.6: Kurzübersicht über die Dienste der Klasse Sachdaten

Manipulation räumlicher Objekte

Von räumlichen Objekten können sowohl die Sachdaten als auch die Geometrie verändert werden. Da Benutzern der API nicht gestattet ist, die Geometrie statischer Objekte zu manipulieren, ist in dieser Klasse kein solcher Dienst enthalten (vgl. Tabelle 6.7).

`setLocation` und `setGeometry` setzen den Schwerpunkt bzw. die Geometrie virtueller und mobiler Objekte. `setAttribute` ändert den Wert von Sachdaten beliebiger räumlicher Objekte. Der Parameter `manipulator` gibt die Instanz an, welche die Operation veranlasst. Er wird zur Rechteprüfung herangezogen.

<code>void setLocation(objectId, point, manipulator)</code>
<code>void setGeometry(objectId, geometry, manipulator)</code>
<code>void setAttribute(objectId, attribute, manipulator)</code>

Tabelle 6.7: Kurzübersicht über die Dienste der Klasse Manipulation

Räumliche Ereignisse

Zwei Dienste genügen für das Anbieten räumlicher Ereignisse (vgl. Tabelle 6.8). `subscribeSpatialEvent` registriert einen Klienten für den Empfang räumlicher Ereignisse. `cancelEventSubscription` beendet das Abonnement eines Ereignisses. Die Ereignisse werden durch Angabe ihres Bezeichners spezifiziert (siehe Wünsche, 1999).

<code>void subscribeSpatialEvent(eventId)</code>
<code>void cancelEventSubscription(eventId)</code>

Tabelle 6.8: Kurzübersicht über die Ereignisse der Klasse Ereignisse

Grafische Repräsentationen

Die letzte Funktionsklasse der Benutzer-API kümmert sich um die grafische Darstellung räumlicher Modelle (vgl. Tabelle 6.9).

`currentView`, `current3DView` und `currentVRMLModel` liefern Ansichten eines Gebiets zurück und stellen die im Parameter `classNames` spezifizierten Objektklassen dar. Wenn der Parameter `zoomToSelected` der beiden erstgenannten Dienste den Wert wahr besitzt, so wird vom räumlichen Modell das kleinste Rechteck (bzw. der kleinste Quader) dargestellt, welches alle räumlichen Objekte umfasst, die zuvor von räumlichen Analyseanfragen selektiert wurden. Andernfalls stellen die Dienste das Gebiet dar, das durch `setMapExtent` definiert wurde. Dies trifft auch auf `currentVRMLModel` zu. Der Parameter `point` von `current3DView` bestimmt den Blickpunkt auf ein dreidimensionales Modell.

<code>Image currentView(classNames[], zoomToSelected)</code>
<code>Image current3DView(classNames[], zoomToSelected, point)</code>
<code>File currentVRMLModel(classNames[])</code>
<code>Point[] getMapExtent()</code>
<code>void setMapExtent(point[])</code>
<code>void setImageSize(width, height)</code>
<code>void setLevelOfDetail(number)</code>

Tabelle 6.9: Kurzübersicht über die Ereignisse der Klasse Grafik

`getMapExtent` gibt das aktuell dargestellte Gebiet an. Dieser Dienst ist insbesondere hilfreich, falls `zoomToSelected` wahr ist, und daher der Benutzer nicht die Ausmaße des Gebiets bestimmen kann. Die Größe der erzeugten Bilder legt `setImageSize` fest. Der Dienst `setLevelOfDetail` definiert den erwünschten Detaillierungsgrad.

6.2.3 Management-API

Wie bereits erwähnt stellt die Management-API Operationen zur Verfügung, die normalen Benutzern weder erlaubt noch zugänglich sind. Sie dienen der Verwaltung räumlicher Modelle durch den „Eigentümer“ oder den Verantwortlichen des Modells. Die genaue Syntax der Operationen der Management-API ist in Anhang A.3 definiert.

Die Tabelle 6.10 listet die Dienste der Management-API auf. Analog zu `createVirtualObject` funktionieren die Operationen zur Erzeugung und zum Entfernen statischer und mobiler Objekte. `defineVirtObjClass` definiert eine Klasse virtueller Objekte. Der Parameter spezifiziert die Attribute und die Eingliederung in die Klassenhierarchie. Räumliche Objekte verfügen über keine Methoden wie die Objekte der objektorientierten Programmierung (vgl. Kapitel 3.1). Die Dienste zum Setzen des Schwerpunkts und der Geometrie statischer Objekte arbeiten wie die entsprechenden Operationen mobiler und virtueller Objekte. Die Operation `createSpatialModel` fügt einem SMS ein neues räumliche Modell hinzu.

<code>ObjectId createStaticObject(objClass, attribute[], geometry)</code>
<code>void destroyStaticObject(objectId)</code>
<code>ObjectId createMobileObject(objClass, attribute[], geometry)</code>
<code>void destroyMobileObject(objectId)</code>
<code>void defineVirtObjClass(objClass)</code>
<code>void setStatObjLocation(objectId, point)</code>
<code>void setStatObjGeometry(objectId, geometry)</code>
<code>void createSpatialModel(name, area, objClass[])</code>

Tabelle 6.10: Kurzübersicht über die Dienste der Management-API

6.2.4 HTML-basierte Schnittstelle

Zielgruppe der HTML-basierten Schnittstelle sind PDAs und PICs. Diese Schnittstelle bietet ihre Dienste daher als „reine“ HTML-Seiten an, d.h. mobiler Code findet keine Verwendung. HTML lässt sich auf WML (*Wireless Markup Language*) reduzieren, das für den Einsatz mit PICs gedacht ist (siehe Wap Forum, 1999).

Optimal wäre es, wenn die HTML-basierte Schnittstelle genauso mächtig wie die Benutzer-API sein könnte. Da die Möglichkeiten von HTML aber auf das Aktivieren von Hyperlinks und die Eingabe von Daten über Forms eingeschränkt ist, lassen sich einige Dienste nicht realisieren. Zu beachten ist, dass die Koordinaten eines Mausklicks auf einem Grafik ausgelesen und an den Server übermittelt werden können. Die nicht — oder nur unzulänglich — realisierbaren Dienste sind wie folgt:

- Die Selektion beliebiger räumlicher Objekte innerhalb eines Gebiets funktioniert nicht, da die HTML-Mechanismen in Grafiken keine Auswahl von Rechtecken oder Kreisen erlauben.
- Abstände zwischen zwei Objekten können nicht ermittelt werden, da ein einzelner Mausklick bereits wieder eine neue HTML-Seite lädt.
- Räumliche Ereignisse können nicht verarbeitet werden, da die Initiative zum Laden neuer Seiten vom Benutzer ausgehen muss.

Mehr oder weniger elegant lassen sich die übrigen Dienste der Benutzer-API umsetzen. Typischerweise werden dabei die Argumente von Operationen durch Eingabe der Argumentwerte in Textfeldern oder durch Anklicken von Bildern spezifiziert.

Die Operation `nearestObjects` kann beispielsweise implementiert werden, indem zuerst die Anzahl der zu bestimmenden Objekte über ein Textfeld eingegeben wird. Anschließend legt ein Mausklick auf das Bild, welches das räumliche Modell darstellt, fest, für welchen Referenzpunkt die nächstgelegenen Objekte zu finden sind.

Die Vielfalt der Möglichkeiten der Benutzer-API kann jedoch nicht erreicht werden, da für viele Operationen einige Hyperlinks und Textfelder bereitgestellt werden müssen. Da die HTML-basierte Schnittstelle hauptsächlich Rechner mit einer kleinen Oberfläche anspricht, wird bei einer Großzahl angebotener Dienste eine praktische Arbeit mit der HTML-basierten Schnittstelle wohl unmöglich.

Die Präsentation der Schnittstelle endet hiermit. Nachfolgend findet ein Vergleich der Schnittstelle mit einer Normschnittstelle statt, hinter der eine ähnliche Intention steht. Dadurch soll demonstriert werden, dass die in dieser Schnittstelle angebotenen Dienste, soweit es Nexus betrifft, insofern vollständig sind als dass kein wichtiger Dienst fehlt.

6.3 Vergleich mit der Open GIS Service Architecture

Die Open GIS Service Architecture wurde vom *Open GIS Consortium* entwickelt (siehe Open GIS Consortium, 1998). Das Konsortium verfolgt das Ziel, Geo-Informationssysteme zu fördern sowie die Interoperabilität verschiedener Systeme zu garantieren. Zu diesem Zweck definiert es im *Open GIS Specification Model*, das allgemeine Anforderungen an GIS formuliert, die *Shared Domain Services*. Die Shared Domain Services bieten über standardisierte Schnittstellen raumbezogene Dienste an. Genauso wie die SMS-Schnittstelle kategorisieren auch die Shared Domain Services ihre Dienste. Dieser Abschnitt vergleicht beide Schnittstellen kurz.

Die Kategorie *Räumliches Modell* der Shared Domain Services speichert, löscht und ändert räumliche Objekte. Außerdem umfasst sie Operationen zur Suche nach Objekten, die Bedingungen über den Sachdaten oder geometrische Bedingungen erfüllen. Räumliche Ereignisse sieht die Kategorie auch vor. Die einzige Operation dieser Kategorie, welche die SMS-Schnittstelle nicht anbietet, ist die Selektion von elliptischen Gebieten.

Die Kategorie *Bildmanipulation* besteht aus Operationen zur Steuerung der Anzeige von Bildern. Konkret sind das Operationen zum Verschieben, Vergrößern, Verkleinern und zum Rotieren des Ausschnitts des räumlichen Modells, der dargestellt wird. Solche Dienste bietet die SMS-Schnittstelle explizit nicht an. Trotzdem sind obige Operationen möglich, da der angezeigte Bildaus-

schnitt durch die Operation `setMapExtent` der Funktionsklasse Grafische Repräsentation geändert werden kann. Dadurch lassen sich die Operationen der Kategorie Bildmanipulation simulieren.

Eine weitere Kategorie der Shared Domain Services wird *Räumliche Analyse* genannt. Sie enthält Dienste zur Zonenerzeugung, zur Klassifikation sowie Computed Geometry- und Navigationsoperationen. Die Zonenerzeugung und Computed Geometry wird beim SMS durch die Funktionsklasse Räumliche Analyse abgedeckt. Die Funktionsklasse Navigation kümmert sich um Operationen der Navigation. Die SMS-Schnittstelle bietet Klassifikationsdienste nicht an, die für Nexus jedoch nicht relevant sind. Die Klassifikationsdienste werden bei der Datenerfassung benötigt. Sie teilen Datensätze in verschiedene funktionale Klassen ein.

Die Kategorie *Koordinatentransformation* setzt sich aus Diensten zur Transformation zwischen verschiedenen Koordinatensystemen zusammen. Der SMS bietet keine solchen Dienste explizit an. Vielmehr besteht die Intention, dass Anwendungsentwickler vom intern verwendeten Koordinatensystem des SMS abgeschirmt werden. Durch Angabe ihres verwendeten Koordinatensystems beim SMS erhalten sie immer Koordinaten in diesem System.

Einige Kategorien der Shared Domain Services werden von der SMS-Schnittstelle nicht berücksichtigt. Dazu gehören unter anderem das *Symbolmanagement*, *mathematische Analysen räumlicher Modelle* und die *photogrammetrische Analyse*. All diesen Kategorien ist gemein, dass sie im Rahmen der hier untersuchten SMS-Schnittstelle irrelevant sind, da deren Dienste von Nexus-Klienten nicht benötigt werden.

Zusammenfassend lässt sich sagen, dass die Schnittstelle der Shared Domain Services in jenen Punkten mit der SMS-Schnittstelle übereinstimmt, die für den SMS wesentlich sind. Die Shared Domain Services verfügen zusätzlich über Dienste, die der SMS nicht erfordert wie beispielsweise mathematische Analysen (vgl. Kapitel 5). In Bereichen, die den SMS betreffen, bestätigen die Shared Domain Services im Großen und Ganzen die SMS-Schnittstelle.

Damit wird die Beschreibung der Schnittstelle des SMS abgeschlossen. Die nächsten drei Kapitel handeln von der Entwicklung eines Systems, das die Dienste dieser Schnittstelle anbietet und die Anforderungen aus Kapitel 5 erfüllt.

7 Konzeptionelle Architektur

Die Zielsetzung dieses Kapitels liegt im Entwurf einer Architektur für den SMS, die unabhängig vom zugrunde liegenden GIS ist. Es sollen prinzipielle Überlegungen angestellt werden, bei denen Restriktionen und Nebenbedingungen, die sich aus dem Einsatz proprietärer GIS ergeben, nicht zu berücksichtigen sind. Kapitel 8 stellt dann die Architektur des SMS vor, welche auf einem ausgewählten GIS aufsetzt und im Rahmen dieser Arbeit implementiert wird.

Zunächst wird die Architektur des SMS erläutert. Dabei wird beachtet, dass in der Nexus-Plattform mehrere Instanzen des SMS enthalten sind. Die Integration des SMS in die Gesamtarchitektur von Nexus wird anschließend angesprochen. Das Augenmerk wird daraufhin auf die Föderation von Diensten gelegt. Den Abschluss dieses Kapitels bildet die Untersuchung geeigneter Kommunikationsinfrastrukturen.

7.1 Architektur des SMS

Dieser Abschnitt präsentiert zunächst die Komponenten eines SMS. Anschließend wird das Konzept des Home Servers dargelegt und erklärt, auf welche Weise die Arbeitslast verteilt wird.

Komponenten des SMS

Die Architektur eines SMS besteht aus den Komponenten, die in Abbildung 7.1 dargestellt sind. Der *Spatial Object Manager* speichert und verwaltet räumliche Objekte. Damit ist er verantwortlich für die Dienste der Funktionsklassen *Hinzufügen und Entfernen räumlicher Objekte*, *Abfragen von Sachdaten* und *Manipulation räumlicher Objekte* der Benutzer-API (vgl. Kapitel 6). Er realisiert darüber hinaus die objektbezogenen Dienste der Management-API. Außerdem stellt er räumliche Teilmodelle für die Basisschnittstelle bereit. Intern ist es sinnvoll, wenn der Spatial Object Manager eine Datenbank verwendet, um die Vorteile der Datenbanktechnologie wie bessere Performanz und Sicherstellung der Datenintegrität auszunutzen

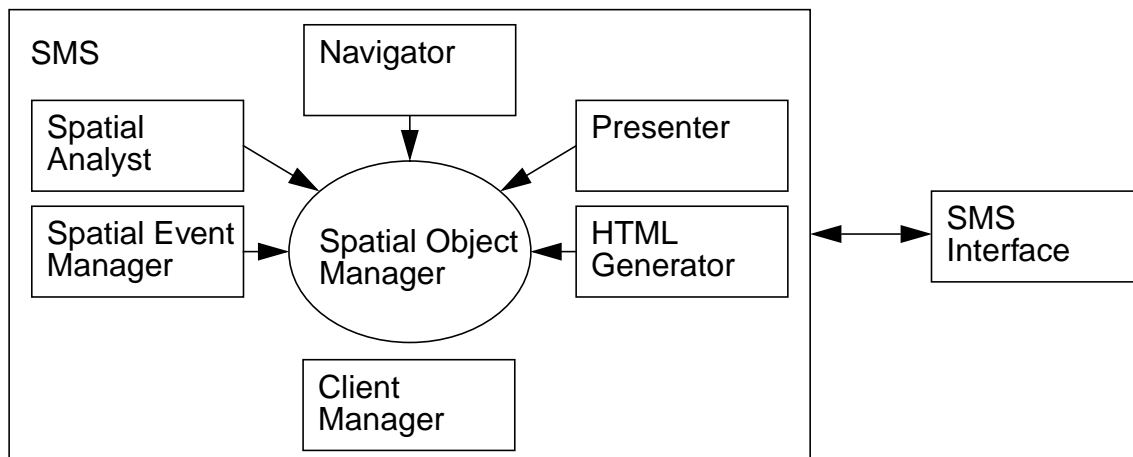


Abbildung 7.1: Komponenten des SMS

Um räumliche Ereignisse kümmert sich der *Spatial Event Manager*. Der *Navigator* realisiert die Dienste der Funktionsklasse *Navigation* der Benutzer-API. Die Klasse *Räumliche Analyse* ist der Bereich der Komponente *Spatial Analyst*, die auch Koordinatentransformationen vornimmt. Der *Presenter* stellt räumliche Modelle grafisch dar. Er bietet die Dienste der Funktionsklasse *Grafische Präsentation* an. Der *HTML Generator* erzeugt HTML-Seiten für die HTML-basierte Schnittstelle.

Es ist vorteilhaft, wenn sich der SMS den Zustand seiner Klienten merkt. Zum Einen kann damit das erforderliche Übertragungsvolumen verkleinert werden, da Daten nicht wiederholt übertragen werden müssen. Darüber hinaus wird verhindert, dass der SMS überflüssige Arbeit verrichtet. Wenn beispielsweise auf eine zuvor spezifizierte räumliche Analyseanfrage aufgebaut werden soll, dann kann der SMS einfach die bestehende Selektionsmenge des Klienten weiter verarbeiten. Falls der SMS zustandslos arbeitet, dann müssen alle Operationen, die einen Einfluss auf die aktuelle Anfrage haben, nochmals ausgeführt werden. Der *Client Manager* übernimmt die Aufgabe, den Zustand der Klienten zu verwalten. Zum Zustand gehören unter anderem die Ergebnisse vorheriger Anfragen, die gewünschte Größe von Bildern, die der Presenter erzeugt und der Typ des Klienten. Letzterer ist besonders wichtig für die adäquate Unterstützung mobiler Rechner.

SMS Interface ist die Bibliothek von Java-Klassen, die von raumbezogenen Applikationen eingebunden wird, um Zugang zum SMS zu erhalten.

Home Server

Der Spatial Object Manager fungiert auch als *Home Server* für mobile Objekte. Der Home Server ist eine zentrale Stelle, wenn ein mobiles Objekt bei mehreren SMS angemeldet ist. Er speichert persistent die Attribute mobiler Objekte, die ihn als Home Server haben, und stellt die Attribute anderen SMS-Instanzen zur Verfügung. Auf diese Weise braucht ein mobiles Objekt nicht seine gesamten Attribute mit sich führen. Sobald es bei einem SMS registriert wird, können seine Attribute beim Home Server des Objekts besorgt werden.

