Modelle und Speicherungsstrukturen für Kontextinformationen

Seminararbeit

Im Rahmen des Seminars
„Mobile und kontextbewusste Datenbank-Technologien und Anwendungen“

Ansgar Lamersdorf

Betreuer: Philipp Dopichaj

1. Einleitung

Eine Anforderung an mobile intelligente Systeme ist *Kontextbewusstsein*. Das bedeutet, dass ein derartiges System abhängig vom Kontext, in dem es sich befindet, unterschiedliches Verhalten zeigt. Als Kontext zählt dabei jede Eigenschaft der Umgebung, die für das Verhalten des Systems von Bedeutung ist. Ein Beispiel für ein solches System ist ein Touristeninformationssystem, welches abhängig von der Position des Nutzers historische Informationen zu seinem aktuellen Standpunkt liefert. In diesem Fall ist der Kontext der Anwendung die aktuelle Position. Aber auch andere Daten wie etwa die Temperatur oder die in der Nähe befindlichen Objekte können als Kontext von Bedeutung sein.

Kontextbewusste Anwendungen müssen die Möglichkeit haben, Kontextdaten zu speichern und abzurufen. Hierzu muss ein Kontextmodell existieren, welches die relevanten Kontexteigenschaften mit ihren Wertebereichen festlegt.

Zur effizienten Speicherung und Anrufung der Daten, etwa um alle Objekte mit bestimmten Kontexteigenschaften zu finden, müssen entsprechende Zugriffsmöglichkeiten vorhanden sein. Analog zu den Zugriffsstrukturen in herkömmlichen Datenbanken geschieht dies vor allem über Indizes, so dass eine kontextbewusste Datenbank Indizes für Kontexteigenschaften ermöglichen muss. Dies ist vor allem dann ein Problem, wenn die Kontexteigenschaften mehrdimensional (wie etwa eine Position im Raum) sind.

In dieser Arbeit sollen verschiedene Ansätze und Entwicklungen auf dem Gebiet der Kontextmodelle und Speicherungsstrukturen für Kontextinformationen vorgestellt werden.

1.1 „Ort“ als Kontext

Zunächst muss dazu der Begriff „Kontext“ genauer eingegrenzt werden. Wie in vorherigen Beiträgen dieses Seminars genauer definiert, gibt es viele verschiedene Kontexteigenschaften, die für eine kontextbewusste Anwendung von Bedeutung sind. In [3] werden sie in zwei Hauptklassen eingeteilt:

- Persönlicher Kontext des Anwenders (Ort, Interessen, Budget, Bedürfnisse…)
- Kontext der Umgebung (Uhrzeit, Wetter, Verkehrsinformationen)

Für jede dieser Kontexteigenschaften entstehen unterschiedliche Anforderungen und Probleme bei der Erstellung passender Modelle und Speicherungsstrukturen. Beispielsweise ist die Speicherung der Uhrzeit nicht sehr aufwändig, da sie schon als modellhaftes, einwertiges Attribut vorliegt und auch in einer normalen Datenbank abgespeichert und indiziert werden kann. Die Modellierung der Interessen eines Anwenders ist dagegen sehr viel schwieriger, da zunächst einmal Wertebereiche eindeutig definiert werden müssen. Aus diesem Grund beschränkt sich diese Arbeit überwiegend auf eine einzelne Kontexteigenschaft, für die Modelle und Datenstrukturen vorgestellt werden. Als diese Eigenschaft wurde der Ort, also die räumlichen Eigenschaften des Kontextes ausgewählt. Diese Auswahl ist nicht willkürlich, sondern basiert auf einer Reihe von Gründen:

- In den meisten kontextbewussten Anwendungen ist der Ort die wichtigste oder sogar die einzige berücksichtigte Kontexteigenschaft, so etwa in [3] und [15].
- Viele kontextbewusste Anwendungen sind mobile Anwendungen, die sich auf ihren aktuellen Ort einstellen, den Ort also als Kontexteigenschaft benutzen.
- Die räumliche Position eines Objektes ist (etwa im Vergleich zu den Interessen des Anwenders) eine vergleichsweise gut zu modellierende Eigenschaft (beispielsweise durch die Angabe von geometrischen Koordinaten).
- Andererseits sind räumliche Eigenschaften so komplex, dass sie zur effizienten Nutzung in Datenbanken eigene Speicherungs- und Indexstrukturen benötigen und nicht einfach mit herkömmlichen Mitteln verwaltet werden können (wie beispielsweise Temperatur oder Budget).
- Räumliche Eigenschaften sind nicht nur in Rahmen von kontextbewussten oder mobilen Anwendungen von Bedeutung, sondern auch in anderen Bereichen (wie etwa bei Geoinformationssystemen), so dass viele verschiedene Forschungsansätze auf diesem Gebiet vorhanden sind.