Ein zweiter Grund, der die Verwendung eines Home Servers nötig macht, ist die Verteilung von Diensten auf mehrere SMS. Falls die Föderation auf ein mobiles Objekte zugreifen möchte — um beispielsweise seinen Abstand zu einem anderen Objekt zu ermitteln —, dann muss es jenen SMS in Erfahrung bringen, der zur Zeit das mobile Objekt verwaltet. Um dies zu ermöglichen, ist im Bezeichner eines mobilen Objekts der Home Server angegeben. Außerdem teilt der SMS bei der Registrierung eines mobilen Objekts dessen Home Server mit, dass das Objekt gerade bei ihm weilt. Dadurch weiß der Home Server zu jeder Zeit, unter welcher Obhut „seine“ mobilen Objekte zur Zeit sind. Diese Information teilt der Home Server der Föderation dann auf Anfrage mit. Das Verfahren ist vom Prinzip her ähnlich dem Routingverfahren von Mobile-IP (siehe Tanenbaum, 1996).

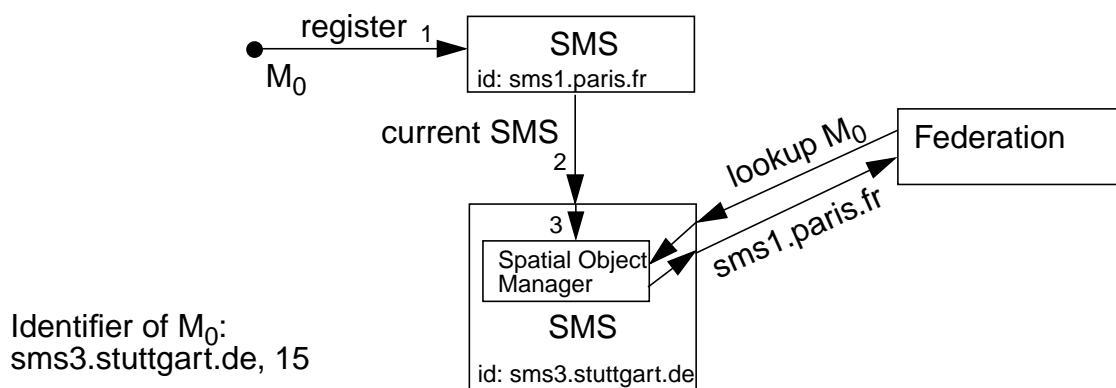


Abbildung 7.2: Arbeitsweise der Home Servers

Im Beispiel in Abbildung 7.2 registriert sich das mobile Objekt M_0 bei einem SMS (in der Abbildung dargestellt durch Pfeil 1). Dieser meldet dem Home Server des Objekts, dass es sich bei ihm aufhält (Pfeil 2). Der Spatial Object Manager speichert dies (Pfeil 3).

Wenn die Attribute eines mobilen Objekts von dessen Home Server kopiert und lokal gespeichert werden, stellt sich die Frage, wie die Konsistenz der Attribute gewahrt werden kann. Bei einer einfachen Realisierung sind die Attribute eines mobilen Objekts jedoch höchstens einmal beim aktuell verantwortlichen SMS repliziert. In diesem Fall kann durch Verwendung des *Lies eine/Schreib alle-Verfahrens* das Problem auf einfache Art gelöst werden (siehe Rothermel, 1997). Es sind jedoch diverse Optimierungen wie Replikationsverfahren möglich.

Wenn Attribute sich ändern, was meist wohl nur selten der Fall ist, müssen bei dem Lies Eine/Schreib alle-Verfahren sowohl das Original beim Home Server als auch die Kopie aktualisiert werden. Dafür sind Leseoperationen, die beim SMS viel häufiger vorkommen als Schreiboperationen, kostengünstig, da es ausreicht, die lokale Kopie zu lesen.

Verteilung der Arbeitslast

Eine Anforderung an den SMS besteht in der möglichen Verteilung der Arbeitslast zwischen Server und Klient (vgl. Kapitel 5.3). Ein einfacher Mechanismus erlaubt dies.

Klienten steuern die Verteilung der Arbeit, indem sie die entsprechenden Teile der SMS-Schnittstelle verwenden. Wählt er zum Beispiel die Basisschnittstelle, so wird auf dem Klient ein Großteil der Arbeit übernommen. Der Server stellt nur die Daten bereit, die der Klient verarbeitet. Bei Verwendung der Benutzer-API wandert die Arbeitslast zum Server. Der Klient führt nur noch vorbereitende Maßnahmen durch wie zum Beispiel das Zwischenspeichern von Anfragen. Die HTML-basierte Schnittstelle schließlich belastet den Klienten minimal.

7.2 Integration in Gesamtarchitektur

In der Einleitung wurden bereits die einzelnen Komponenten der Nexus-Plattform vorgestellt. Es ist beabsichtigt, diese Komponenten auf verschiedene Rechner zu verteilen, um Skalierbarkeit zu gewährleisten. Das Zusammenspiel zwischen SMS und den anderen Komponenten wird nachfolgend kurz erläutert (vgl. Abb. 7.3). Sowohl die Klienten als auch der Information Service werden hier nicht betrachtet, da sie mit dem SMS nicht direkt interagieren.

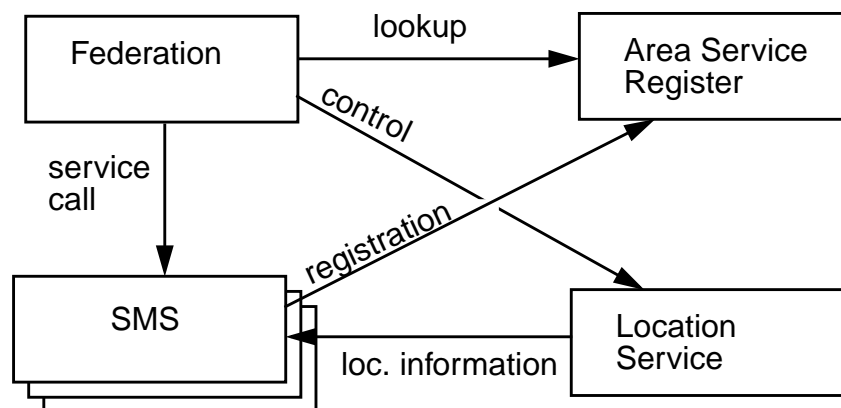


Abbildung 7.3: Integration des SMS in Nexus-Plattform

Wie schon erwähnt hat die Föderation die Aufgabe, eine einheitliche Schnittstelle nach außen anzubieten und angeforderte Operationen auf diejenigen Nexus-Komponenten zu verteilen, welche die erforderlichen Informationen besitzen. Um heraus zu finden, welcher SMS ein bestimmtes räumliches Modell anbietet, inspiziert die Föderation das Area Service Register. Das Area Service Register verwaltet die Zuordnung von SMS zu räumlichen Modellen. Daher registriert sich jeder SMS beim Area Service Register und gibt an, welches räumliche Modell er verwaltet.

Falls ein einzelner SMS das räumliche Modell umfasst, das zur Beantwortung einer Anfrage notwendig ist, so leitet die Föderation die Anfrage an den SMS weiter und propagiert dessen Ergebnis zum Klienten. Falls jedoch das notwendige Modell auf mehrere SMS verteilt ist, so muss eine Föderation der Anfrage durchgeführt werden. Das bedeutet, dass die Föderation an alle SMS, welche ein Teil des erforderlichen räumlichen Modells verwalten, geeignete Anfragen stellt. Anschließend integriert sie die Ergebnisse der SMS zum Gesamtergebnis der Anfrage oder führt selber Verarbeitungsschritte durch. Im nächsten Abschnitt werden die Mechanismen der Föderation näher untersucht.

Die Föderation steuert den Location Service. Sie spezifiziert den Dienst, der benötigt wird. Ferner bestimmt sie, an welchen SMS die räumlichen Informationen zu liefern sind. Der SMS nimmt diese Informationen vom Location Service entgegen und aktualisiert das entsprechende räumliche Modell.

7.3 Föderation

Die Funktionsweise der Föderationskomponente soll zunächst beispielhaft demonstriert werden (vgl. Abb. 7.4). Die Annahme ist, dass ein Klient alle räumlichen Objekte innerhalb des Gebiets *a* in Erfahrung bringen möchte (in der Abbildung dargestellt durch Pfeil 1). Da die Menge der Objektklassen nicht eingeschränkt ist, wird das Area Service Register nach allen SMS gefragt, die dieses Gebiet zumindest teilweise abdecken (Pfeil 2). Das Area Service Register liefert die SMS mit den Bezeichnern 1 und 3 zurück (Pfeil 3). Die Föderation findet anschließend heraus, dass mehr als ein SMS zur Verarbeitung der Anfrage notwendig ist. Daher bestimmt sie, welche Anfragen sie an die beteiligten SMS stellen muss, damit diese die richtigen Teilergebnisse liefern. Im Beispiel kann die Originalanfrage einfach weitergeleitet werden, da diese den gewünschten Effekt besitzt (Pfeil 4). Die Ergebnisse der SMS 1 und 3 (Pfeil 5) müssen von der Föderation nur noch konkateniert und an den aufrufenden Klienten zurückgeliefert werden (Pfeil 6). Die Integration der Teilergebnisse ist für andere Operationen, wie das Bestimmen optimaler Pfade, wesentlich komplexer.

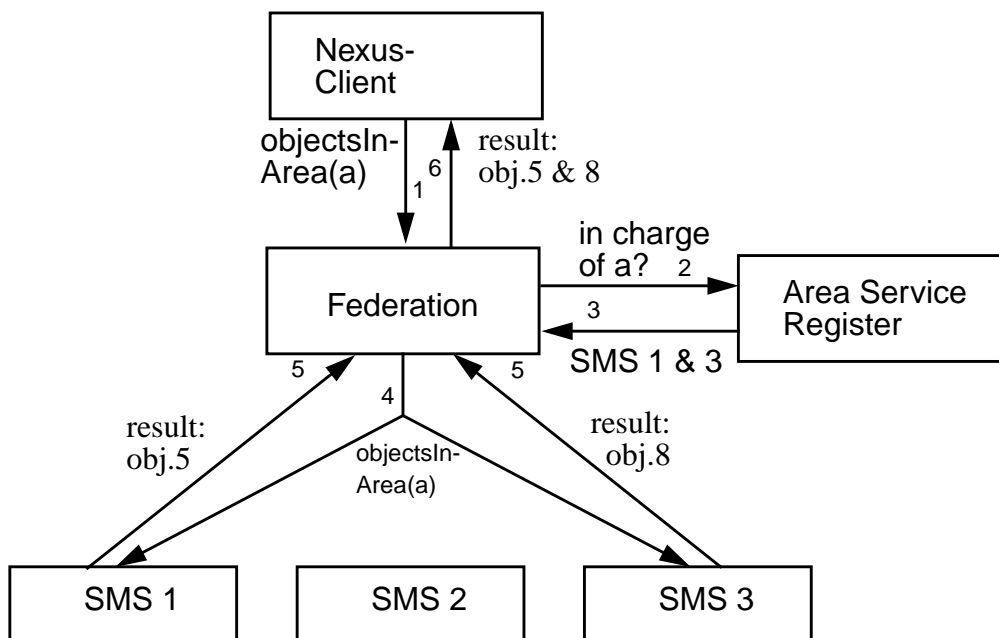


Abbildung 7.4: Ablauf der Föderation

Im Zusammenhang mit der Föderation werden anschließend drei Punkte betrachtet. Die Einbettung der Föderation in die Architektur wird als Erstes angesprochen. Zweitens wird untersucht, welche Charakteristika räumlicher Modelle SMS dem Area Service Register zur Spezifikation der Modelle angeben müssen. Die föderierte Realisierung der Dienste aus Kapitel 6, dabei insbesondere die Integration von Teilergebnissen, wird drittens erörtert.

7.3.1 Einbettung der Föderation

Die Föderation ist innerhalb der Nexus-Plattform eine eigenständige Komponente (vgl. Abb. 7.3). Es bestehen jedoch Möglichkeiten, sie auf andere Art in die Plattform einzubetten. Der Beweggrund, sie als eigenständige Komponente anzusehen, wird in diesem Abschnitt erklärt.

Prinzipiell kann die Föderation

- beim Klienten stattfinden,
- in jedem SMS enthalten sein oder
- in die Nexus-Plattform integriert werden.

Die Entscheidung, die Föderation beim Klienten durchzuführen, ist nicht empfehlenswert. Die Föderation stellt eine weitere Belastung für Klienten dar. Diese Lösung setzt weiter voraus, dass Klienten in der Lage sind, Teilergebnisse zusammen führen zu können. Speziell für Klienten, die die HTML-basierte Schnittstelle verwenden, ist dies nicht praktikabel. Außerdem lässt sich das Informationsvolumen, das übertragen werden muss, verkleinern, wenn die Föderation schon beim Server stattfindet und aus den Teilergebnissen die tatsächlich notwendigen Bestandteile selektiert werden. Der Klient müsste außerdem mit dem Area Service Register und jedem involvierten SMS kommunizieren und so drahtlose Kommunikation, die ja mit einigen Nachteile behaftet ist (vgl. Kapitel 4.2), ausgiebiger einsetzen. Aus diesen Gründen sollte die Föderation serverseitig stattfinden.

Wenn die Föderation in jeden SMS integriert ist, dann können Anfragen an einen beliebigen SMS gerichtet werden. Sie werden von dem angesprochenen SMS dann an die erforderlichen SMS weitergeleitet. Damit wäre die Föderation inhärent verteilt. Nachteilig bei diesem Ansatz ist, dass die Föderation in jedem SMS enthalten sein muss.

Bei einer eigenständigen Föderationskomponente findet eine Trennung von SMS und Föderation statt. Diese Lösung ist daher die flexibelste. Es sind damit auch Föderationskomponenten denkbar, die in keinem direkten Zusammenhang mit einem bestimmten SMS stehen. Letztere können außerdem kleiner sein und erfordern weniger Rechenleistung. Bei einer eigenen Föderationskomponente ergeben sich auch Vorteile bei der Skalierbarkeit. Falls eine Vielzahl von Applikationen auf die Nexus-Plattform zugreifen, kann es ausreichen, die Föderationskomponente zu replizieren. Im Gegensatz dazu muss ein vollständiger SMS repliziert werden, wenn die Föderationskomponente in ihn integriert ist. Da dann mehrere SMS dieselben räumlichen Daten verwalten, ergeben sich auch Konsistenzprobleme. Bei der Verwendung von DBMS sollten diese aber recht einfach lösbar sein.

Die in Kapitel 5.2 geforderte Skalierbarkeit der Nexus-Plattform ist mit der letzten Lösung möglich. Falls ein größeres räumliches Gebiet abgedeckt werden soll, können einfach neue SMS in die Plattform eingebracht und das Gebiet in kleinere Bereiche aufgeteilt werden. Wenn sich innerhalb eines räumlichen Modells zu viele Objekte befinden, kann das Modell aufgeteilt werden. Eine andere Möglichkeit besteht in der Replikation von SMS. Zur Konsistenzsicherung der raumbezogenen Daten sollten dann DBMS verwendet werden. Zur Unterstützung vieler Klienten kann die Föderationskomponente repliziert werden.

7.3.2 Räumliches Modell

Das Area Service Register ordnet einem vorgegebenen räumlichen Modell diejenigen SMS zu, welche Teile des Modells verwalten und die gemeinsam das Modell beinhalten. Damit es dazu in der Lage ist, müssen die Parameter festgelegt werden, die räumliche Modelle spezifizieren.

Die folgenden Parameter werden vom Klienten vorgegeben, um das gewünschte räumliche Modell zu spezifizieren:

- die Geometrie des abgedeckten Gebiets der realen Welt
- der erforderliche Detaillierungsgrad
- die gewünschten Objektklassen

Damit das Area Service Register ermitteln kann, welche SMS vorgegebene räumliche Modelle abdecken, benötigt er von den SMS weitere Informationen. Die Parameter, welche SMS zur Definition von räumlichen Modellen beim Area Service Register verwenden, sind wie folgt:

- die Geometrie des abgedeckten Gebiets der realen Welt
- die verfügbaren Detaillierungsgrade
- die verfügbaren Objektklassen
- die allgemeinen Zugriffsrechte auf das Modell
- die Beziehungsverhältnisse zu anderen Modellen

Auf die Zugriffsrechte wird im Rahmen dieser Arbeit nicht eingegangen (vgl. Kapitel 5). *Beziehungsverhältnisse* zu anderen Modellen werden angegeben, da sie dem Area Service Register mitteilen, auf welche Art Modelle miteinander kombinierbar sind. Mit dieser Information kann das Area Service Register seine Arbeit optimieren. Räumliche Modelle, die in Bezug auf eine spezielle Anfrage in einem inkompatiblen Beziehungsverhältnis stehen, brauchen nicht näher betrachtet werden. Mögliche Beziehungsverhältnisse werden im Anschluss näher untersucht.

Beziehungsverhältnisse räumlicher Modelle

Die Beziehungen von Modellen lassen sich untergliedern nach räumlichen und inhaltlichen Gesichtspunkten. Nachfolgend werden die einzelnen Beziehungsverhältnisse der beiden Typen vorgestellt. Daraufhin wird geprüft, welche Verhältnisse der unterschiedlichen Typen gleichzeitig vorliegen können.

Räumlich können zwei Modelle *sich überlappen*, *aneinander grenzen* oder *entfernt liegen* (vgl. Abb. 7.5). Zwei räumliche Modelle überlappen sich, falls sie zumindest teilweise das gleiche räumliche Gebiet abdecken. Ein Spezialfall der Überlappung ist das räumliche Enthalten-Sein. Ein Modell ist in einem anderen *enthalten*, falls es räumlich vollkommen von diesem umgeben ist. Falls die Modelle sogar das exakt gleiche Gebiet abdecken, sind sie räumlich *identisch*.

Zwei Modelle grenzen aneinander, wenn sie eine gemeinsame Grenze besitzen und sich nicht überlappen. Die Grenze kann aus einer Linie oder auch aus einem Punkt bestehen. Modelle liegen voneinander entfernt, falls es keinen gemeinsamen Punkt gibt.