1.2 Inhalt der Arbeit

Im weiteren Verlauf gliedert sich der Text in drei Teile:

Im zweiten Abschnitt werden verschiedene Arten von Kontextmodellen vorgestellt. Hierbei gibt es zwei verschiedene Ansätze (geometrisch und hierarchisch), die einander gegenübergestellt werden, aber auch Mischformen und weitergehende Modelle.

Abschnitt 3 beschäftigt sich mit verschiedenen Indexstrukturen zur Speicherung von räumlichen Informationen.

Der letzte Abschnitt präsentiert zwei unterschiedliche Systeme zur Anwendung und Speicherung von Kontextmodellen. Zum einen wird der Standard OpenGIS vorgestellt, der zur Verarbeitung geografischer Daten dient. Darauf folgt eine Einführung in Nexus, ein komplexes System, welches auf der Basis eines räumlichen Weltmodells verschiedene Kontextdaten speichern und anwenden kann.

2. Kontextmodelle

Eine kontextbewusste Anwendung, die Daten wie etwa räumliche Informationen speichern und verarbeitet, muss über ein *Kontextmodell* verfügen, welches festlegt, welche Kontextdaten gespeichert werden, wie sie gespeichert und abgerufen werden und wie Kontextdaten untereinander und mit anderen Daten verknüpft werden können. Solche Modelle werden in diesem Abschnitt vorgestellt. Dabei werden zuerst zwei grundlegende Arten von Kontextmodellen (geometrische und hierarchische Modelle) gegenübergestellt. Daraufhin werden einzelne Bespiele für Kontextmodelle präsentiert. Im Schluss dieses Abschnittes werden Möglichkeiten vorgestellt, Kontextmodelle mit zusätzlichen Objekten anzureichern.

2.1 Geometrische vs. hierarchische Kontextmodelle

Grundsätzlich gibt es zwei verschiedene Arten, ein räumliches Kontextmodell aufzubauen (siehe [4] und [11]): Geometrische (metrische) und hierarchische (symbolische, topologische).

Geometrische Kontextmodelle modellieren räumliche Informationen in (zwei- oder dreidimensionalen) Koordinatensystemen. Jeder Ort und jedes Objekt können in diesen Modellen durch Angabe von einem oder mehreren Punkten gespeichert werden. Dadurch können etwa einfache Punkte, räumliche Gebiete oder auch komplexe Körper exakt modelliert werden. Ein geometrisches Kontextmodell kann auch aus mehreren Koordinatensystemen bestehen, die mithilfe relativ einfacher mathematischer Operationen ineinander überführt werden können. Abfragen, wie etwa das Enthaltensein eines Objektes in einem andern Objekt oder Gebiet, oder die Entfernung zwischen zwei Orten können ebenfalls durch einfache geometrische Berechnungen beantwortet werden.

Ein Beispiel für ein geometrisches Kontextmodell ist das GPS-System, wie es zum Beispiel bei einem Navigationssystem eingesetzt wird. Es benutzt als Koordinatensystem die Längen- und Breitengrade der Erde sowie die Höhe über dem Meeresspiegel. Durch Abgleich der Daten verschiedener Satelliten kann das Navigationssystem die Koordinaten seines aktuellen Ortes feststellen. Diese können dann mit den gespeicherten Koordinaten von Straßen verglichen werden. Daraus kann das System nun die aktuell befahrene Straße und die weitere Weganweisung berechnen.

Hierarchische Kontextmodelle modellieren Orte als abstrakte Strukturen, die selber weitere Mengen von Orten enthalten können. Jedes Objekt wird einem oder mehreren Orten zugeordnet, die wiederum übergeordneten Orten zugeordnet sind. Beispielsweise gehört die Postadresse „Paul-Ehrlich-Straße 1, 67653 Kaiserslautern, Germany“ zu einem hierarchischen Kontextmodell, indem diese Adresse der Hausnummer 1 zugeordnet ist, die dem Ort „Paul-Ehrlich-Straße“ zugeordnet ist, die wiederum zum Ort „67653 Kaiserslautern“ gehört, der selber schließlich in dem Ort „Germany“ enthalten ist.

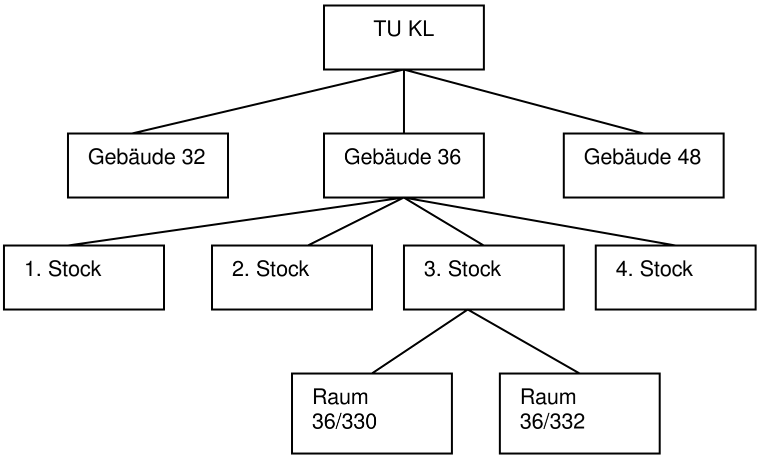

Fig. 1. Beispiel eines hierarchischen Kontextmodells

Abhängig vom jeweiligen Modell, können Orte zu einem oder mehreren übergeordneten Orten gehören. Im letzteren Fall überschneiden sich dann die übergeordneten Orte. Hierarchische Kontextmodelle ergeben also einen Baum (falls Orte sich nicht überlappen) oder einen gerichteten azyklischen Graph, die die Enthaltungs-Beziehungen zwischen den Orten angeben. Die Anwesenheit eines Objektes an einem bestimmten Ort ist in diesen Modellen sehr leicht überprüfbar, indem einfach festgestellt werden kann, ob das Objekt diesem Ort direkt oder indirekt untergeordnet ist. Dagegen können geometrische Abfragen, wie etwa die Entfernung zwischen zwei Orten, oft nicht beantwortet werden. Abbildung 1 zeigt als Beispiel eines hierarchischen Modells die Modellierung eines Gebäudes.
Hierarchische und geometrische Kontextmodelle haben jeweils verschiedene Vor- und Nachteile:

Geometrische Modelle haben den Vorteil, dass sie Räume und Objekte sehr genau modellieren können. Dadurch, dass exakte Koordinaten angegeben werden, können räumliche Informationen genau die Wirklichkeit wiedergeben und auch komplexere Anfragen, wie etwa die exakte Entfernung zwischen zwei Objekten oder der Grad der Überlappung zweier Gebiete, beantwortet werden. Außerdem sind diese Modelle recht einfach, da sie nur aus Koordinatenachsen und den Koordinaten der Punkte bestehen. Dadurch können leicht verschiedene Modelle ineinander übertragen und Sensordaten integriert werden. Andererseits haben sie den Nachteil, dass sie sehr schnell aus einer sehr großen Menge Daten bestehen und auch die mathematischen

Berechnungen sehr aufwändig werden können. In vielen Anwendungen ist die exakte Angabe von Koordinaten auch gar nicht nötig.

Die Nachteile von geometrischen Modellen sind gleichzeitig die Vorteile von hierarchischen Modellen. Hier müssen nur wenige Daten gespeichert und auch keine aufwändigen Berechnungen durchgeführt werden. Andererseits ist es hier schwieriger, verschiedene Modelle zu integrieren und genaue geometrische Aussagen können nicht gemacht werden.

Neben den „reinen“ geometrischen und hierarchischen Kontextmodellen gibt es auch noch hybride Modelle, die Elemente aus beiden Ansätzen übernehmen. Diese werden in Abschnitt 2.3 vorgestellt.

2.2 Semantic Spaces

In Beispiel für ein hierarchisches Kontextmodell sind die bei Microsoft entwickelten *Semantic Spaces* (siehe [2]). Die Semantic Spaces wurden als ein möglichst einfaches System zum Speichern und Verarbeiten von räumlichen Informationen entwickelt. Aus diesem Grunde wurde ein Modell entwickelt, welches nur einfache Enthaltungsbeziehungen von Räumen und Objekten in anderen Räumen abbildet und auf ein komplexes geometrisches Modell mit genauen Koordinaten aller Objekte verzichtet. Realisiert wurden die Semantic Spaces als relationales Datenbankschema, so dass das Modell in einer einfachen Datenbank gespeichert werden kann.

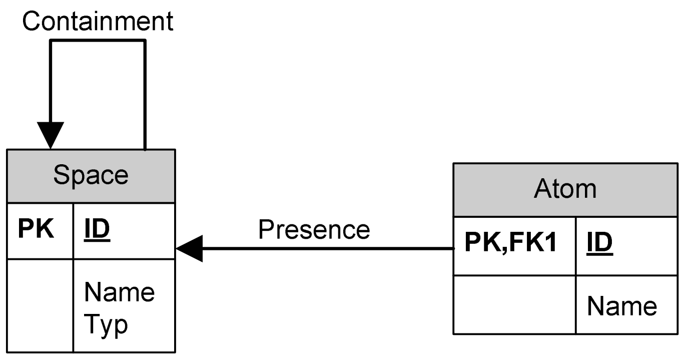

Fig. 2. Schema des Semantic Space Weltmodells

Das Weltmodell der Semantic Spaces besteht aus zwei Objektarten: *Räume* (*Spaces)* und *Atome*. Räume sind Abbildungen der realen Welt ab, wie etwa Gebäude, Stockwerke oder Zimmer (Der deutsche Begriff „Raum“ ist nicht sehr eindeutig, da er sowohl den englischen Begriff „space“ wie auch den Begriff „room“ bezeichnet. Im weiteren Verlauf der Arbeit wird zur Vereinfachung stets der Begriff „space“ mit „Raum“ und „room“ mit „Zimmer“ bezeichnet). Jeder Raum wird in dem Datenbankschema repräsentiert durch eine eindeutige ID, einen Namen und einen Typ, welcher angibt, um was für eine Art von Raum es sich handelt (etwa um ein Gebäude). Atome entsprechen beliebigen Objekten in der realen Welt (z.B. Personen, Sensoren, Computer).