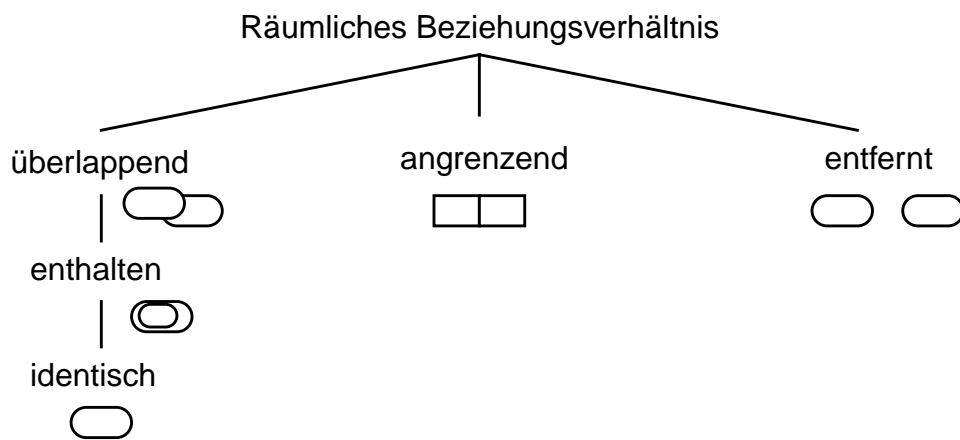


Abbildung 7.5: Räumliche Beziehungsverhältnisse von Modellen

Inhaltliche Beziehungsverhältnisse sind die *Verfeinerung*, die *attributive Ergänzung*, die *partielle Identität* sowie die *Unabhängigkeit* (vgl. Abb. 7.6). Ein Modell verfeinert ein anderes Modell, falls es räumliche Informationen mit einem höheren Detaillierungsgrad darstellt. Wenn in einem Modell von einem Kaufhaus beispielsweise nur die Außenfront dargestellt wird, so enthält ein verfeinerndes Modell etwa den Aufbau und die Innenarchitektur des Gebäudes. Bei der Verfeinerung wird der innere Aufbau von Objekten also detaillierter beschrieben. Objekten des größeren Modells entsprechen mehrere genauere Objekte des verfeinernden Modells.

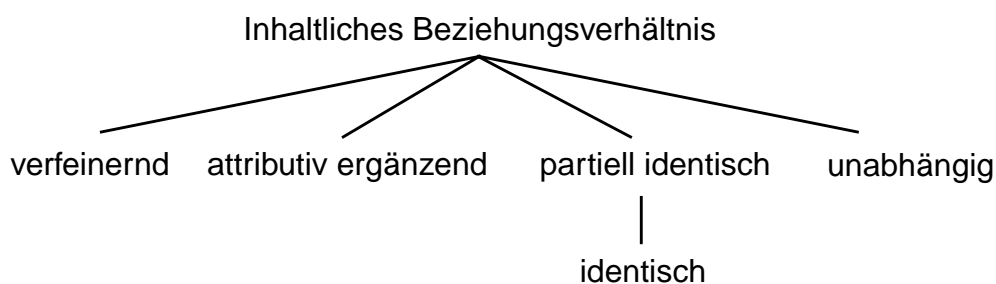


Abbildung 7.6: Inhaltliche Beziehungsverhältnisse von Modellen

Ein Modell ergänzt ein anderes attributiv, wenn es zu räumlichen Objekten des Modells neue Attribute hinzufügt. Die Geometrie der Objekte bleibt jedoch unverändert. Dieses Verhältnis liegt zum Beispiel vor, wenn die Betreiber von Supermärkten einem Stadtmodell Informationen zu den Objekten hinzufügen, die ihre Supermärkte repräsentieren. Die Stadtverwaltung möchte diese Informationen wahrscheinlich nicht in ihrem Modell enthalten haben.

Zwei Modelle sind partiell identisch, wenn sie mindestens eine identische Objektklasse besitzen. Falls sie über die selben Objektklassen verfügen, sind sie *identisch*. Sowohl die partielle Identität als auch die Identität sind hilfreich bei der Bestimmung der Modelle, die ein bestimmtes räumliches Modell fortsetzen.

Modelle sind schließlich unabhängig im Fall, dass jegliche Klassen unterschiedlich sind und keines der übrigen inhaltlichen Beziehungsverhältnisse vorliegt. Falls Modelle nicht unabhängig voneinander sind, so können mehrere inhaltliche Beziehungsverhältnisse vorliegen. Beispielsweise kann ein Modell ein anderes sowohl verfeinern als auch attributiv ergänzen.

Wenn die Beziehung zweier Modelle untersucht wird, müssen beide Beziehungstypen berücksichtigt werden. Manche Kombinationen der räumlichen und inhaltlichen Beziehungen sind jedoch nicht möglich. Die Tabelle 7.1 gibt an, welche Beziehungsverhältnisse zueinander passen.

Bei der Überlappung sind jegliche inhaltlichen Verhältnisse möglich. Wenn Modelle aneinander grenzen, sind sie entweder partiell identisch oder unabhängig. Das gleiche trifft zu, falls Modelle sich entfernt voneinander befinden. Ein Modell kann nicht von einem anderen Modell verfeinert oder ergänzt werden, wenn es sich mit diesem überhaupt nicht überschneidet.

Räumliches Verhältnis	Passende Inhaltliche Verhältnisse
überlappend	verfeinernd, attributiv ergänzend, partiell identisch, unabhängig
angrenzend	partiell identisch, unabhängig
entfernt	partiell identisch, unabhängig

Tabelle 7.1: Kombinierbare Beziehungsverhältnisse

Wenn zum Beispiel der kürzeste Pfad zu bestimmen ist, brauchen Modelle nicht berücksichtigt werden, die in Bezug auf ein vorgegebenes Modell unabhängig voneinander oder überlappend und sich gleichzeitig attributiv ergänzen. Damit wird die Menge der zu untersuchenden Modelle eingeschränkt.

7.3.3 Föderierte Realisierung der Dienste

Nicht alle Operationen, die in Kapitel 6 eingeführt wurden, benötigen einen Föderationsdienst. Das Registrieren eines mobilen Objekts oder das Berechnen der Größe eines räumlichen Objekts erfordern beispielsweise keinen Föderationsdienst. Für sie ist es im Fall eines mobilen Objekts nur nötig, den Home Server des Objekts zu konsultieren (vgl. Kapitel 7.1). Auf Dienste ohne Föderationsbedarf wird hier nicht eingegangen.

Die Tabelle 7.2 stellt die Operationen dar, welche eine Föderation benötigen. Die Parametertypen sowie der Ergebnistyp sind zur Veranschaulichung ebenfalls angegeben. Das generelle Vorgehen der Föderation bei der Ermittlung des Ergebnisses wird für jeden Dienst des Weiteren genannt. Nachfolgend wird beschrieben, was die Föderationskomponente leisten muss, um die Dienste aus Tabelle 7.2 realisieren zu können, ohne auf Optimierungsmöglichkeiten einzugehen.

Dienst	Parameter	Ergebnis	Handlung
objectsInArea	Area, int	ObjectId[]	Vereinigung
objectsSatCond	AttrCond, int	ObjectId[]	Vereinigung
neighborsFrom	ObjectId[]	ObjectId[]	Vereinigung
objectsAroundSelected	long	ObjectId[]	Vereinigung
nearestObjects	Point	ObjectId[]	Rekursion
objectInDirection	Point, Line	ObjectId	Vereinig.+Vergleich
quantityOfSelected	-	long	Addition
distanceTo	Point, Point	long	distance
currentView	String, boolean, AttrCond	Image oder File	Vereinigung
getModel	Area, ObjClass[]	File	Vereinigung
findPath	Point[], boolean, AttrCond	Image, String oder Point[]	findPath

Tabelle 7.2: Dienste, die eine Föderation erfordern

Die ersten fünf Dienste sind die räumlichen Analyseanfragen. Bis auf `nearestObjects` sind sie recht einfach föderiert umzusetzen. Die Föderation hat einzig die Aufgabe, die Spezifikation der Anfrage an alle beteiligten SMS zu verteilen und die zurückgelieferten Bezeichner der räumlichen

Objekte zusammenzufügen. Bei `nearestObjects` bestimmt die Föderation aus den erhaltenen Teilergebnissen rekursiv die nächstgelegenen. Dazu ist erforderlich, dass sie von allen erhaltenen Objekten die Position in Erfahrung bringt.

Die Operationen `nearestObjects` und `objectsSatCond` sollten räumlich eingeschränkt werden. Ansonsten ist es möglich, dass das globale räumliche Modell abgesucht wird.

Aus dem gleichen Grund ist die Operation `objectInDirection` räumlich einzuschränken. Zur Realisierung leitet die Föderation die Operation an jeden SMS weiter, der ein Modell im eingeschränkten Bereich besitzt. Anschließend ermittelt die Föderation das Objekt, welches am nächsten zum angegebenen Referenzpunkt liegt.

Die Operation `quantityOfSelected` kann durch eine einfache Addition der gelieferten Werte implementiert werden.

Die Operation `distanceTo` erfordert, dass die Föderation fähig ist, Abstände in den Koordinatensystemen der SMS zu berechnen. Die Koordinaten der beteiligten Objekte erfährt sie durch `locationOfObject`-Anfragen an den SMS, der aus dem Bezeichner des Objekts entweder direkt oder über den Home Server hervorgeht.

`currentView` in Tabelle 7.2 repräsentiert die SMS-Operationen `currentView`, `current3DView` und `currentVRMLModel`. Die Operationen erfordern, dass die Föderation das Format der von den SMS gelieferten Bilder versteht. Das Zusammenfügen der VRML-Modelle sollte einfach sein. Praktisch ist, wenn Bilder als Vektorgrafik vorliegen. Dann ergeben sich die wenigsten Schwierigkeiten bei der Integration. Bei Vorliegen von Rastergrafiken ist dies mit etwas mehr Aufwand jedoch auch möglich.

Die Operation `getModel` kann realisiert werden, indem die Föderation von allen beteiligten SMS die erforderlichen Teilmodelle anfordert. Zur Bildung des Gesamtmodells muss sie Einblick in das interne Format der Modelle haben.

Die Operation `findPath` kann im Gegensatz zu anderen Diensten nicht realisiert werden, indem die Föderation Teilergebnisse von SMS weiterverarbeitet. Ursache dafür ist, dass dieses Problem nicht gierig ist, d.h. lokale Optima führen nicht unbedingt zum globalen Optimum. Daher sollte die Föderation die Objektklassen der SMS laden, die die erforderlichen Straßen und Wege repräsentieren. Nach einer Integration der Modelle kann die Föderation den optimalen Pfad dann

bestimmen. Die Selektion der SMS, die mögliche Pfade enthalten, ist jedoch nicht trivial. Hilfreich dabei ist das Wissen über die Beziehungsverhältnisse räumlicher Modelle. Empfehlenswert ist ein inkrementeller Ansatz, der den Kreis der untersuchten SMS immer weiter ausdehnt. Das Verfahren kann abgebrochen werden, sobald sichergestellt ist, dass der kürzeste mögliche Weg länger ist als eine vorgegebene Maximaldistanz.

7.4 Kommunikationsinfrastruktur

Bisher wurde außen vor gelassen, auf welche Art und Weise die Kommunikation der einzelnen Komponenten vonstatten gehen soll. Da die Komponenten sich im Allgemeinen nicht auf demselben Rechner befinden, kommen lokale Kooperationsansätze wie *Pipes* oder *COM (Component Object Model)* nicht in Frage. Die möglichen Kandidaten sind stattdessen:

- Sockets
- HTTP/CGI (*Hypertext Transfer Protocol/Common Gateway Interface*)
- RPC (*Remote Procedure Call*) bzw. RMI (*Remote Method Invocation*) in Java
- DCOM (*Distributed Component Object Model*)
- CORBA (*Common Object Request Broker Architecture*)

Sockets eignen sich nicht gut, da sie auf einem sehr niedrigen Abstraktionsniveau agieren. Allerdings sind sie performant. Die Kombination von *HTTP* und *CGI* ist ebenfalls suboptimal, da *HTTP* ineffizient arbeitet und das objektorientierte Paradigma nicht unterstützt wird (siehe Spero, 1994). *HTTP-NG (HTTP-Next Generation)* mildert nur den ersten Nachteil. Ein Vorteil von *HTTP* und *CGI* ist, dass es auf den meisten Plattformen unterstützt wird.

Der *RPC* ist prinzipiell interessant für Nexus, da er auf einem höheren Abstraktionsniveau operiert. Er hat jedoch den Nachteil, dass er aus der imperativen Programmierwelt stammt. Besser geeignet ist *RMI*. Dieser in die Programmiersprache Java integrierte Mechanismus ermöglicht auf elegante Art die Kooperation von Objekten, die auf verschiedenen Rechnern residieren. Außerdem ist er auf mobilen Rechnern meist vorhanden und relativ performant (siehe Orfali und Harkey, 1997).

DCOM und *CORBA* sind Infrastrukturen, welche die Kooperation von Objekten realisieren, die in verschiedenen Sprachen geschrieben sind. Die Schnittstellen von Objekten können dynamisch zur Laufzeit bestimmt werden. Darüber hinaus werden eine Vielzahl von Diensten angeboten wie zum Beispiel Transaktionsunterstützung (siehe Orfali und Harkey, 1997). Für Nexus sind diese Infrastrukturen nicht unbedingt erforderlich, da zumindest die Schnittstellen aller Komponenten in Java implementiert werden. Sie besitzen zudem einen Overhead, da um jedes Objekt eine Schicht gelegt wird, die zwischen dem internen Format und dem Format der Objekte vermittelt. Ein weiterer Nachteil einer Infrastruktur wie *CORBA* ist, dass die Belastung des Klienten weiter steigt. Auf vielen mobilen Rechnern sind sie zudem nicht verfügbar.

Diese Gründe führen zur Wahl von RMI als Kommunikationsinfrastruktur. In der Implementierung wird jedoch HTTP/CGI verwendet, da das eingesetzte GIS dies erforderlich macht (vgl. Kapitel 8).

Im anschließenden Kapitel folgt die Übertragung der Konzepte dieses Kapitels auf ein konkretes GIS-basiertes System.

8 ArcView-basierte Architektur

Dieses Kapitel präsentiert die Architektur des SMS, die auf einer ausgewählten ArcView-Konfiguration fundiert. *ArcView* ist ein kommerzielles GIS der Firma ESRI, das meist auf Windows 98 bzw. NT aufsetzt (siehe ESRI, 1996a). Es ist das weltweit am meisten eingesetzte GIS. ArcView kann durch Skripte konfiguriert und programmiert werden, die in der Programmiersprache *Avenue* geschrieben sind (siehe ESRI, 1996c). Andere Produkte wie *MapInfo*, *Autodesk MapGuide* oder *Microstation GeoGraphics* werden nicht in die engere Wahl gezogen. Ein Überblick über den Markt ergab, dass sie in Bezug auf den SMS nicht mehr bieten als ArcView.

Zu Beginn des Kapitels wird ermittelt, welche ArcView-Konfiguration sich am besten für den SMS eignet. Dazu werden Kriterien definiert, an denen sich die einzelnen Konfigurationen messen lassen müssen. Anschließend wird die Architektur beschrieben, die auf der ausgewählten Konfiguration basiert.

8.1 Auswahl einer Systemkonfiguration

8.1.1 Auswahlkriterien

Die ArcView-Konfiguration, die als Basis des SMS eingesetzt wird, sollte möglichst gut die Ziele des SMS unterstützen (vgl. Kapitel 5). Aufgrund des prototypischen Charakters des Systems, das im Rahmen dieser Arbeit zu entwickeln ist, spielt der Zeitfaktor auch eine wichtige Rolle. Zur Bewertung möglicher Konfiguration werden Kriterien eingeführt. Die Kriterien sind wie folgt:

- *Funktionalität*: Das GIS sollte einen Großteil der Funktionalität gemäß den Anforderungen aus Kapitel 5 besitzen. Insbesondere Netzwerkanalysen und die Unterstützung dreidimensionaler Objekte sind erforderlich. Um die exakte Funktionalität zu erhalten, die der SMS benötigt, sollte das GIS programmierbar sein.
- *Datenaustausch*: Zwischen GIS und SMS müssen sowohl Daten als auch Bilder ausgetauscht werden können; letztere jedoch nur vom GIS zum SMS.

- *Steuerbarkeit*: Das GIS sollte eine Schnittstelle anbieten, über die es von außen gesteuert werden kann. Das bedeutet, dass von außerhalb zum Beispiel bestimmte Aktionen des GIS angestoßen werden können. Dieser Punkt ist essenziell, da ansonsten ein GIS nicht in den SMS eingebunden werden kann. Erstrebenswert ist eine Interaktion zwischen Objekten, die einen Zustand besitzen.
- *Leistungsanforderungen an Klienten*: Die Leistungsanforderungen an den Klienten sollten minimal sein. Es sollte insbesondere nicht nötig sein, spezielle Software auf dem Klienten zu installieren. Das würde die Menge der möglichen mobilen Rechner, die als Klientenplattform in Frage kommen, unnötig einschränken.
- *Entwicklungszeit*: Da der in dieser Arbeit zu entwickelnde SMS prototypischen Charakter besitzt, sollte seine Entwicklung schnell zu einem Ergebnis führen.

Anhand dieser Kriterien werden die folgenden ArcView-Konfigurationen beurteilt.

8.1.2 Beurteilung potenzieller Konfigurationen

Bevor die einzelnen ArcView-Konfigurationen vorgestellt werden, soll das erste Kriterium, die Funktionalität, untersucht werden. Da alle Konfigurationen auf ArcView als Basissystem basieren, genügt es, die Funktionalität von ArcView einmal zu beurteilen.

ArcView stellt Funktionen zur Verfügung, die die Aufgaben der Komponenten Spatial Object Manager, Spatial Analyst und Presenter unterstützen (vgl. Kapitel 7.1). Die dritte Dimension der räumlichen Objekte wird dabei durch die ArcView-Erweiterung *3D Analyst* bereitgestellt. Zur Realisierung des Navigators ist die Erweiterung *Network Analyst* erforderlich. Sie führt Netzwerkanalysen durch. Der HTML Generator sowie der Client Manager werden nicht direkt unterstützt, sie können durch Avenue-Skripte jedoch implementiert werden. Die Funktionen des Spatial Event Managers werden leider nicht angeboten. Ein weiterer Nachteil ist, dass ArcView nicht objektorientiert arbeitet, sondern die räumlichen Objekte in flachen Tabellen hält. Die Forderung der Programmierbarkeit erfüllt ArcView.

Avenue stellt verschiedene Mechanismen zur Verfügung, die ArcView mit seiner Umgebung kommunizieren lassen. Die folgenden Konfigurationen unterscheiden sich primär durch den Kommunikationsmechanismus, der verwendet wird.

ArcView mit RPC

Eine erste mögliche Konfiguration basiert auf dem RPC-Mechanismus. Der RPC von Avenue erfüllt die *ONC (Open Network Computing)*-Norm, die von Java in der Version 1.2 nicht unterstützt wird.

Die Steuerbarkeit dieser Konfiguration ist gegeben. Es können Befehlswoorte übermittelt werden, die die gewünschten Operationen auslösen. Weiter ist kein Installationsbedarf beim Klienten vorhanden. Die Entwicklungszeit ist — was den Server betrifft — akzeptabel, da der RPC-Server nur so programmiert werden muss, dass die erwünschten Skripte aufgerufen werden. Nachteilig ist, dass ein Webserver konfiguriert werden muss, um die HTML-basierte Schnittstelle anzubieten. Darüber hinaus sind Java-Klassen zu entwickeln, die die Kommunikation mit dem RPC-Server realisieren und der ONC-Norm entsprechen. Schwerwiegender ist, dass der Datenaustausch nicht adäquat ist. Der RPC von ArcView erlaubt nämlich nur den Austausch von Zeichenketten und Zahlen. Daher können keine Bilder übergeben werden. Aus diesem Grund kommt diese Konfiguration nicht in Frage.

ArcView mit DDE

DDE (Dynamic Data Exchange) ist ein Windows-proprietärer Mechanismus zum Austausch von Daten zwischen Applikationen. Avenue unterstützt DDE. Da DDE nur lokal funktioniert, benötigt es für jede Kommunikation nach außen den Aufruf einer Kommunikationseinheit, an die es die Daten mittels DDE weiterleitet.

Wieder sind die Leistungsanforderungen an Klienten gering, da kein Installationsbedarf besteht. Diese Konfiguration eignet sich jedoch genauso wenig für den SMS wie die RPC-basierte Architektur. Der DDE-Mechanismus von ArcView ist bezüglich der Parameter nämlich noch restriktiver als RPC. Er erlaubt lediglich, Zeichenketten auszutauschen, die nicht länger als 255 Zeichen sind. Dies verhindert nicht nur einen adäquaten Datenaustausch, sondern erschwert auch die Steuerung von außen. Daher kommt auch diese Konfiguration nicht in Frage.

MapObjects

MapObjects stammen aus dem ArcView-Umfeld. Es sind Active-X Komponenten, die GIS-Funktionalität in Applikationen einbetten (siehe ESRI, 1996d). Sie besitzen eine außergewöhnliche Flexibilität, um eine an bestimmte Bedürfnisse angepasste Benutzungsschnittstelle zu entwickeln.

Ein ausreichender Datenaustausch sowie eine angemessene Steuerbarkeit sind vorhanden. Dieser Ansatz wird jedoch nicht weiterverfolgt, da die Voraussetzungen für den Applikationseinsatz beim Klienten inakzeptabel sind. MapObjects erfordern mindestens 16 MB Arbeitsspeicher.

Basierend auf MapObjects wird der *MapObjects Internet Map Server (IMS)* angeboten (siehe ESRI, 1996e). Er erzeugt WWW-Seiten, in denen Bilder eingebettet sind, welche nach den Vorgaben von Benutzern dynamisch erstellt werden. Die Leistungsanforderungen an den Klienten sind daher gering. Diese Variante besitzt jedoch den Nachteil, dass die Entwicklungszeit recht hoch ist. Der Server muss von Grund auf entwickelt werden. Ansonsten ist diese Lösung gut, da die Vorzüge der MapObjects auch für den MapObjects-IMS gelten.

SDE

Zum Zugriff auf entfernte Datenbanken verfügt ArcView über die Möglichkeit der Anbindung an die *Spatial Database Engine (SDE)*, (siehe ESRI, 1998a; ESRI, 1998b). Die SDE ist in der Lage, räumliche Analysen im zweidimensionalen Raum durchzuführen.

Ein weitere Konfiguration besteht darin, dass die SMS-Schnittstelle direkt auf die Dienste der SDE zurückgreift. Da die Funktionalität der SDE nicht vergleichbar ist mit der von ArcView, müsste jedoch innerhalb der Schnittstelle ein Großteil der Arbeit verrichtet werden. Navigationsfunktionalität sowie räumliche Ereignisse wären beim Klienten zu realisieren. Dies widerspricht jedoch der Forderung, die Leistungsanforderungen an den Klienten zu minimieren. Außerdem kann die HTML-basierte Schnittstelle nicht angeboten werden, da als Server SDE verwendet wird, der die Anfragen in HTTP/CGI-Format nicht versteht.

ArcView-Internet Map Server

Auf der Basis von ArcView bietet ESRI einen Internet Map Server (IMS) an (siehe ESRI, 1996b). Dieser erweitert ArcView insofern, als dass von ArcView verwaltete Karten in Kooperation mit einem Webserver im Internet angeboten werden. Als Kommunikationsprotokoll muss daher HTTP verwendet werden.

Der Datenaustausch mit ArcView funktioniert bei dieser Konfiguration, da ArcView sowohl Zeichenketten als auch Bilder liefern kann. ArcView lässt sich außerdem steuern, da in den URLs (*Uniform Resource Locator*), die er vom Webserver erhält, beliebige Befehle und Argumente enthalten sein können. In Bezug auf die Leistungsanforderungen an den Klienten ist diese Lösung

ebenfalls gut. Die HTML-basierte Schnittstelle wird sogar direkt unterstützt. Das letzte Kriterium ist die Entwicklungszeit. Auch in diesem Punkt schneidet die hier untersuchte ArcView-Konfiguration gut ab. Auf Seiten des Servers müssen lediglich Avenueskripte implementiert werden, die die gewünschten Funktionen realisieren. Um Kommunikationsaspekte kümmert sich der IMS.

Die Wahl der ArcView-Konfiguration fällt daher auf den ArcView-IMS. Er bietet einen Großteil der erforderlichen Funktionalität an, erlaubt angemessenen Datenaustausch, ist von außen steuerbar und stellt keinen unnötigen Forderungen an das Leistungsvermögen der Klienten. Ein weiterer wichtiger Punkt ist die einfache und schnelle Realisierung, die mit dem ArcView-IMS möglich ist. Der MapObjects-IMS ist jedoch nur wenig schlechter geeignet. Sein Hauptmanko ist die lange Entwicklungszeit. Der folgende Abschnitt stellt die Architektur, die auf dem ArcView-IMS gründet, näher vor.

8.2 Beschreibung der Architektur

Bevor auf die Architektur an sich eingegangen wird, soll Avenue kurz vorgestellt werden. Das ist notwendig, da sich die Wahl der Programmiersprache auf die Architektur eines Systems auswirkt. Anschließend wird die Architektur des SMS mitsamt der Föderation und den Schnittstellen zu den anderen Komponenten erläutert. Danach wird auf die Komponenten näher eingegangen, die zu implementieren sind (vgl. Kapitel 9.1). Dabei stehen zunächst die Skripte im Vordergrund, die nötig sind, um die Funktionalität des SMS anzubieten. Da ArcView dem relationalen Paradigma folgt, geht es anschließend um die Tabellen, die verwendet werden, um sowohl räumliche Objekte als auch weitere Informationen wie den Klientenzustand zu speichern. Erforderliche globale Variablen werden dort ebenfalls behandelt. Zum Abschluss wird die Architektur der Benutzer-API erläutert.

8.2.1 Einführung in Avenue

Avenue ist eine kompilierte untypisierte Programmiersprache, die objektorientierte Elemente enthält und turingmächtig ist (siehe ESRI, 1996c). Zielsetzung von Avenue ist die Konfigurierung von ArcView, daher können Avenueskripte nur innerhalb von ArcView ausgeführt werden.

Im Gegensatz zu herkömmlichen Programmiersprachen erlaubt es Avenue nicht, Typen oder Klassen zu definieren. Nachteilig ist weiter, dass keine Konstanten definiert werden können. Auch bei der Generierung von Objekten existieren Restriktionen. Eine Vielzahl von Klassen kann nur einmal instanziiert werden, und diese Instanz wird vom ArcView-System bereitgestellt. Zum Beispiel gibt es ein *application object*, das die aktuelle ArcView-Instanz widerspiegelt. Über dieses Objekt kann dann auf die gegenwärtig verfügbaren Projekte, Sichten, Themen, etc. von ArcView zugegriffen werden. ArcView hat zu jedem Zeitpunkt maximal ein Projekt geöffnet, in dem unter anderem Skripte, Tabellen, Sichten und dreidimensionale Szenen enthalten sind. Eine Sicht entspricht einer Karte. Sie umfasst mehrere Themen, die jeweils eine Objektklasse verwalten.

Avenue kennt nicht das Konzept der Modularisierung. Prozeduren, Funktionen und Methoden gibt es nicht. Als Ersatz können Skripte geschrieben werden, die von anderen Skripten aufgerufen werden. Die Übergabe eines Parameters, der auch aus einer Liste bestehen kann, ist möglich. Es ist Aufgabe des aufgerufenen Skripts zu überprüfen, ob die erwarteten Parameter übergeben wurden bzw. ob sie den richtigen Typ besitzen. Avenue ist weiter nicht rekursiv.

Mittels einer *DLL (Dynamic Link Libraries)* kann Funktionalität zu ArcView hinzugefügt werden. DLL ist ein Windows-Konzept, das es ermöglicht, in binärform vorliegende Programme von Windows-Programmen zur Laufzeit aufzurufen und in ihren Prozess einzubinden. DLLs können jedoch nicht selbständig ausgeführt werden.

DLL-Funktionen können direkt von Avenue aus aufgerufen werden, wobei Daten in beide Richtungen übergeben werden können. über den Befehl *AVExec* können in aufgerufenen DLLs Avenue-Befehle ausgeführt werden. Das Ergebnis wird dabei immer als Zeichenkette zurückgegeben.

8.2.2 Architektur auf Basis des ArcView-IMS

ESRI bietet zusammen mit dem ArcView-IMS das Applet *MapCafé* an, das von ArcView erzeugte JPEG- oder GIF-Bilder darstellt. Es bietet außerdem einige Operationen wie Zoomen oder Selektieren an.

Die Architektur, die auf dem IMS aufbaut, verwendet MapCafé jedoch nicht (vgl. Abb. 8.1). Ursache dafür ist, dass sich MapCafé nur eingeschränkt programmieren lässt. Stattdessen werden die von ArcView erzeugten Daten oder Bilder in der Schnittstelle entgegen genommen und in der Benutzer-API in Java-Objekte umgewandelt. Diese werden Applikationen dann zur Verfügung gestellt. Die Basisschnittstelle erhält vom SMS räumliche Modelle in Dateiform.

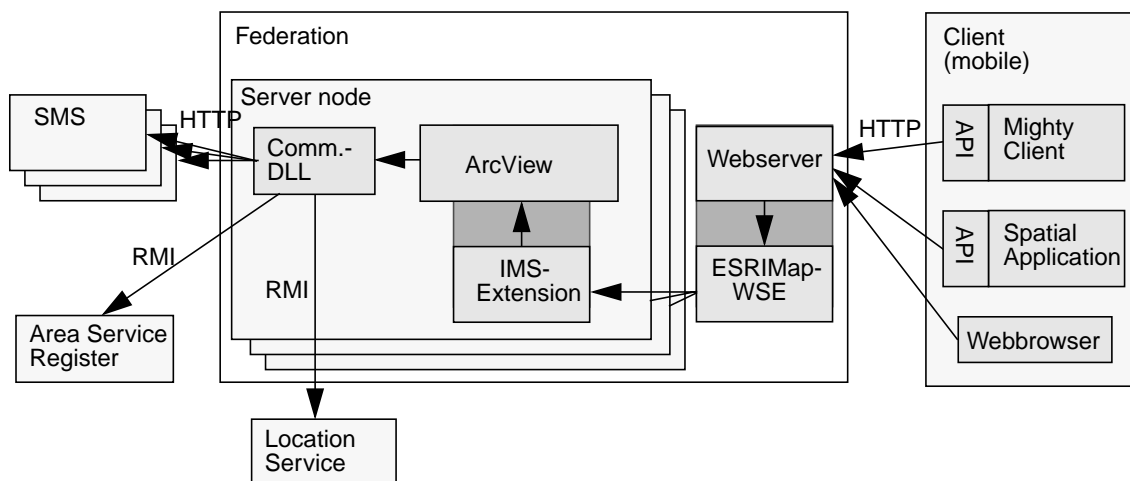


Abbildung 8.1: Architektur basierend auf ArcView-Internet Map Server (Teil 1)

Die Abbildung 8.1 betont die Rolle der Föderation in der Architektur. Die Föderation beinhaltet einen Webserver, der URLs empfängt. Die *ESRIMap Web Server Erweiterung (WSE)* leitet den Argumententeil der URLs an die *Internet Map Server Erweiterung (IMS)* weiter. Letztere übergibt die empfangenen Zeichenketten an ArcView, wo diese verarbeitet werden. Sowohl ESRIMap WSE als auch IMS werden von ESRI bereitgestellt.

ArcView ist Teil der Föderation, da dessen Funktionalität zur Integration von Teilergebnissen benötigt wird. Beim Dienst FindPath muss die Föderation beispielsweise den optimalen Weg finden.

Zur Leistungssteigerung kann die ArcView-Instanz repliziert werden. Die ESRIMap WSE verteilt Anfragen dann gleichmäßig auf die verschiedenen Instanzen.

Da in dieser Architektur Anfragen von Webservern entgegen genommen werden, ist es zwingend, HTTP als Kommunikationsprotokoll einzusetzen. Probleme, die sich aus dem parallelen Zugriff mehrerer Klienten ergeben, brauchen nicht berücksichtigt werden, da die ESRIMap WSE immer nur eine Anfrage an eine ArcView-Instanz weiterleitet. Sie blockiert weitere Anfragen bis sie eine Antwort von ArcView erhält. Erst dann wird die als nächste eingetroffene Anfrage weitergeleitet.

Der Zugriff der Föderation auf SMS geschieht über eine *Communication DLL*. Das ist notwendig, da der IMS immer nur eine Verbindung zum Webserver aufrechterhalten kann. Da HTTP verbindungsorientiert arbeitet, blockiert der Klient, der den Dienst aufgerufen hat und auf eine Antwort wartet, diese Verbindung. Daher wird in ArcView die DLL aufgerufen. Eine weitere Aufgabe der Communication DLL ist die Einbindung des Area Service Registers über RMI. Die Steuerung des SMS geschieht ebenfalls über die Communication DLL.

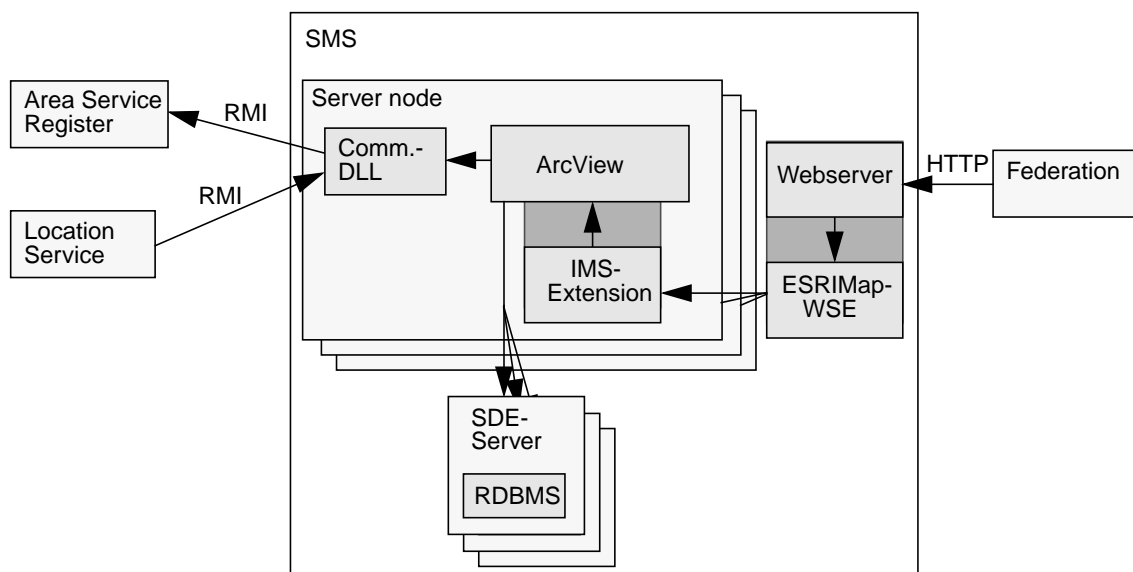


Abbildung 8.2: Architektur basierend auf ArcView-Internet Map Server (Teil 2)

Der SMS ist ähnlich wie die Föderation aufgebaut. Abbildung 8.2 stellt ihn detailliert dar. Er verfügt ebenfalls über einen Webserver, der Anfragen an ArcView weiterleitet. Falls ArcView repliziert ist, müssen jedoch Datenbanken verwendet werden, da beim Zugriff auf Dateien Konsistenzprobleme auftreten können. Die Daten können über SDE auf mehrere Datenbanken verteilt sein.

Die Communication DLL ermöglicht die Registrierung des SMS beim Area Service Register. Des Weiteren nimmt sie Ortsinformationen vom Location Service entgegen.

Um Arbeitslast zum Klienten zu verteilen, kann der mächtige Klient lokal einen eigenen SMS einsetzen.

Im nächsten Abschnitt werden die Avenueskripte besprochen, die zur Realisierung des SMS eingesetzt werden. Die Communication DLL, die Föderation und die HTML-basierte Schnittstelle werden in diesem Kapitel nicht weiter betrachtet, da sie nicht implementiert werden (vgl. Kapitel 9.1).