Um aus diesen beiden Arten ein hierarchisches Modell aufzubauen, gibt es zwei verschiedene Beziehungstypen: Die *Enthaltungsbeziehung* zwischen zwei Räumen, gibt an, dass ein Raum komplett in einem anderen Raum enthalten ist (beispielsweise ein Stockwerk in einem Gebäude). Diese Beziehung entspricht der Teilmengenbeziehung in der Mengenlehre: Ist Raum A in Raum B enthalten, so ist jedes Element (Raum oder Atom), welches in A liegt, auch in B. Das Modell ist hier also sehr einfach gehalten, es erlaubt etwa nicht, dass ein Raum nur zum Teil in einem übergeordneten Raum liegt (z.B. ein Flur, der sich über mehrere Gebäude erstreckt). Allerdings ist es möglich zu modellieren, dass ein Raum in mehreren übergeordneten Räumen enthalten ist, diese sich also überschneiden. Die *Anwesenheitsbeziehung* zwischen einem Atom und einem Raum gibt an, dass das Atom sich in dem Raum befindet (z.B. ein PDA ist in einem Zimmer). Das bedeutet gleichzeitig, dass das Atom sich auch in allen Räumen befindet, in denen dieser Raum enthalten ist (im Beispiel befindet sich der PDA dann automatisch auch im Stockwerk und im Gebäude des Zimmers). Atome können selber keine weiteren Atome oder Räume enthalten. Das sich daraus ergebende Schema ist in Abbildung 2 dargestellt.

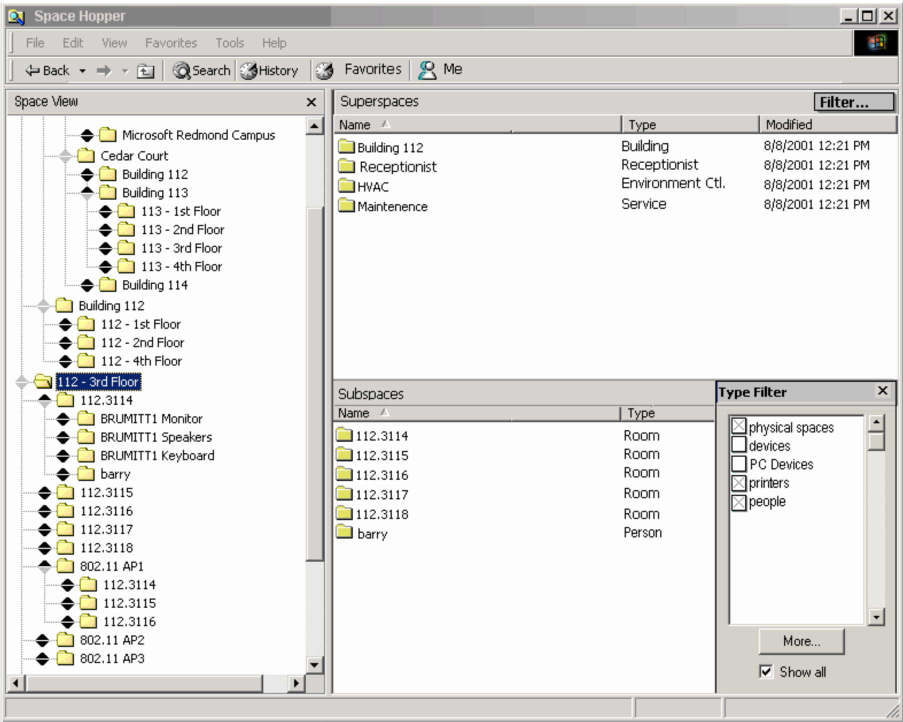

Fig. 3. Grafisches Nutzerinterface der Semantic Spaces (aus [2])

Da dieses hierarchische Weltmodell den Ort eines Objektes nur über das Enthaltensein in einem bestimmten Raum modelliert, ist es vergleichsweise einfach, den Ort beweglicher Objekte ohne eigene Sensorik (wie etwa GPS) festzustellen und zu aktualisieren. So kann der Aufenthalt einer Person etwa durch den Ort der Geräte, die sie benutzt, festgestellt werden: Loggt sie sich an einem bestimmten Computer ein, so befindet sie sich im selben Raum wie der Computer. Eine andere Möglichkeit wäre es, den Aufenthaltsort mitgeführter mobiler Geräte zu registrieren: Eine Person befindet sich immer in dem Raum, in dem der Funknetzknoten liegt, an dem ihr PDA angemeldet ist.

Die Entfernung zwischen zwei Objekten kann in einem solchen Modell nicht festgestellt werden, allerdings können einfache geometrische Abfragen, wie etwa die Suche nach dem nächstgelegenen Objekt einer bestimmten Sorte, trotzdem beantwortet werden: Soll beispielsweise der für eine bestimmte Person am schnellsten erreichbare Drucker ermittelt werden, so kann zunächst im direkt übergeordneten Raum (z.B. dem Zimmer) der Person gesucht werden und dann die Suche Schritt für Schritt auf übergeordnete Räume ausgedehnt werden (Stockwerk, Gebäude), bis ein Drucker gefunden wurde.

Brumitt und Shafer [2] stellen auch eine Möglichkeit vor, das Weltmodell der Semantic Spaces grafisch darzustellen, um die Administration des Modells zu erleichtern (siehe Abbildung 3). Die Darstellung repräsentiert das Modell als baumartige Struktur und ist an den Windows Explorer angelehnt. Dadurch kann der Nutzer sehr einfach durch die verschiedenen Räume navigieren. Der Nachteil hierbei ist allerdings, dass es in einer Baumstruktur nicht gut dargestellt werden kann, wenn sich zwei Räume überlappen. So können einzelne Räume oder Objekte in der Baumdarstellung mehrfach auftauchen, wenn sie in verschiedenen übergeordneten Räumen liegen.

2.3 Hybride Kontextmodelle

Wie in Abschnitt 2.1 vorgestellt, haben geometrische und hierarchische Kontextmodelle verschiedene Vor- und Nachteile. Um die Vorteile beider Modelle miteinander zu vereinen, wurden verschiedene Ansätze von Weltmodellen entwickelt, die sowohl über die einfachen Enthaltenseinsbeziehungen hierarchischer Modelle verfügen als auch geometrische Koordinaten oder Entfernungen speichern können, um genaue Aussagen über Position und Entfernung einzelner Objekte zueinander machen zu können. Im Folgenden sollen zwei unterschiedliche Ansätze näher vorgestellt werden

2.3.1 Anreicherung eines hierarchischen Modells

Hu und Lee [9] stellen ein Kontextmodell vor, welches im Wesentlichen hierarchisch strukturiert ist. Darüber hinaus ist es aber auch mit geometrischen Informationen über die Entfernungen zwischen verschiedenen Orten angereichert.

Die wichtigsten Elemente im Modell sind *Orte (Locations)* und *Ausgänge (Exits)*. Ein Ort ist ein abgeschlossener räumlicher Bereich, der durch verschiedene Ausgänge betreten und verlassen werden kann. Ein Ausgang ist also eine durchlässige Grenze zwischen verschiedenen Orten. Beispielsweise wäre eine Zimmertür ein Ausgang zwischen den beiden Orten „Zimmer“ und „Flur“. Neben Ausgängen, die zwei verschiedene Orte miteinander verbinden, gibt es auch Ausgänge, die aus dem gesamten Modell herausführen, also die Grenze zwischen modellierten und nicht mehr modellierten Bereich bilden. In dem Modell eines Gebäudes bilden etwa die Außentüren derartige Ausgänge.

Orte und Ausgänge bilden jeweils verschiedene Hierarchien, die aber ähnlich aufgebaut werden. Im Gegensatz zu den bisher vorgestellten hierarchischen Weltmodellen wird die Hierarchie zwischen verschiedenen Objekten hier nicht durch das Enthaltensein definiert. Orte und Ausgänge existieren nur nebeneinander, eine Überlappung oder das Enthaltensein zwischen verschiedenen Orten ist nicht vorgesehen. Die Hierarchie in diesem Modell entsteht durch die Erreichbarkeit von „außen“: Bei den Ausgängen bilden alle Ausgänge, die aus dem Modell herausführen, die oberste Hierarchieebene. Unter jedem Ausgang werden dann rekursiv alle Ausgänge eingeordnet, die noch nicht auf gleicher und höherer Ebene liegen und direkt (das heißt durch Durchquerung eines Ortes und ohne Passierung weiterer Ausgänge) von ihm erreichbar sind. Abbildung 4 zeigt ein Beispiel einer Ausgangs-Hierarchie im Modell eines Stockwerks. Hier führt der Lift aus dem Modell heraus, die Türen des Liftes bilden also einen Ausgang der obersten Ebene.

Bei der Orts-Hierarchie bilden analog die Orte, die direkt an die Ausgänge des Modells grenzen, die oberste Ebene. Ebenso werden rekursiv alle Orte die von Orten in der Hierarchie direkt (durch nur einen Ausgang) erreichbar sind, unter diesen eingehängt.