8.2.3 Eingesetzte Skripte

Dieser Unterabschnitt beschreibt, wie in ArcView die erforderlichen Funktionen des SMS bereitgestellt werden. In Kapitel 7.1 wurde eine allgemeine Architektur des SMS entworfen, die mehrere Komponenten umfasst. Da Avenue nicht objektorientiert ist, müssen diese Komponenten auf Skripte abgebildet werden. Aufgrund der mangelhaften Modularisierungs- und Abstraktionsmöglichkeiten von Avenue ist die prinzipielle Vorgehensweise die, dass für jeden Dienst, der in der Schnittstelle angeboten wird, ein Skript bereitgestellt wird. Die Abbildungsregel ist also recht einfach. Zur Wiederverwendung von Code ist es an einigen Stellen sinnvoll, andere Skripte in diese „Dienstskripte“ einzubinden.

Folgend werden zuerst die Dienstskripte und anschließend einige administrative Skripte angesprochen. Abschließend wird das Verhalten des SMS an einem Beispiel veranschaulicht.

Die Abbildung 8.3 stellt dar, wie ArcView Dienste des SMS realisiert. Der Argumententeil von URLs wird über die IMS-Erweiterung an ein designiertes Skript, das Dispatch-Skript, übergeben. In ihm ist ein Befehl, der den gewünschten Dienst spezifiziert, sowie dessen Parameter enthalten.

Die Dienstskripte arbeiten folgendermaßen: Sie lesen die Parameter ein und weisen sie lokalen Variablen zu. Das aufgerufene Skript `ProcessStandardArguments` verarbeitet die so genannten Standardargumente. Dies sind Argumente, die den Zustand von Klienten verändern. Sie können in jeder URL enthalten sein. Dann überprüfen die Dienstskripte, ob alle erforderlichen Parameter vorhanden sind. Ist dies nicht der Fall, dann bricht das Skript ab und sendet dem aufrufenden Klienten eine Fehlermeldung zurück. Ansonsten führt das Skript seine Operation durch und sendet

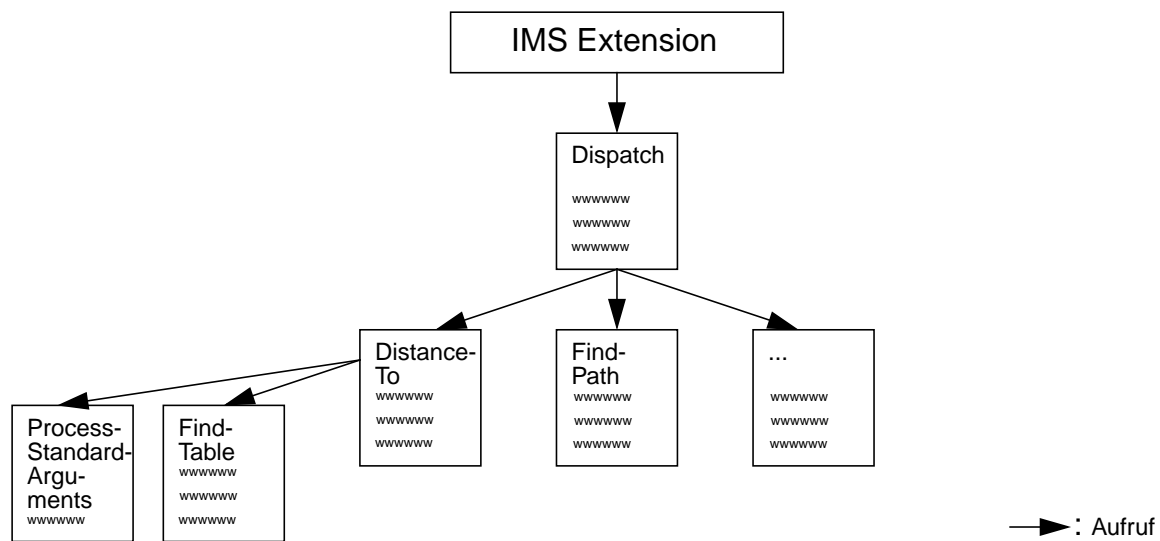


Abbildung 8.3: Aufrufhierarchie der Avenueskripte

das Ergebnis dem Klienten zu. Da die Daten der räumlichen Objekte und auch viele andere Daten in Tabellen vorliegen, werden typischerweise Tupel bestimmter Tabellen gelesen, manipuliert oder hinzugefügt. Um die Tabelle zu finden, die ein durch einen Bezeichner spezifiziertes räumliches Objekt enthält, wird das Skript `FindTable` eingesetzt. Dieses Skript durchsucht die Tabellen, welche räumliche Objekte verwalten, und liefert die Tabelle zurück, die das gesuchte Objekt beinhaltet. Die übrigen Skripte, die nicht direkt einen Dienst anbieten, werden nur von einzelnen Dienstsripten verwendet und sollen daher hier nicht näher betrachtet werden.

Falls der Bedarf besteht, neue Dienstsripte hinzuzufügen, so muss nur das `Dispatch`-Skript modifiziert sowie im neuen Skript `ProcessStandardArguments` aufgerufen werden.

Administrative Skripte

Der SMS verfügt über Skripte, die ihn initialisieren, starten und beenden. Das Initialisierungsskript erzeugt alle erforderlichen Tabellen und setzt die globalen Variablen auf Startwerte. Das Skript, das den SMS stoppt, schreibt alle globalen Variablen in eine Datei und aktualisiert die Tabellen. Dies erlaubt es dem Startskript, den SMS durch Laden der Datei wieder in genau jenen Zustand zu bringen, in dem er beendet wurde.

Beispiel für das Vorgehen der Dienstsckripte

Nachfolgend soll an einem konkreten Beispiel das Vorgehen des SMS erläutert werden. Eine vom Klienten spezifizierte und von der Benutzer-API erzeugte URL sieht zum Beispiel aus wie in Abbildung 8.4 angegeben. Neben dem Server und dem Port wird die DLL genannt, welche die ESRIMap WSE realisiert. Der bei nameX angegebene Name identifiziert ArcView-Instanzen, die gewünschte Modelle anbieten. Die restliche Zeichenkette wird an das Skript Dispatch von ArcView übergeben.

URL specified by client:

`http://sms.stuttgart.de:80/scripts/esrimap.dll?nameX=nexusMap&Cmd=FindPath
&model=airport&sessionID=8&oD=3&orig=0.1,4.3&dest=29.1,4.6`

Input to Dispatch:

`Cmd=FindPath&model=airport&sessionID=8&oD=3&orig=0.1,4.3&dest=29.1,4.6`

script name mandatory processed by ProcessStandardArguments arguments of FindPath

Abbildung 8.4: Zeichenkette, die das Dispatch-Skript erhält

Die Zeichenkette, die Dispatch übergeben bekommt, besteht aus Name/Wert-Paaren. Über den Wert von Cmd findet Dispatch heraus, welches Dienstsckript initiiert werden soll. Die Werte von model und sessionID dienen der Identifizierung des räumlichen Modells und der Sitzung. Die Angabe des Modells ist notwendig, da ein SMS mehrere räumliche Modelle verwalten kann. Die Parameter loD, orig und dest werden an das Skript FindPath weitergereicht. Dort wird das Skript ProcessStandardArguments aufgerufen, das den Zustand des Klienten durch Änderung des Detaillierungsgrads aktualisiert. FindPath verarbeitet die übrigen Parameter und sendet das Ergebnis über die bestehende Verbindung zum Klienten zurück.

Es wird betont, dass Anwendungsentwickler, die den SMS verwenden, mit obigen URLs nicht hantieren müssen. Sie werden durch die Benutzer-API davor abgeschirmt. Anwendungsentwickler greifen durch den einfachen Aufruf von Methoden auf den SMS zu.

8.2.4 Erforderliche Tabellen und globale Variablen

Wie schon angedeutet, stellt ArcView Mechanismen zur Verfügung, um Daten in Tabellenform zu speichern. Die Daten können entweder als Datei oder innerhalb einer Datenbank gehalten werden. Im Rahmen dieser Arbeit werden Dateien verwendet, da sich dies positiv auf die Entwicklungszeit auswirkt. Der Übergang auf eine Datenbank ist jedoch einfach möglich.

Folgend werden die Tabellen beschrieben, die zur Realisierung des SMS eingesetzt werden. Im Anschluss daran wird auf die genutzten globalen Variablen eingegangen.

- Die Tabelle **Models** speichert die Namen der räumlichen Modelle eines SMS. Diese Tabelle liegt pro SMS nur einmal vor. Die folgenden Tabellen werden für jedes räumliche Modell separat benötigt.
- **ClientStates** enthält Informationen über den Zustand von jedem Klienten, mit dem gerade eine Sitzung besteht. Die Klienten werden anhand eines Sitzungsbezeichners unterschieden. Weiter enthält diese Tabelle die gewünschte Bildgröße, die Eckpunkte des Ausschnitts der Welt, der dargestellt wird, den Detaillierungsgrad, Kennzahlen, welche die Leistungsfähigkeit von Klient und Netzwerk angeben, und schließlich den Zeitpunkt der Sitzungseröffnung. Der letzte Wert wird verwendet, um Sitzungen zu schließen, die vergessen wurden zu beenden. Die Selektionsmengen der Klienten werden mit globalen Variablen verwaltet.
- **ExemptList** hält die Namen der Objektklassen, die in räumlichen Analyseanfragen nicht verwendet werden sollen. Die Intention hinter dieser Tabelle ist es zu verhindern, dass beispielsweise Straßen als Ergebnis von räumlichen Analyseanfragen zurückgeliefert werden, obwohl dies nicht erwünscht ist.
- **SaveIDs** speichert die Bezeichner neu erzeugter Objekte, um die *At Most Once*-Semantik der Erzeugung neuer Objekte zu gewährleisten (siehe Rothermel, 1997). Für die übrigen Dienste genügt die *At Least Once*-Semantik, da sie idempotent sind. Kapitel 9 behandelt dieses Thema ausführlicher.
- **ObjCIH** verwaltet die Vererbungshierarchie von Klassen räumlicher Objekte. Die Objekte jeder Klasse werden in einer eigenen Tabelle gespeichert. Das Konzept hinter diesem Vorgehen wird im Anschluss an die Auflistung kurz erläutert.

In ArcView ist es nicht möglich, Vererbungsbeziehungen zwischen Objektklassen zu definieren. Da das objektorientierte Paradigma jedoch Vererbungen zwischen Klassen vorsieht, ist es nötig, Vererbungshierarchien für Klassen räumlicher Objekte selbst aufzubauen (vgl. Kapitel 5.2). Dazu wird die Tabelle `ObjCIH` verwendet. Dort ist für jede Klasse angegeben, welche Superklasse sie besitzt. Auf diese Art ist die Einfachvererbung zwischen Klassen möglich. Rekursiv kann bestimmt werden, ob eine Klasse von einer beliebigen anderen erbt.

Globale Variablen

Avenue erlaubt die Verwendung globaler Variablen. Im Gegensatz zu den gewöhnlichen Variablen, die nur innerhalb eines Skripts gültig sind, erstreckt sich der Gültigkeitsbereich globaler Variablen auf ein gesamtes ArcView-Projekt.

Der SMS nutzt globale Variablen, um den letzten vergebenen Sitzungsbezeichner zu speichern. Bei der Öffnung einer neuen Sitzung kann so immer ein eindeutiger Bezeichner erzeugt werden. Der letzte zugewiesene Objektbezeichner sowie der Pfad des Verzeichnisses, in dem vom SMS benötigte Dateien angelegt werden, werden ebenfalls in globalen Variablen gespeichert. In einer weiteren globalen Variable wird eine Liste gehalten, die der Verwaltung der aktuellen Selektionsmenge für jeden Klienten dient. Tabellen eignen sich dafür nicht, da in ihnen nur Zahlen und Zeichenketten bis zu einer bestimmten Länge enthalten sein können.

8.2.5 Architektur der Benutzer-API

Da die Benutzer-API das Herzstück der SMS-Schnittstelle ist, wird ihre Architektur vorgestellt. Die Basisschnittstelle und die Management-API können analog entworfen werden. Die Entwicklung der HTML-basierten Schnittstelle wird nicht weiter betrachtet. Einen Überblick über die Architektur der Benutzer-API liefert Abbildung 8.5. Die Kooperation der Komponenten wird im Anschluss an die statische Analyse der Architektur betrachtet. Die Darstellungsform ist *UML* (*Unified Modeling Language*; siehe Jacobson et al., 1999).

Die zentrale Komponente der API ist `SMAccess`. Sie bietet die Dienste der Benutzer-API an. Außerdem enthält sie Methoden, die eine Sitzung mit dem SMS eröffnen und schließen.

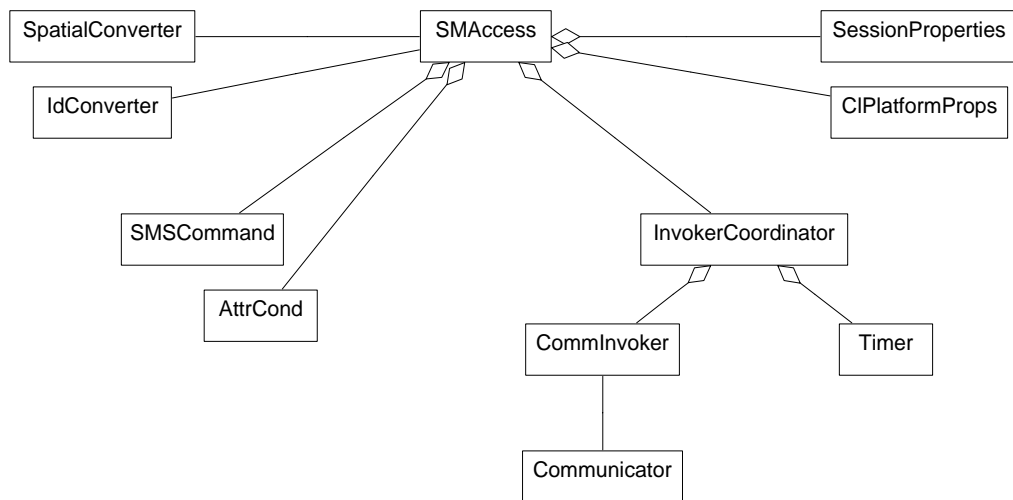


Abbildung 8.5: Architektur der Benutzer-API

Die Kommunikation mit dem SMS übernimmt die Klasse `InvokerCoordinator`. Über die Klasse `Commlnvoke`, die hauptsächlich der Bereitstellung eines Threads dient, ruft sie den `Communicator` auf. Dort wird eine Verbindung mit dem Server aufgebaut und findet der Austausch von Daten statt. Der `InvokerCoordinator` unterbricht die Verbindung beim Überschreiten einer vorgegebenen Timeout-Zeit, die von der Klasse `Timer` gestoppt wird. Falls es Probleme mit dem Verbindungsaufbau gibt, versucht es der `InvokerCoordinator` erneut. Die Anzahl der Versuche sowie das Timeout-Intervall können variiert werden.

`SessionProperties` und `CIPlatformProps` halten Eigenschaften der aktuellen Sitzung mit dem SMS bzw. allgemeine Eigenschaften der zugrunde liegenden Plattform fest.

`SMSCommand` repräsentiert eine Zeichenkette, die als Teil einer URL zum SMS gesendet wird und die den gewünschten Dienst angibt sowie die nötigen Parameter enthält. Es stellt weitere Mechanismen bereit, die das Erzeugen korrekter Zeichenketten erleichtern. `AttrCond` ist eine Klasse, die Bedingungen auf den Sachdaten räumlicher Objekte spezifiziert. Eine solche Bedingung spezifiziert zum Beispiel alle virtuellen Objekte einer bestimmten Klasse, die schon länger als ein bestimmter Zeitraum bestehen. `AttrCond` verfügt unter anderem über eine Methode zur Generierung einer Zeichenkette, welche die Bedingung repräsentiert und für den SMS verständlich ist.

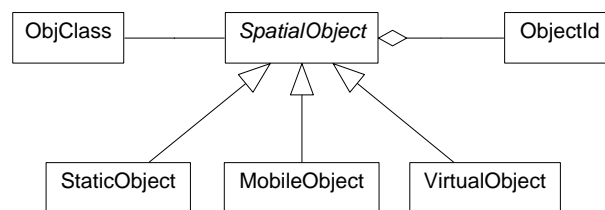


Abbildung 8.6: Klassen für räumliche Objekte

`SpatialConverter` und `IdConverter` sind schließlich Klassen, deren Zweck darin liegt, zwischen Javaobjekten und den Zeichenketten zu vermitteln, die an den SMS geschickt bzw. von diesem empfangen werden. `SpatialConverter` kümmert sich um die Klassen des Pakets `nexus.location` (siehe Fritz, 1999). Diese Klassen stellen beispielsweise geometrische Formen und Koordinaten dar. `SpatialConverter` transformiert darüber hinaus Koordinaten zwischen verschiedenen Koordinatensystemen. `IdConverter` betrifft Bezeichner aus dem Paket `nexus.ls` (siehe Wünsche, 1999). Dazu gehören etwa Objektbezeichner.

Die Abbildung 8.6 zeigt die Klassen, die in der Benutzer-API räumliche Objekte repräsentieren. Die Klasse `Objectid` stammt aus dem Paket `nexus.ls`. `ObjClass` repräsentiert die Klasse, die ein räumliches Objekt beim SMS besitzt.

Im Anschluss an diese statische Betrachtung der Architektur der Benutzer-API wird nun beispielhaft das Zusammenspiel der einzelnen Komponenten erklärt (vgl. Abb. 8.7).