Zusätzlich zu dem hierarchischen Modell wird jeweils zwischen zwei direkt nebeneinander liegenden Ausgängen die Entfernung gespeichert. Für die Entfernung sind verschiedene Skalen denkbar, wie etwa die Distanz in Metern oder der Energieverbrauch auf der Strecke. Durch diese geometrische Information kann auch die Entfernung oder der kürzeste Weg zwischen beliebigen Orten berechnet werden, wozu geeignete Algorithmen von Hu und Lee vorgestellt werden.

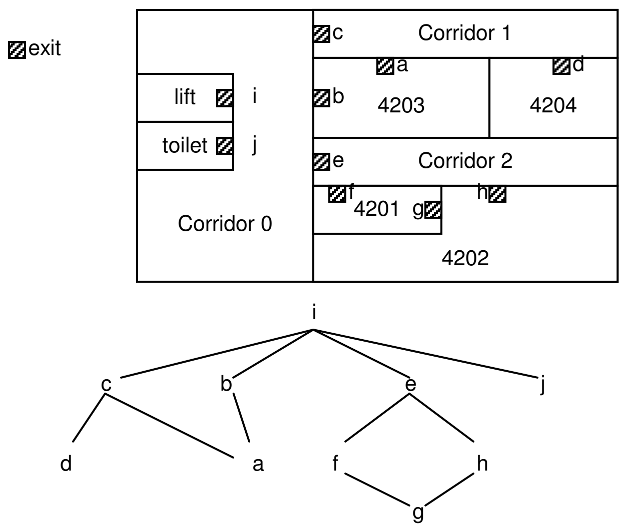

Fig. 4. Abbildung eines Gebäudestockwerks und daraus erstellte Exit-Hierarchie (nach [9])

2.3.2 Ein Modell mit hierarchischen Koordinatensystemen

Im Rahmen eines Projektes zur Entwicklung kontextsensitiver Anwendungen wurde an der Carnegie Mellon University ein weiteres Kontextmodell entwickelt, welches sowohl hierarchisch aufgebaut ist als auch über Koordinatensysteme verfügt (siehe [10]).

Grundtypen dieses Modells sind drei verschiedene Arten von Orten oder Räumlichkeiten: *Raum (Space)*, *Gebiet (Area)* und *Punkt (Point)*. Ein Raum ist die Repräsentation eines physikalischen Raumes, der auch in der realen Welt von anderen Räumen eindeutig abgegrenzt ist (z.B. ein Zimmer oder ein Gebäude) und weiter unterteilt werden kann. Ein Gebiet ist zwar auch ein dreidimensionaler Raum in der realen Welt, im Gegensatz zu einem Raum ist es aber nicht physisch abgegrenzt, sondern wird nur durch virtuelle Eckpunkte oder geometrische Formen eingegrenzt (beispielsweise der Empfangsbereich eines W-LAN-Routers). Mit einem Punkt kann schließlich der Ort von einzelnen Objekten modelliert werden, bei denen nur die Position und nicht die räumliche Ausdehnung relevant ist (z.B. Drucker, Personen).

Räume können selber weitere Räume, Gebiete und Punkte enthalten, aus ihnen kann also ein einfaches hierarchisches Weltmodell zusammengesetzt werden, wie es in Abschnitt 2.1 beschrieben wurde. Gebiete sind dagegen nicht weiter hierarchisch unterteilbar.

Um nun auch geometrische Informationen verarbeiten zu können, verfügt jeder Raum über ein eigenes Koordinatensystem. In diesem Koordinatensystem werden für die enthaltenen Punkte die Position, für die enthaltenen Gebiete die Eckpunkte und für die enthaltenen Räume die geometrische Form und Ausdehnung angegeben. Damit können Entfernungsangaben und weitere geometrische Berechnungen innerhalb eines einzelnen Raumes durchgeführt werden.

Darüber hinaus ist zu jedem Raum angegeben, welche Position sein Koordinatenursprung im Koordinatensystem des übergeordneten Raumes (bzw. im globalen Koordinatensystem, falls der Raum auf der obersten Hierarchieebene steht) hat und wie sein Koordinatensystem im Verhältnis zum übergeordneten gedreht ist. Dadurch ist es möglich, alle Koordinaten innerhalb eines Raumes in Koordinaten übergeordneter Räume umzurechnen. Somit ist es auch möglich, Berechnungen zu beliebigen Objekten in verschiedenen Räumen durchzuführen, indem die jeweiligen Koordinaten Schritt für Schritt in Koordinaten der übergeordneten Räume transformiert werden, bis beide Koordinaten in einem Koordinatensystem vorliegen, das beiden Objekten übergeordnet ist.

Der hierarchische Baum des Weltmodells wird also mit weiteren geometrischen Informationen zu jedem Raum (geometrische Form, Ausdehnung, Ursprung und Drehung des Koordinatensystems) angereichert, wie in Abbildung 5 gezeigt.

Zur Adressierung von Objekten im Raum wurde ein so genannter *Aura Location Identifier* (ALI) entwickelt, womit sowohl Räume als auch Gebiete und Punkte identifiziert werden können. Ein Raum wird hierbei durch den Pfad aller über im liegenden Spaces angegeben. Bei Gebieten und Punkten werden der jeweils übergeordnete Raum und die Koordinaten innerhalb dieses Raumes genannt. Beispiele sind:

- Für einen Raum: „*ali://TU-KL/36/3/330*“ (Zimmer 330 im 3. Stock in Gebäude 36 der Uni Kaiserslautern)

- Für einen Punkt: „*ali://TUi-KL/36/3/330#(1,4,5)*" (Ein Punkt in Zimmer 330 mit den Koordinaten (1,4,5) im Koordinatensystem des Zimmers)
- Für ein Gebiet: „*ali://TU-KL/36/3/330#{(0,0),(1,0),(2,3)-(2,3)}*" (Ein Raum in Zimmer 330, der die Grundfläche eines Dreiecks hat und sich von der Höhe 2 bis zur Höhe 3 erstreckt)

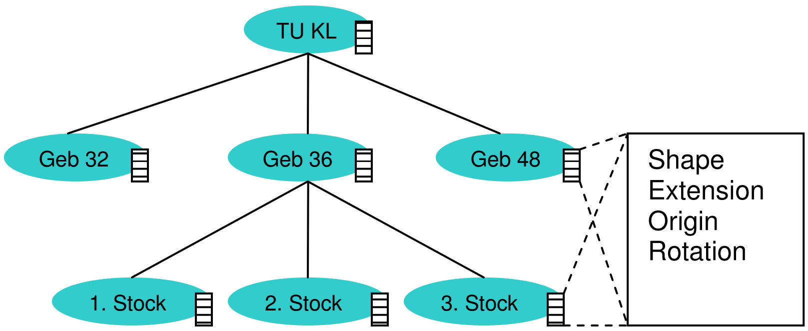

Fig. 5. Hierarchisches Weltmodell, angereichert mit Informationen zu den jeweiligen Koordinatensystemen (nach [10])

Auf solchen ALIs können nun verschiedene Operatoren angewendet werden, die sich sowohl auf die hierarchischen als auch auf die geometrischen Eigenschaften beziehen. Als mögliche Operatoren nennen Jiang und Steenkiste [10] unter anderem die Distanz zwischen zwei ALIs, die Angabe der über- oder untergeordneten Räume eines ALIs oder die Frage, ob sich zwei ALIs überschneiden.

2.4 Ein Indoor-Kontextmodell

Neben den hierarchischen und geometrischen Weltmodellen gibt es aber auch Modelle, die auf spezielle Anforderungen zugeschnitten sind und in keines der beiden Muster passen. Ein solches Beispiel soll im Folgenden vorgestellt werden: Schindler et al. [17] stellen ein Modell zur Speicherung des räumlichen Kontextes vor, welches sich vor allem zur Modellierung von Umgebungen innerhalb eines Hauses, das heißt Räume und Durchgänge, eignet.
Aus der zu modellierenden Umgebung wird ein Graph erstellt, der aus Knoten und (gerichteten und ungerichteten) Kanten besteht:

- Jeder Knoten repräsentiert eine Seite eines Durchganges.
- Ungerichtete Kanten repräsentieren den Raum zwischen zwei verschiedenen Durchgängen, also Räume oder Flure. Für das Modell bedeutet eine ungerichtete Kante einen Zustand, in dem sich die Umgebung nicht ändert. Jede Kante hat als Attribut ihre Länge in Schritten zwischen den beiden Durchgängen.

- Gerichtete Kanten repräsentieren eine Richtung eines Durchganges. Ein Durchgang besteht also immer aus zwei entgegen gesetzten Kanten. Sie haben standardmäßig die Länge 1 (Schritt).

In diesem Modell werden also Durchgänge als Paar von Knoten, die durch gerichtete Kanten verbunden sind, und Räume durch mit ungerichteten Kanten verbundene Knoten dargestellt. Räume mit nur einem Ausgang bestehen aus einem durch eine ungerichtete Kante mit sich selber verbundenen Knoten. In Abbildung 6 wird der schon in Abbildung 4 verwendete Grundriss im Modell dargestellt.