Es wird angenommen, dass die raumbezogene Applikation zuvor eine Sitzung eröffnet hat. `SMAccess` nimmt in diesem Zustand einen Dienstaufwurf der räumlichen Applikation entgegen. Es werden die Objekte gesucht, die sich innerhalb eines bestimmten Kreises befinden. `SMAccess` bindet den `SpatialConverter` ein, um die Koordinaten des angegebenen Kreises in das Koordinatensystem des SMS zu transformieren. Anschließend erzeugt sie mit Hilfe von `SMSCommand` einen adäquaten Befehl, den der SMS verarbeiten kann. Diesen transferiert der `Communicator` zum SMS (`InvokerCoordinator` und `CommInvoker` sind der Übersichtlichkeit halber nicht eingezeichnet). Der Webserver nimmt den Befehl entgegen und leitet ihn an `ArcView` weiter. Dort wird das entsprechende Dienstsript ausgeführt und das Ergebnis in Form eines Stroms von Zeichen zurückgeliefert (vgl. Kapitel 8.2.3). Die Zeichen stellen die Bezeichner der selektierten

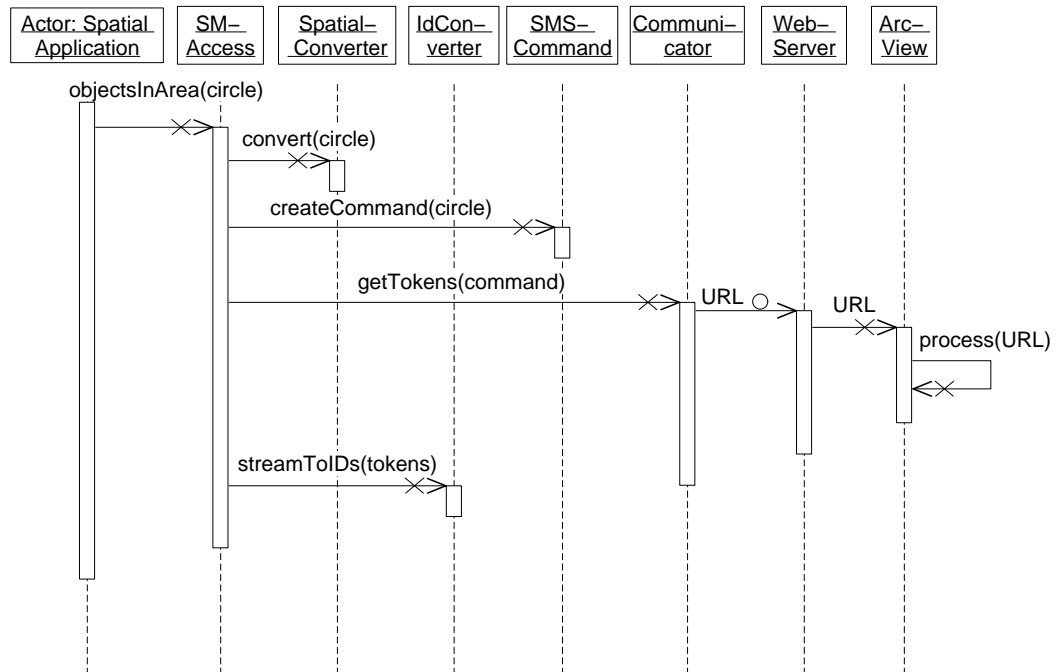


Abbildung 8.7: Dynamische Betrachtung der Architektur der Benutzer-API

räumlichen Objekte dar. `SMAccess` leitet den Strom an `IdConverter` weiter, der daraus ein Feld von `ObjectId`-Instanzen erzeugt. `SMAccess` gibt diese Instanzen als Ergebnis der Anfrage an die raumbezogene Applikation zurück.

Das folgende Kapitel stellt relevante Aspekte der Implementierung dieser Architektur vor.

9 Implementierungsaspekte

Dieses Kapitel widmet sich ausgewählten Aspekten der Implementierung des SMS. Es besteht aus vier Abschnitten. Im ersten Abschnitt wird angegeben, in welchem Umfang der SMS realisiert wird. Der zweite Abschnitt spricht die Implementierung der Avenueskripte an. Der dritte Abschnitt betrifft die Implementierung der Benutzer-API. Die Beispielapplikation, die entwickelt wurde, um die Fähigkeiten des implementierten SMS zu demonstrieren, wird im letzten Abschnitt vorgestellt.

9.1 Implementierungsumfang

Da der in dieser Arbeit zu entwickelnde SMS prototypischen Charakter hat, wurde aus Zeitgründen nur ein repräsentativer Anteil implementiert. Es wurde sich auf Funktionen konzentriert, die elementar sind und die einen Großteil der Anforderungen abdecken. Bei der Auswahl der Funktionen wurden die Funktionsklassen der Benutzer-API aus Kapitel 6.2.2 berücksichtigt. Die Föderation wird nicht realisiert, weil sie im Rahmen dieser Arbeit nur konzeptionell zu untersuchen ist.

Im Einzelnen bestehen folgende Einschränkungen des SMS:

- Die Implementierung unterstützt nur zweidimensionale Objekte.
- Die Basisschnittstelle sowie die HTML-basierte Schnittstelle werden nicht betrachtet, da sich die Kernfunktionalität des SMS in der Benutzer-API befindet.
- Die Föderation existiert nicht, da sie nur konzeptionell angedacht werden sollte.
- Der Zugriff eines SMS auf den Home Server eines mobilen Objekts ist nicht möglich.
- Die Communication DLL wird nicht entwickelt, da sie erst im Zusammenhang mit der Föderation wichtig wird.
- Nicht alle Dienste aus Kapitel 6 werden implementiert. Die implementierten Dienste sind in Anhang A.4 aufgelistet.

9.2 Implementierung der Avenueskripte

Wie im vorherigen Kapitel beschrieben, wird jede Funktion der Schnittstelle durch ein Dienstskript realisiert. Die prinzipielle Vorgehensweise dieser Skripte wurde dort erläutert. Bei der Implementierung der Skripte ist darauf zu achten, die Funktionalität von ArcView möglichst umfassend einzusetzen. Das vereinfacht die Realisierung des SMS. Zur Berechnung der Distanz zwischen zwei räumlichen Objekten wird beispielsweise die Operation `Distance` der Avenueklasse `Shape` verwendet. Die Geometrie eines räumlichen Objekts wird in Avenue von Objekten der Klasse `Shape` repräsentiert. An manchen Punkten ist es jedoch nicht möglich, die ArcView-Funktionalität direkt zu verwenden, da erforderliche Funktionen nicht angeboten werden. Auf diese Punkte wird hier besonders eingegangen.

Nachfolgend werden die Handhabung der Selektionsmengen von Klienten, die Berücksichtigung der Vererbung zwischen Objektklassen, Kommunikationsaspekte sowie die Realisierung des Dienstes `nearestObjects` beschrieben.

Selektionsmengen der Klienten

Der Entwurf des Systems sieht vor, dass der Server für jeden Klient die aktuelle Selektionsmenge räumlicher Objekte speichert, um bei neuen räumlichen Analyseanfragen auf diese aufsetzen zu können. Zur Umsetzung dieses Vorgehens müssten für jeden Klienten von allen Tabellen, die räumliche Objekte enthalten, Kopien der so genannten *Selection Bitmap* gespeichert werden. Die Selection Bitmap spezifiziert die selektierten Objekte einer Tabelle. Das Vorgehen scheitert jedoch daran, dass Avenue nicht in der Lage ist, beliebige Selection Bitmaps zu kopieren und zu speichern.

Daher verfügt der SMS über eine globale Variable, die für jeden Klienten eine Liste der zuvor gestellten und noch relevanten Analyseanfragen enthält. Analyseanfragen sind relevant, wenn nach ihnen keine Anfrage mit der Integrationssemantik *NewSet* formuliert wurde. Bei Eintreffen einer neuen Analyseanfrage führt der SMS alle relevanten Anfragen des Klienten inklusive der neu eingetroffenen Anfrage der Reihe nach aus. Auf diese Weise wird die Speicherung der Selektionsmengen simuliert.

Berücksichtigung der Vererbung

Der SMS verwaltet raumbezogene Objekte von Klassen, die in eine Vererbungshierarchie eingebettet sind. Da ArcView die Vererbung nicht unterstützt, werden die Objekte jeder Klasse in einer eigenen Tabelle gehalten (vgl. Kapitel 8.2.4). Der Zugriff auf die Objekte einer Klasse muss auch die Objekte jener Klassen zurückliefern, welche die spezifizierte Klasse erweitern. Daher werden bei Operationen auf Klassen alle Tabellen bestimmt, die Klassen verwalten, welche von der vorgegebenen Klasse erben. Die Vererbungshierarchie räumlicher Objekte ist dabei hilfreich.

Kommunikationsaspekte

In Bezug auf die Kommunikation wird hier zunächst erklärt, wie die Skripte des SMS ihre Ergebnisse an die Klienten übermitteln. Anschließend wird kurz auf zwei generelle Punkte eingegangen.

Die Kommunikationsprimitive, die in Avenue verfügbar sind, lassen dem Entwickler wenig Spielraum. Über den Aufruf der Methode `WebLink.The` erhält man Zugang zu der HTTP-Verbindung, die vom Klienten aufgebaut wurde. Die Methode `WriteReposeHeader` erlaubt die Spezifikation des Inhaltstyps der Antwort, wie zum Beispiel `text/html` oder `image/gif`. Mittels `WriteString` kann anschließend eine Zeichenkette zum Klienten übertragen werden. `WriteString` kann wiederholt aufgerufen werden. Dateien lassen sich durch einen Aufruf von `WriteFile` übertragen. Sie können auch in den Cache des Webservers geschrieben werden. Dazu wird die Methode `WriteCache` verwendet. An den Klienten wird im letzteren Fall die Adresse der Datei im Cache zurückgeliefert.

Um den Klienten über eingetretene Fehler beim Server zu informieren, führt der SMS den Inhaltstyp `text/servererror` ein. Dieser Typ signalisiert der API, dass ein Fehler eingetreten ist und dass der Inhalt der Nachricht als Fehlermeldung zu interpretieren ist. Die API erzeugt daraus eine Java-Ausnahme, die sie wirft und von der Klientenapplikation abzufangen ist.

Die Dienstsckripte übertragen Daten, die von der Applikation weiterverarbeitet werden sollen, als Zeichenkette. Falls das Ergebnis eines Dienstes zum Beispiel mehrere Bezeichner von Objekten sind, werden diese Bezeichner als Zeichenketten, die durch Leerzeichen voneinander getrennt sind, an den Klienten geliefert. Dazu wird `WriteString` verwendet.

Das Dienstsript `currentView` macht es erforderlich, Bilder in Dateiform an Klienten zu übermitteln. `currentVRMLModel` erfordert die Übertragung von Dateien. Beide Skripte verwenden die Option, Dateien in den Cache des Webservers zu schreiben. Damit ist ArcView kürzer blockiert, da an Klienten nur die Adresse der Datei übertragen wird.

Im Zusammenhang mit einer möglichen Realisierung der Communication DLL sei hier erwähnt, dass sie nicht in Standardjava der Version 1.2 codiert werden kann. Alleine das nicht-standardkonforme Microsoft-Java ermöglicht nämlich die Generierung von DLLs.

Es wird noch auf ein Problem des ArcView-IMS hingewiesen, das von ESRI bestätigt wird. Aus unerfindlichen Gründen kappt der IMS gelegentlich die Verbindung zum Webserver. ArcView kann dann von außen nicht mehr erreicht werden und folglich der SMS seine Dienste nicht mehr anbieten. Bei der bisherigen Arbeit mit dem ArcView-IMS ist dieser Fehler jedoch praktisch nie aufgetreten.

Der Dienst `nearestObjects`

Bei der Realisierung des Dienstes `nearestObjects` kann der ArcView-Mechanismus, der sich anbietet, nicht angewandt werden. ArcView ermöglicht einen *Spatial Join* zwischen Tabellen, die raumbezogene Objekte enthalten. Dieser bestimmt für alle Objekte der einen Tabelle das nächstgelegene Objekt der anderen Tabelle. Gegen die Verwendung des Spatial Joins sprechen drei Gründe:

- Er ermittelt nur das allernächste Objekt. Die Semantik von `nearestObjects` erfordert aber die Bestimmung von mehreren Objekten.
- Der Spatial Join arbeitet nur auf kompletten Tabellen, `nearestObjects` jedoch auf Selektionsmengen, die aus Objekten mehrerer Tabellen bestehen können.
- Bei Polygonen findet der Spatial Join heraus, welche Objekte innerhalb eines Objekts der anderen Tabelle liegen. In diesem Zusammenhang ist das nicht wünschenswert.

Daher verwendet das Dienstsript von `nearestObjects` einen *Brute Force*-Algorithmus, der die aktuelle Selektionsmenge des jeweiligen Klienten verarbeitet.

9.3 Implementierung der APIs

Im Folgenden werden Aspekte der Implementierung der Benutzer- und Management-APIs erläutert. Dabei wird als Erstes die Idempotenz von Operationen untersucht. Danach wird erklärt, welche Informationen in den APIs zur Steigerung der Performanz zwischengespeichert werden.

Idempotenz von Operationen

Wie schon in Kapitel 8.2 angesprochen, muss die Idempotenz von Operationen berücksichtigt werden. Bei Verwendung von RPCs würden sich diese um die hier besprochene Problematik kümmern. Die meisten Dienste können mehrfach aufgerufen werden, ohne dass die Integrität der Daten angegriffen wird. Sie sind also idempotent. Dazu gehört die Abfrage der Position eines räumlichen Objekts.

Operationen, die ein neues Objekt zu einem räumlichen Modell hinzufügen, dürfen jedoch nur einmal ausgeführt werden. Ansonsten könnten Objekte erzeugt werden, die es nicht „gibt“. Aus diesem Grund initiiert der InvokerCoordinator nur einmal den Communicator, um eine solche Operation beim SMS aufzurufen. Der Verlust dieser Nachricht kann einfach kompensiert werden.

Es ist jedoch möglich, dass der SMS das spezifizierte Objekt erzeugt, aber den Bezeichner des Objekts nicht mehr zum Klienten zurücksenden kann, da zum Beispiel die Verbindung unterbrochen wurde (vgl. Abb. 9.1). In diesem Fall ist die API bzw. die raumbezogene Applikation nicht befähigt, den Bezeichner des neu erzeugten Objekts in Erfahrung zu bringen.

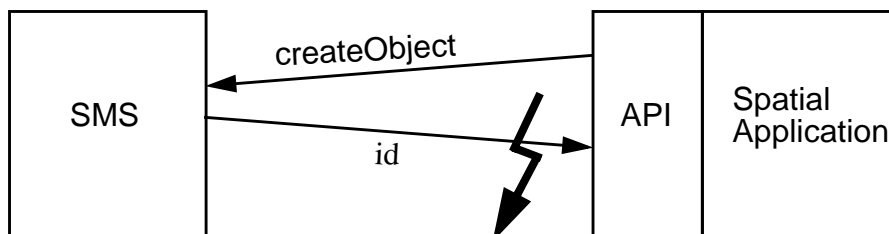


Abbildung 9.1: Szenario, indem die Antwort des SMS die API nicht erreicht

Um dieses Problem zu lösen, enthält jede Operation, die ein Objekt erzeugt, vom Klienten eine für ihn eindeutige Befehlsnummer zugewiesen. Im Beispiel in Abbildung 9.1 bedeutet dies, dass an die Nachricht `createObject` eine konkrete Zahl angefügt wird.

Der SMS speichert die Nummer der Sitzung des Klienten, die übermittelte Befehlsnummer sowie den Bezeichner des neu erzeugten Objekts in der Tabelle `SaveIDs`. Falls ein Klient nun den Objektbezeichner als Ergebnis einer Operation nicht erhält, so meldet er dies dem SMS. Dabei gibt er die Befehlsnummer des ursprünglichen Befehls an. Anhand dieser inspiziert der SMS die Tabelle `SaveID` und sendet dem Klienten den gespeicherten Objektbezeichner zu. Da Avenue keine Transaktionsmechanismen bereitstellt, funktioniert dieses Verfahren bei Systemabstürzen nicht unbedingt.

Zwischenspeicherung in der API

Ein weiterer Punkt aus dem Bereich Kommunikation, der angesprochen werden soll, ist die Übertragung von räumlichen Analyseanfragen an den SMS. Da möglicherweise nur das Ergebnis einer Sequenz von Anfragen interessiert, führt die Benutzer-API lokal eine Zwischenspeicherung aller Anfragen durch. Erst wenn sie explizit aufgefordert wird, überträgt die Benutzer-API alle angesammelten Anfragen auf einmal an den SMS. Dies mindert den Kommunikationsaufwand, da nicht für jede einzelne Anfrage eine HTTP-Verbindung aufgebaut werden muss.

Der SMS speichert lokal die Zustände aller Klienten. Falls sich der Zustand eines Klienten ändert, so ist es nötig, dass der Klient ihn darüber informiert. Damit nicht bei jeder Zustandsänderung eine Nachricht an den SMS übermittelt werden muss, führt auch hier die Benutzer-API eine Zwischenspeicherung durch. Bei der vorliegenden Implementierung des SMS genügt es, wenn der SMS bei einem Dienstaufwurf über den aktuellen Zustand des aufrufenden Klienten Bescheid weiß, da er zuvor nicht auf dessen Zustand zugreift. Daher ist es ausreichend, wenn die Benutzer-API die Werte, die den geänderten Zustand spezifizieren, beim nächsten Dienstaufwurf mitschickt. Die neuen Werte werden einfach an die URL, die den Dienst definiert, hinten angehängt. Das `ProcessStandardArguments`-Skript, das in Kapitel 8.2.3 eingeführt wurde, aktualisiert daraufhin den Zustand beim Server. Die Methode ähnelt dem *Piggybacking* in Rechnernetzen, bei dem Bestätigungen für eingegangene Nachrichten mit dem Versenden eigener Nachrichten übermittelt werden (siehe Tanenbaum, 1996).

9.4 Beispielapplikation

Zur Demonstration der Fähigkeiten des SMS wurde eine Beispielapplikation entwickelt. Diese nutzt einige der vom SMS angebotenen Dienste. Abbildung 9.2 zeigt die Benutzungsoberfläche der Applikation.

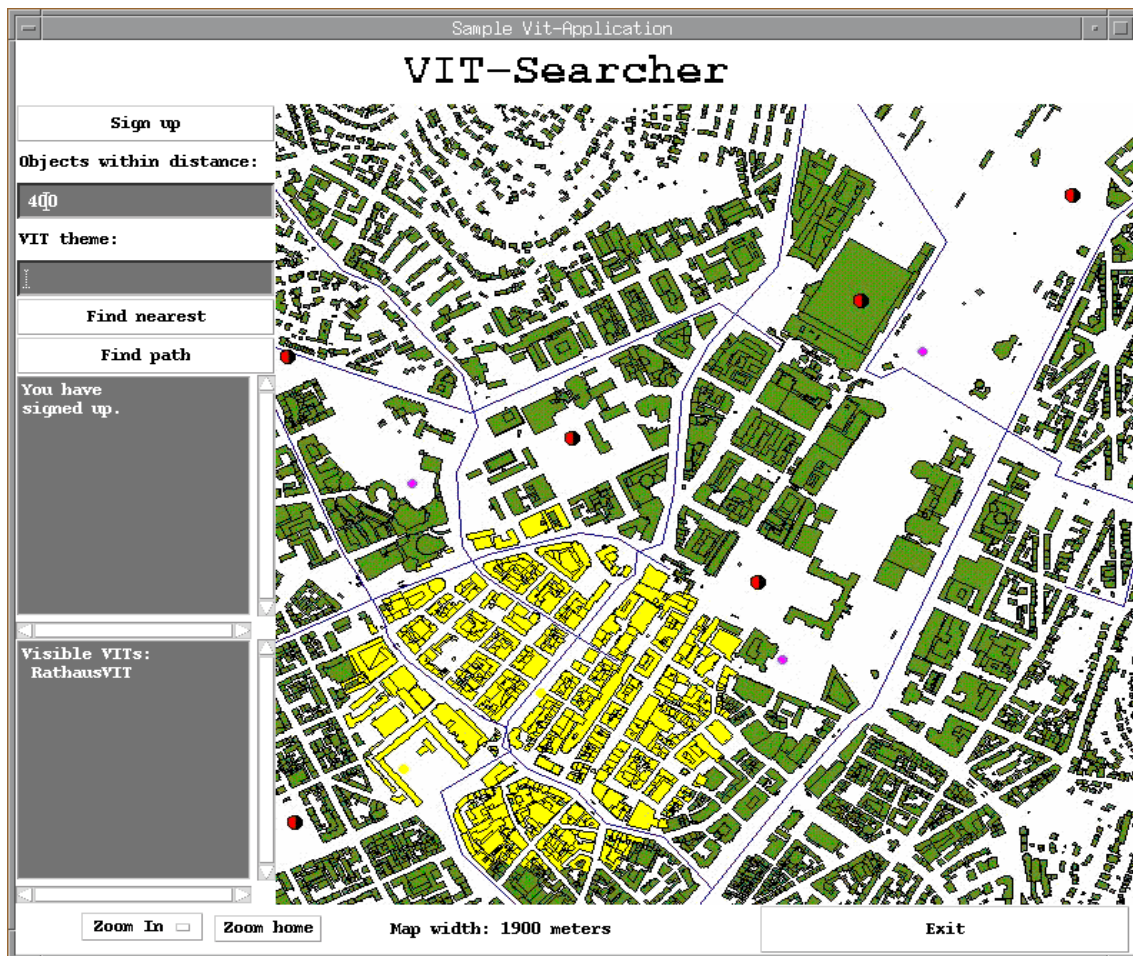


Abbildung 9.2: Benutzungsoberfläche der Beispielapplikation

Die prinzipielle Aufgabe der Applikation ist die Suche nach virtuellen Litfasssäulen, die bestimmte Kriterien erfüllen. In der Applikation werden diese Litfasssäulen *VIT* (für engl. *virtual information tower*) genannt und als schwarz-rot schraffierte Kreise dargestellt.

Nachfolgend wird erklärt, welche Fähigkeiten die Applikation besitzt. Die Realisierung der einzelnen Funktionen wird insofern angesprochen, als dass darauf eingegangen wird, welche Dienste des SMS genutzt werden.

Manipulation der Darstellung

Wie in der Abbildung zu sehen ist, zeigt die Applikation eine Karte des Modells der Innenstadt von Stuttgart und die darin enthaltenen VITs und mobilen Benutzer an. Es könnten aber auch beliebige andere Modelle angeboten werden. Unterhalb der Karte ist angegeben, welcher Entfernung die Breite des aktuell dargestellten Gebiets entspricht. Im Anfangszustand ist dies 1900 Meter. Durch Anklicken eines Punktes der Karte vergrößert oder verkleinert die Applikation den dargestellten Ausschnitt. Das hängt davon ab, in welchem Modus die Auswahl Taste ist, die sich am linken unteren Rand der Benutzungsoberfläche befindet. Der neue Mittelpunkt der Karte ist der angeklickte Punkt. Alternativ kann auf der Karte durch Ziehen der Maus mit gedrückter linker Maustaste auch ein Rechteck aufgespannt werden. Die Applikation stellt dann das kleinste Quadrat der Karte dar, die das Rechteck vollständig umfasst.

Beim Start der Applikation wird implizit mit dem Dienst `openSession` die Sitzung mit dem SMS eröffnet. Die Änderung des dargestellten Ausschnitts wird realisiert durch Aufrufe des Dienstes `setMapExtent`. Die Bilder werden vom Dienst `currentView` bereit gestellt.

Analyse des Gebiets

Über die Tasten und Textfelder auf der linken Seite der Benutzungsoberfläche kann das Gebiet analysiert werden. Die oberste Taste (*Sign Up*) meldet das mobile Objekt, das den Benutzer der Applikation repräsentiert, beim SMS an. Dabei wird dessen aktuelle Position mitgeliefert. Daraufhin wird eine Karte der Gegend angezeigt, in die das mobile Objekt als rosa Punkt eingezeichnet ist. Außerdem gibt das untere Textfeld die VITs an, in deren Gültigkeitsbereichen sich das Objekt befindet. Jeder VIT verfügt über ein kreisförmiges Gebiet, in dem er über die Nexus-Plattform wahrnehmbar ist.

Zur Realisierung dieser Funktionen werden die Dienste `registerMobileObject`, `getAttribute` und `objectsInArea` verwendet. `getAttribute` ermittelt die Radien der VITs. `objectsInArea` bestimmt anschließend, ob das mobile Objekt, das den Benutzer repräsentiert, im abgedeckten Umkreis der VITs befindet.

Durch Eingabe einer Entfernungsangabe in das Feld unterhalb der Angabe *Objects Within Distance* werden alle räumlichen Objekte selektiert, die sich innerhalb des Kreises mit dem mobilen Objekt, das dem Benutzer entspricht, als Mittelpunkt und der eingegebenen Entfernung in Metern als Radius befinden. In dem Beispiel aus Abbildung 9.2 wurde 400 Meter eingegeben. Die Objekte, welche die Bedingung erfüllen, werden in gelber Farbe in die Karte eingezeichnet.

Das nächste Textfeld erlaubt die Eingabe eines Themas. Dort können Begriffe eingegeben werden, die für den Benutzer nützliche Themen beschreiben, beispielsweise *Shopping*, *Sport* oder *Sightseeing*. Die Applikation gibt die Namen der VITs, die das spezifizierte Thema anbieten, im mittleren Textfeld an. Gleichzeitig werden die selektierten VITs gelb in die Karte eingezeichnet. Falls zuvor schon Objekte selektiert wurden, dann liefert diese Funktion nur VITs zurück, die zur alten Selektionsmenge gehörten und das angegebene Thema anbieten. Ähnlich regulären Ausdrücken kann zur Spezifikation des Themas auch ein Asterisk für eine willkürliche Zeichenkette angegeben werden.

Die Taste *FindNearest* bestimmt das bezüglich des mobilen Objekts nächstgelegene Objekt, welches sich in der aktuellen Selektionsmenge befindet.

Die letzten drei vorgestellten Funktionen werden mit Hilfe der räumlichen Analyseanfragen des SMS umgesetzt.

Die Taste *FindPath* bietet schließlich einen Navigationsdienst an. Nach dem Betätigen dieser Taste kann der Benutzer einen oder mehrere Punkte auf den Straßen anklicken. Die Applikation zeichnet den kürzesten Weg vom mobilen Objekt zu den spezifizierten Punkten ein. Falls ein Punkt selektiert wurde, der sich nicht in der Nähe einer Straße befindet, so wird eine Fehlermeldung generiert.

Zur Realisierung von *FindPath* wird der SMS-Dienst *findPath* verwendet.

Eine Aktivierung der *Exit*-Taste meldet das mobile Objekt beim SMS ab und beendet sowohl die Sitzung mit dem SMS als auch die Applikation.

Mit *unregisterMobileObject* wird das mobile Objekt des Benutzers dabei abgemeldet. Ein Aufruf von *closeSession* schließt die Sitzung mit dem SMS.

Damit endet die Betrachtung der Implementierung des Systems. Das nächste Kapitel fasst die Arbeit zusammen und gibt einen Ausblick.

10 Zusammenfassung und Ausblick

Zusammenfassung

Diese Arbeit befasste sich mit der Untersuchung und Realisierung eines Servers für räumliche Modelle. Der Server ist Teil der Nexus-Plattform. Eine Betrachtung verwandter Arbeiten ergab, dass dort eingesetzte Komponenten sich für den Server nicht eignen, da sie nur einen Bruchteil der gestellten Anforderungen erfüllen. Viele Ansätze besitzen beispielsweise keine räumlichen Analysefähigkeiten oder kennen nicht das Konzept des virtuellen Objekts.

Nach einer Einführung in GIS wurden die Randbedingungen festgehalten, die sich aus der Mobilität der Klienten ergeben. Im Wesentlichen sind dies die geringere Leistungsfähigkeit mobiler Rechner, deren endliche Energiequelle und relativ kleines Display, die geringe Verbindungsqualität und deren hohe Varianz. Anschließend wurden die Anforderungen an den Server mittels eines Use Case-Modells konkretisiert. Wichtige Forderungen sind dabei die Skalierbarkeit sowie die Berücksichtigung unterschiedlich leistungsfähiger Klienten. Die Definition der Dienste, die der Server anzubieten hat, komplementierte die Spezifikation der funktionalen Anforderungen. Die Dienste wurden dabei gruppiert, um die verschiedenen Arten von Klienten zu berücksichtigen.

Die Entwicklung der Lösung bestand zunächst im Entwurf einer konzeptionellen Architektur. Der Entwurf umfasste eine Beschreibung der generellen Komponenten des Servers sowie dessen Eingliederung in die Nexus-Plattform. Überdies wurden Gesichtspunkte der Föderation erörtert. Beziehungsverhältnisse zwischen räumlichen Modellen wurden in diesem Zusammenhang vorgestellt. Für die Dienste, die eine Föderation erfordern, wurde außerdem beschrieben, wie sie föderiert realisiert werden können. Schließlich wurde eine geeignete Kommunikationsinfrastruktur ausgewählt: RMI.

Die Umsetzung der konzeptionellen Architektur in ein konkretes System bestand zunächst in der Projektion in eine Architektur, die auf ArcView fundiert. Die ArcView-Konfiguration, die am schnellsten zum Ziel führt, basiert auf dem ArcView-Internet Map Server (IMS). Der IMS erfordert die Verwendung von HTTP zur Kommunikation. Diese Konfiguration ermöglicht eine einfache Umsetzung der Anforderungen an den Server und erlaubt kurze Entwicklungszeiten. Die auf dem ArcView-IMS aufbauende Architektur wurde erläutert, indem sowohl die eingesetzten Ave-

nueskripte als auch die erforderlichen Tabellen und globalen Variablen beschrieben wurden. Ferner wurde die Architektur der Benutzer-API stellvertretend für die gesamte Schnittstelle präsentiert.

Zum Abschluss der Arbeit wurden Aspekte der Implementierung behandelt. Dabei wurde auch angegeben, in welchem Umfang der Server implementiert wurde. Eine Applikation, die der Demonstration der Fähigkeiten des Servers dient, wurde des Weiteren vorgestellt.

Ausblick

Der implementierte Server besitzt prototypischen Charakter. Er nutzt die Funktionalität eines GIS und wurde nur partiell realisiert. Der Einsatz von ArcView hat einige Einschränkungen zur Folge. Die Kooperation mit anderen Instanzen muss beispielsweise mittels DLLs realisiert werden, was fehleranfällig ist. Ein weiterer Nachteil besteht darin, dass einige Funktionen nicht angeboten werden, wie räumliche Ereignisse. Deren Realisierung durch Avenueskripte ist aufgrund der angesprochenen Mängel von Avenue sehr umständlich. Ferner ist die Performanz des bestehenden Systems dürftig. Das schließt die Kommunikation über HTTP ein.

Da Nexus ein langfristig angelegtes Forschungsprojekt ist und der SMS eine essenzielle Komponente der Nexus-Plattform darstellt, scheint es angebracht, den SMS vollständig neu zu entwickeln. Damit muss keine Rücksicht auf Restriktionen bestehender GIS genommen werden. Weiterhin kann die konzeptionelle Architektur, die in dieser Arbeit beschrieben wurde, direkt umgesetzt werden. Die Vorteile der Objektorientierung sind dann konsequent auszunutzen. Komponenten, die weiter zu realisieren sind, sind die Föderation und das Area Service Register.

Weitergehende Untersuchungen sind außerdem nötig. Zum Beispiel ist zu untersuchen, welche räumlichen Dienste von Klienten benötigt werden. Diese Arbeit hat zwar Dienste definiert, die über eine Schnittstelle anzubieten sind. Ohne eine nähere Betrachtung der Bedürfnisse von Klienten ist dies jedoch schwierig.

Darüber hinaus ist ein Konzept zu entwickeln, das die Arbeitslast flexibler als in dieser Arbeit zwischen Server und Klient verteilt. Erstrebenswert ist eine automatische Aufteilung der Arbeit, die von der Leistungsfähigkeit des Servers, des Klienten und des verwendeten drahtlosen Netzwerks abhängt.

Für Demonstrationszwecke und vorläufige Untersuchungen ist der bestehende SMS jedoch geeignet. Er ermöglicht die Nutzung räumlicher Modelle, die für ArcView entwickelt wurden. Möglicherweise sollte er ausgebaut werden, um bisher nicht implementierte Funktionen demonstrieren zu können.

A Anhang

A.1 Glossar des Use Case-Modells

Föderation von Diensten. Die Föderation von Diensten bezeichnet die Verteilung von Diensten auf die SMS, welche die erforderlichen räumlichen Objekte verwalten, und die anschließende Generierung eines Resultats aus den Informationen, die von den SMS geliefert werden.

Geometrisches Attribut. Das geometrische Attribut legt die relative Lage und Form eines räumlichen Objekts fest.

Information Service. Der Information Service ist Teil der Nexus-Plattform. Er verwaltet den Informationsgehalt virtueller Objekte.

Location Service. Der Location Service stellt dem SMS Positionsinformationen über mobile Objekte bereit.

Mobiles Objekt. Mobile Objekte sind hauptsächlich Menschen, können aber theoretisch auch reale mobile Gegenstände der Welt sein, wie Autos oder Roboter.

Objektklassen. Raumbezogene Objekte sind Instanzen von Objektklassen. Zwischen Objektklassen können Vererbungsbeziehungen bestehen.

Raumbezogene Applikation. Die raumbezogene Applikation ist die Repräsentantin der Applikationen, welche die Nexus-Plattform nutzen. Sie setzt typischerweise auf mobile Rechner auf und kommuniziert mit der Nexus-Plattform drahtlos.

Raumbezogener Klient. Der raumbezogene Klient ist ein Oberbegriff für die raumbezogene Applikation und für den Information Service.

Räumliches Modell. Ein räumliches Modell wird durch die Eckpunkte des Gebiets, die verfügbaren Detaillierungsgrade und die vorhandenen Objektklassen spezifiziert. Ein SMS kann mehrere räumliche Modelle verwalten.

Räumliches Objekt. Der Oberbegriff für statische, mobile und virtuelle Objekte ist räumliches Objekt. Ein räumliches Objekt besitzt eine räumliche Position, Form, Attribute und einen global eindeutigen Bezeichner. Es kann über eine Schnittstelle manipuliert werden.

Räumliche Position. Die räumliche Position eines raumbezogenen Objekts wird durch dreidimensionale Koordinaten spezifiziert. Zur Unterstützung von Positionierungssystemen in der freien Natur werden geografische Koordinaten in WGS-84-Format und Gauss-Krueger-Format verwendet. Innerhalb Gebäude werden lokale Koordinatensysteme unterstützt.

Spatial Model Server (SMS). Die Server-Komponente, die ein oder mehrere räumliche Modelle der Nexus-Plattform verwaltet, heißt Spatial Model Server.

SMS-Manager. Der SMS-Manager verwaltet die räumlichen Modelle eines SMS.

Statisches Objekt. Statische Objekte sind Gegenstände der Natur (wie Berge, Flüsse, Meere, Bäume) sowie künstliche Gegenstände (wie Straßen, Gebäude, Brücken). Sie sind immobil und verfügen über Form und Ausdehnung.

Thematisches Attribut. Thematische Attribute von räumlichen Objekten sind dessen nichtgeometrischen Sachdaten.

Topologisches Attribut. Nachbarschaftsverhältnisse zwischen räumlichen Objekten werden von topologischen Attributen der Objekte definiert.

Virtuelles Objekt. Virtuelle Litfasssäulen oder Post-It-Notizen sind beispielsweise virtuelle Objekte. Sie beinhalten Informationen, die einer räumlichen Position zugeordnet wird. Falls ein Objekt sich in einem wohldefinierten Bereich um die Position befinden, können ihm die Informationen über eine normalerweise drahtlose Verbindung übermittelt werden

A.2 Definition der Benutzer-API

A.2.1 Hinzufügen und Entfernen räumlicher Objekte

Klassen

- **Attribute** repräsentiert ein Sachdatum von räumlichen Objekten.

- AttrCond spezifiziert eine Anfrage auf den thematischen Attributen von räumlichen Objekten
- ObjectId repräsentiert den Bezeichner eines räumlichen Objekts. Diese Klasse stammt aus dem Paket `nexus.ls`.
- LSId ist der Bezeichner eines Location Service. Er ist ebenfalls aus dem Paket `nexus.ls`.
- HSId ist der Bezeichner eines Home Servers.
- ObjClass spezifiziert die Attribute und die hierarchische Einordnung einer Klasse räumlicher Objekte.

Dienste

- `void registerMobileObject(ObjectId objectId, LSId locServ, HSId homeServ, boolean isAccessible)`
- `void registerMobileObject(ObjectId objectId, LSId locServ, HSId homeServ, boolean isAccessible, Attribute cacheAttrs[])` Die Attribute in `cacheAttrs` werden vom SMS in einem Cache zwischengespeichert.
- `void unregisterMobileObject(ObjectId objectId)`
- `ObjectId createVirtualObject(ObjectId creator, ObjClass objClass, Attribute attribute[], Object geometry)` Die Geometrie hat die Klasse `Object`, da es in `nexus.location` keine gemeinsame Oberklasse aller geometrischen Formen gibt.
- `void destroyVirtualObject(ObjectId remover, ObjectId objectId)`

A.2.2 Räumliche Analyse

Klassen

- `Area` ist die Oberklasse räumlicher Gebiete wie `Polygon` oder `Kreis`. Sie stammt aus dem Paket `nexus.location`.
- `Point` und `Line` sind ebenfalls aus dem Paket `nexus.location`. Diese Klassen repräsentieren Punkte bzw. Geraden. Sie sind keine Unterklassen von `Area`.
- Auch `CoordSystem` wird von `nexus.location` bereitgestellt. Eine Instanz dieser Klasse definiert ein Koordinatensystem.

Dienste

- `ObjectId[] objectsInArea(Area area, int inclusionType, int integrStrategie)` *inclusionType* und *integrStrategie* werden Konstanten zugewiesen, die die in Kapitel 6.2.2 vorgestellten Werte bereitstellen.
- `ObjectId[] objectsSatCond(AttrCond attrCond, int integrStrategie)`
- `ObjectId[] neighborsFrom()`
- `ObjectId[] nearestObjects(Point point, int number)`
- `Point locationOfObject(ObjectId objectId)`
- `Object geometryOfObject(ObjectId objectId)`
- `void setCoordSystem(CoordSystem coordSystem)`
- `ObjectId objectInDirection(Point point, Line line)`
- `long distanceTo(Point pointA, Point pointB)`
- `long sizeOfObject(ObjectId objectId)`
- `ObjectId[] objectsAroundSelected(long distance)`
- `int quantityOfSelected()`

A.2.3 Navigation

Dienste

- `String findPathExpl(Point points[], boolean bestOrder, AttrCond constraints)`
- `Point[] findPathCorners(Point points[], boolean bestOrder, AttrCond constraints)`
- `Image findPathImage(Point points[], boolean bestOrder, AttrCond constraints)`

A.2.4 Abfragen von Sachdaten

Dienste

- `Object getAttribute(ObjectId objectId, String attrName)`
- `String[] attributeNames(ObjectId objectId)`
- `ObjClass classOfObject(ObjectId objectId)`

A.2.5 Manipulation räumlicher Objekte

Dienste

- void setLocation(ObjectId objectId, Point point, ObjectId manipulator)
- void setGeometry(ObjectId objectId, Object geometry, ObjectId manipulator)
- void setAttribute(ObjectId objectId, Attribute attribute, ObjectId manipulator)

A.2.6 Räumliche Ereignisse

Klassen

- EventId spezifiziert ein räumliches Ereignis. Die Klasse ist Teil des Pakets `nexus.ls`.

Dienste

- void subscribeSpatialEvent(EventId eventId)
- void cancelEventSubscription(EventId eventId)

A.2.7 Grafische Repräsentationen

Klassen

- Image repräsentiert ein Bild und wird vom Paket `java.awt` zur Verfügung gestellt.
- File repräsentiert eine Datei und kommt aus dem Paket `java.io`.

Dienste

- Image currentView(String classNames[], boolean zoomToSelected)
- Image current3DView(String classNames[], boolean zoomToSelected, Point point)
- File currentVRMLModel(String classNames[]) Diese Operation liefert die Hauptdatei des VRML-Modells zurück. In dieser Datei wird auf die weiteren Dateien verwiesen, die auch zu diesem Modell gehören. VRML-Modelle werden in mehreren Dateien gespeichert (siehe Carey und Bill, 1997).
- Point[] getMapExtent()
- void setMapExtent(Point point[])

- void setImageSize(long width, long height)
- void setLevelOfDetail(int number) *number* ist eine Zahl, die einen Detaillierungsgrad festlegt.

A.3 Definition der Management-API

Dienste

- ObjectId createStaticObject(ObjClass objClass, Attribute attribute[], Object geometry)
- void destroyStaticObject(ObjectId objectId)
- ObjectId createMobileObject(objClass, attribute[], geometry)
- void destroyMobileObject(objectId)
- void defineVirtObjClass(ObjClass objClass)
- void setStatObjLocation(ObjectId objectId, Point point)
- void setStatObjGeometry(ObjectId objectId, Object geometry)
- void createSpatialModel(String name, Object area, ObjClass objClass[])

A.4 Implementierungsumfang

Die folgenden Dienste wurden im Rahmen dieser Arbeit implementiert:

Hinzufügen und Entfernen räumlicher Objekte

- void registerMobileObject(ObjectId objectId, LSId locServ, HSId homeServ, boolean isAccessible)
- void registerMobileObject(ObjectId objectId, LSId locServ, HSId homeServ, boolean isAccessible, Attribute cacheAttrs[])
- void unregisterMobileObject(ObjectId objectId)

- `ObjectId createVirtualObject(ObjectId creator, ObjClass objClass, Attribute attribute[], Object geometry)`
- `void destroyVirtualObject(ObjectId destructor, ObjectId objectId)`

Räumliche Analyse

- `ObjectId[] objectsInArea(Area area, int integrStrategie)` Der Inklusionstyp ist immer *beliebiges Schneiden*.
- `ObjectId[] objectsSatCond(AttrCond attrCond, int integrStrategie)`
- `ObjectId[] nearestObjects(Point point, int number)`
- `Point locationOfObject(ObjectId objectId)`
- `Object geometryOfObject(ObjectId objectId)`
- `long distanceTo(Point pointA, Point pointB)`

Navigation

- `Image findPathImage(Point points[])`

Abfragen von Sachdaten

- `Object getAttribute(ObjectId objectId, String attrName)`
- `String[] attributeNames(ObjectId objectId)`
- `ObjClass classOfObject(ObjectId objectId)`

Manipulation räumlicher Objekte

- `void setLocation(ObjectId objectId, Point point)` Die Rechte des Dienstaufrufers werden nicht überprüft.
- `void setGeometry(ObjectId objectId, Object geometry)`
- `void setAttribute(ObjectId objectId, Attribute attribute)`

Grafische Repräsentationen

- `Image currentView(String classNames[], boolean zoomToSelected)`
- `File currentVRMLModel(String classNames[])`
- `void setMapExtent(Point point[])`

- void setImageSize(long width, long height)
- void setLevelOfDetail(int number) Dieser Dienst hat keine Auswirkung auf die Darstellungsweise räumlicher Objekte.

Management-API

- void defineVirtObjClass(ObjClass objClass)
- ObjectId createMobileObject(objClass, attribute[], geometry)
- void destroyMobileObject(objectId)

Literatur

Abowd et al. (1997):

Gregory D. Abowd, Christopher G. Atkeson, Jason Hong, Sue Long, Rob Kooper, Mike Pinkerton

Cyberguide: A Mobile Context-Aware Tour Guide

Baltzer/ACM Wireless Networks, Band 3 (5), Seiten 421 - 433, 1997

Baumann (1998):

Joachim Baumann

Rechnernetze II

Skript zur gleichnamigen Vorlesung, Fakultät Informatik, Universität Stuttgart, Juli 1998

Beadle et al. (1997):

H.W. Peter Beadle, B. Harper, G.Q. Maguire Jr., J. Judge

Location Aware Mobile Computing

Proceedings of the IEEE/EEE International Conference on Telecommunications (ICT '97),
Seiten 1319 - 1324, Melbourne, Australien, April 1997

Benelli et al. (1997):

Giuliano Benelli, Alberto Bianchi, Jonathan Broadbent, Patrizia Marti

The Design of New Tools, Using Wireless Technology for the Navigation in Physical Spaces

Technischer Bericht Nr. 24/97, Dipartimento di Ingegneria dell'Informazione, Università
Degli Studi Di Siena, Italien, 12. Dezember 1997

Bill und Fritsch (1997):

Ralf Bill, Dieter Fritsch

Grundlagen der Geo-Informationssysteme — Hardware, Software und Daten

Band 1, 3. Auflage, Wichmann Verlag, Heidelberg, 1997

Brown (1996):

P.J. Brown

The Stick-e Document: A Framework for Creating Context-Aware Applications

Electronic Publishing, Band 9 (1), Seiten 1 - 14, September 1996

Brown (1998):

Peter Brown

Some Lessons for Location-Aware Applications

Proceedings of the First Workshop on HCI for Mobile Devices, Glasgow University, Glas-
gow, Schottland, Mai 1998

Carey und Bill (1997):

Rikk Carey und Gaven Bill

The Annotated VRML 2.0 Reference Manual

Addison-Wesley, Reading, MA, USA, u.a., 1997

Dana (1998):

Peter H. Dana

The Geographer's Craft

www.utexas.edu/depts/grg/gcraft/contents.html, Department of Geography, University of Texas at Austin, Austin, TX, USA, April 1998

Fitzke et al. (1997):

Jens Fitzke, Claus Rinner, Dirk Schmidt

GIS-Anwendungen im Internet

GIS Geo-Informationssysteme — Zeitschrift für raumbezogene Informationen und Entscheidungen, 6/1997

Fitzmaurice (1993):

George W. Fitzmaurice

Situated Information Spaces and Spatially Aware Palmtop Computers

Communications of the ACM, Band 36, Ausgabe 7, Seiten 39 - 49, Juli 1993

Forman und Zahorjan (1994):

George H. Forman, John Zahorjan

The Challenges of Mobile Computing

IEEE Computer, Band 27, Ausgabe 4, Seiten 38 - 49, April 1994

Fritz (1999):

Andreas Fritz

Positionsabhängiger Zugriff auf WWW-Inhalte

Diplomarbeit Nr. 1709, Fakultät Informatik, Universität Stuttgart, Februar 1999

Gellersen (1999):

Hans-Werner Gellersen

Handheld/Wearable CSCW — „tragbare“ Kooperationsunterstützung

Vortrag im Informatik-Kolloquium, Fakultät Informatik, Universität Stuttgart, 27. April 1999

Halsall (1996):

Fred Halsall

Data Communication, Computer Networks and Open Systems

Vierte Auflage, Addison-Wesley, Reading, MA, USA, u.a., 1996

Hascher (1999):

Wolfgang Hascher

Funk statt Kabel

Elektronik 9/1999

Hohl (1999):

Fritz Hohl

Nexus — Benötigte Schnittstellen

Internes Dokument, Fakultät Informatik, Universität Stuttgart, 26. Februar 1999

Hohl et al. (1999):

Fritz Hohl, Uwe Kubach, Alexander Leonhardi, Kurt Rothermel, Markus Schwehm

Next Century Challenges: Nexus — An Open Global Infrastructure for Spatial-Aware

Applications

T. Imielinski, M. Steenstrup (Herausgeber): Proceedings of the Fifth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'99), Seiten 249 - 255, Seattle, WA, USA, 15. - 20. August, 1999

Imielinski und Navas (1999):

Tomasz Imielinski und Julio C. Navas

Geographic Addressing, Routing, and Resource Discovery with the Global Positioning System

Communications of the ACM, 1999

IrDA (1998):

Infrared Data Association (IrDA)

A Wireless Connectivity Technologies Comparison — Infrared and Radio Frequency

White Paper, IrDA, Walnut Creek, CA, USA, 10. September 1998

Jacobson et al. (1999):

Ivar Jacobson, Grady Booch, James Rumbaugh

The Unified Software Development Process

Grady Booch, Ivar Jacobson, James Rumbaugh (Herausgeber der Serie): Object Technology Series, Addison-Wesley, Reading, MA, USA, u.a., 1999

Leonhardi (1999):

Alexander Leonhardi

Architektur eines Prototyps für die Nexus-Plattform

Internes Dokument, Fakultät Informatik, Universität Stuttgart, 14. Mai 1999

Leonhardi und Kubach (1999):

Alexander Leonhardi und Uwe Kubach

An Architecture for a Universal, Distributed Location Service

European Wireless '99 Conference, München, 1999

Leonhardt (1998):

Ulf Leonhardt

Supporting Location-Awareness in Open Distributed Systems

Dissertation, Department of Computing, Imperial College London, UK, Mai 1998

Malaka (1999)

Rainer Malaka

Deep Map (GIS)

www.villa-bosch.de/eml/englisch/projekte/deepmap/deepgis.html, Januar 1999

Mann (1998):

Steve Mann

„Wearable Computing“ as Means for Personal Empowerment

Eingeladener Vortrag der 1998 International Conference on Wearable Computing, Fairfax, VA, USA, 12. Mai 1998

Not et al. (1998):

E. Not, M. Sarini, O. Stock, C. Strapparava, M. Zancanaro

Information Adaptation for Physical Hypernavigation

i3 Annual Conference, Nyborg, Juni 1998

Open GIS Consortium (1998):

Open GIS Consortium

The Open GIS Specification Model — Topic 12: The Open GIS Service Architecture

Arbeitspapier, Version 3, Open GIS Project Dokument Nummer 98-112, Wayland, MA, USA, 1998

Orfali und Harkey (1997):

Robert Orfali und Dan Herkey

Client/Server Programming with Java and CORBA

John Wiley & Sons, New York, NY, USA, 1997

Pascoe (1997):

Jason Pascoe

The Stick-e Note Architecture: Extending the Interface Beyond the User

Johanna Moore, Ernest Edmonds, Angel Puerta (Herausgeber): 1997 International Conference on Intelligent User Interfaces, Seiten 261 - 264, ACM Order Department, PO Box 12114, Church Street Station, New York, NY 10257, USA, Januar 1997

Rothermel (1997):

Kurt Rothermel

Grundlagen der Verteilten Systeme

Skript zur gleichnamigen Vorlesung, Fakultät Informatik, Universität Stuttgart, Juli 1997

Ryan et al. (1998):

N.S. Ryan, J. Pascoe, D.R. Morse

FieldNote: Extending a GIS into the Field

J.A. Barcelo, I. Briz, A. Vila (Herausgeber): New Techniques for Old Times — Computer Applications in Archaeology, 1998

Ryan et al. (1999):

N. S. Ryan, D. R. Morse, J. Pascoe

FieldNote: A Handheld Information System for the Field

R. Laurini (Herausgeber): Proceedings of the First International Workshop on Tele-Geo-Processing (TeleGeo '99), Lyon, Frankreich, Mai 1999

Salber und Abowd (1998):

Daniel Salber und Gregory D. Abowd

The Design and Use of a Generic Context Server

Proceedings of the Perceptual User Interfaces Workshop, Seiten 63 - 66, November 1998

Salber et al. (1999):

Daniel Salber, Anind K. Dey, Gregory D. Abowd

The Context Toolkit: Aiding the Development of Context-Enabled Applications

Proceedings of CHI'99, Pittsburgh, PA, USA, ACM Press, Mai 1999

Satyanarayanan (1996):

M. Satyanarayanan

Fundamental Challenges in Mobile Computing

Proceedings of the Fourteenth Annual ACM Symposium on Principles of Distributed Com-

puting, Seiten 1 - 7, Philadelphia, PA, USA, 1996

Smailagic und Martin (1997):

Asim Smailagic und Richard Martin

Metronaut: A Wearable Computer with Sensing and Global Communication Capabilities

Technischer Bericht, Engineering Design Research Center, Carnegie Mellon University, Pittsburgh, PA, USA, 1997

Spero (1994):

Simon E. Spero

Analysis of HTTP Performance Problems

www.w3.org/Protocols/HTTP/1.0/HTTPPerformance.html, Juli 1994

Spohrer (1998):

Jim Spohrer

WorldBoard — What Comes After the WWW?

www.worldboard.org, Mai 1998

Stahl (1997):

Roland Stahl

GIS in Internet und Intranet

gio.uni-muenster.de, geoinformatik_online, Ausgabe 2/1997

Swain (1994):

Robert S. Swain

UMTS — A 21st Century System

Whitepaper, RACE Mobile Project Line Assembly, RSS Telecon LTD., Suffolk, UK, 30. September 1994

Tanenbaum (1996):

Andrew Tanenbaum

Computer Networks

3. Auflage, Prentice Hall, Englewood Cliffs, NJ, USA, 1996

Tscheuschner (1997):

Eberhard Tscheuschner

ESRI heute: GIS im Zeichen des technologischen Wandels

gio.uni-muenster.de, geoinformatik_online, Ausgabe 2/1997

Wap Forum (1999):

Wap Forum

Wireless Application Protocol

www.wapforum.org, September 1999

Worboys (1995):

Michael F. Worboys

GIS — A Computing Perspective

Taylor & Francis, Philadelphia, PA, USA, 1995

Wünsche (1999):

Wolfgang Wünsche

Detailentwurf eines allgemeinen Location-Service und Erstellung eines einfachen Prototyps

Diplomarbeit Nr. 1759, Fakultät Informatik, Universität Stuttgart, August 1999

Produktreferenzen

Breezecom (1998):

Breezecom Corporation

BreezeNet - Pro series — Wireless Ethernet

www.breezecom.com, Breezecom Corporation, USA, 1998

ESRI (1996a):

ESRI

ArcView® GIS — The Geographic Information System for Everyone

Dokumentation, Environmental Systems Research Institute, Inc., Redlands, CA, USA, 1996

ESRI (1996b):

ESRI

ArcView® Internet Map Server — Map Publishing on the Web

Dokumentation, Environmental Systems Research Institute, Inc., Redlands, CA, USA, 1996

ESRI (1996c):

ESRI

Avenue — Customization and Application Development for ArcView® GIS

Dokumentation, Environmental Systems Research Institute, Inc., Redlands, CA, USA, 1996

ESRI (1996d):

ESRI

Building Applications with MapObjects

Dokumentation, Environmental Systems Research Institute, Inc., Redlands, CA, USA, 1996

ESRI (1996e):

ESRI

MapObjects Internet Map Server — Using MapObjects on the Internet

Dokumentation, Environmental Systems Research Institute, Inc., Redlands, CA, USA, 1996

ESRI (1998a):

ESRI

Getting Started With SDE

Dokumentation, Environmental Systems Research Institute, Inc., Redlands, CA, USA, 1998

ESRI (1998b):

ESRI

Spatial Database Engine

White Paper, Environmental Systems Research Institute, Inc., Redlands, CA, USA, April 1998

Lucent (1999):

Lucent Technologies

WaveLan Wireless Solutions — Wireless Technology to Support a World on the Go

www.wavelan.com, Lucent Technologies, USA, 16. März 1999

Microsoft (1999):

Microsoft Corporation

Windows CE — Home Page

www.microsoft.com/windowsce, Microsoft Corporation, USA, 1999

3Com (1999):

3Com Corporation

Palm Computing Platform — Development Zone

www.palm.com/devzone/30overview.html, 3Com Corporation, USA, 1999

Erklärung

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

Stuttgart, den 7. Oktober 1999