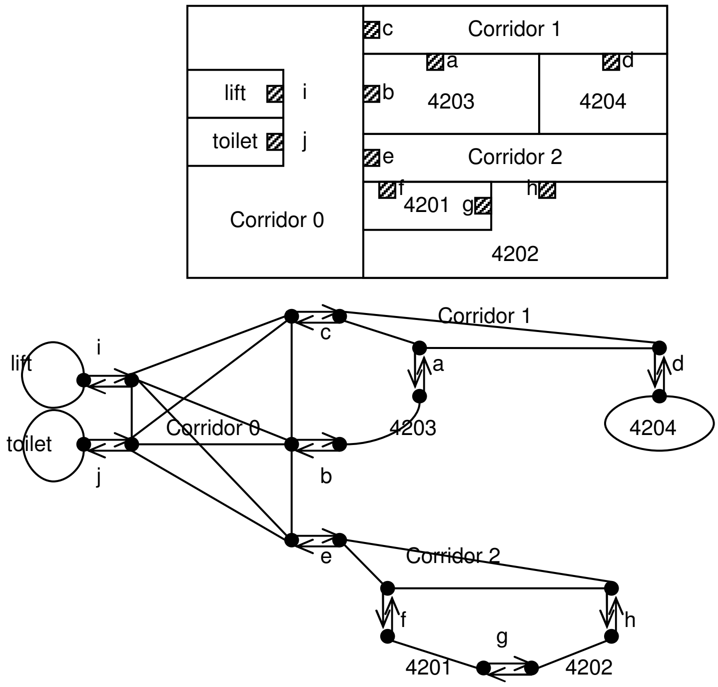

Fig. 6. Modellierung eines Grundrisses mit 5 Räumen A-E

Im Vergleich zu den vorher vorgestellten Kontextmodellen ist dieses Modell einfacher und daher auch weniger mächtig. So kann, im Gegensatz zu den geometrischen Modellen, bis auf einfache Entfernungsangaben kaum eine Aussage zu Größe, Lage oder Entfernung von Objekten gemacht werden. Im Vergleich zu hierarchischen Modellen ist hier die Zerlegung von Räumen in kleinere Räume nicht möglich.

Andererseits ermöglicht diese Einfachheit aber eine sehr einfache automatische Modellierung von Räumlichkeiten. In [17] ist beschrieben, wie ein Modell einer Umgebung durch ein tragbares System erstellt werden kann. Hierzu trägt der Anwender einen Entfernungssensor am Kopf, der durch Abstandsmessung zur Wand den Unterschied zwischen Räumen und Durchgängen feststellen kann. Verschiedene Durchgänge können durch ihr Profil für den Sensor und durch Handgesten des Anwenders identifiziert werden. Aus diesem Grund besteht auch ein Durchgang aus zwei entgegengesetzt gerichteten Kanten, da der Sensor auf einer Seite des Körpers getragen je nach Bewegungsrichtung unterschiedliche Seiten des Durchgangs aufnimmt. Auch die Entfernung zwischen Durchgängen kann automatisch gemessen werden, indem das System einfach die Schritte des Anwenders zählt.

2.5 Anreicherungen von Kontextmodellen

Damit eine kontextbewusste Anwendung flexibel auf sowohl die Umgebung als auch auf die jeweiligen Anforderungen reagieren kann, genügt es meist nicht, einfach nur ein Modell der realen Welt und ihrer Räumlichkeiten zu speichern und zu nutzen. Darüber hinaus muss es auch die Möglichkeit geben, in dem Modell der Umgebung nicht nur reale, sondern auch virtuelle Objekte abzulegen, die in der realen Umgebung keine Entsprechung haben. So kann es etwa nötig sein, an bestimmten Punkten Informationen bereitzustellen, die für den Anwender an genau diesem Ort von Bedeutung sind (Beispielsweise Erläuterungen zu Sehenswürdigkeiten). Auf der anderen Seite ist es auch denkbar, dass der Nutzer an bestimmten Punkten Eingaben in das System mit seiner aktuellen Position verknüpfen möchte, so dass diese Einträge als virtuelle Objekte im Raummodell vorhanden sind. Ein Beispiel hierfür liefert Pascoe [15], wo Beobachtungen von Tieren an den jeweiligen Punkten im geographischen Modell eingetragen werden.

Die Erweiterung von räumlichen Kontextmodellen um virtuelle Objekte führt zu der Entwicklung von solchen angereicherten Modellen, für die unterschiedliche Ansätze existieren.

Im Nexus-Projekt (siehe Abschnitt 4.2) wurde das so genannte *Augmented World Model*, ein objektorientiertes Kontextmodell entwickelt. Nach Grossmann et al. [7] besteht es im Wesentlichen aus geographischen und mobilen Objekten, welche reale Objekte widerspiegeln, und aus virtuellen Objekten. Diese haben eine räumliche Position und können von Anwendungen zusammen mit den realen Objekten dargestellt werden (beispielsweise als virtuelle Litfasssäule, die vor einem Gebäude steht). Sie können auf externe Informationen wie etwa Internetseiten oder Multimediainhalte verweisen, die dann an dieser Stelle angezeigt werden.

Ein ähnlicher Ansatz wird bei den *stick-e notes* [1] verfolgt. Diese sind Objekte, die, analog zu Klebezetteln, an einen bestimmten Kontext angeheftet werden können. Eine stick-e note besteht dazu zum einen aus Kontextinformationen und zum anderen aus ihrem Inhalt (etwa einem Text). Beide Teile werden in einem standardisiertem Format (SGML) gespeichert. Als Kontext können unter anderem der Ort, die Zeit und die Bewegungsrichtung angegeben werden. Gibt der Nutzer eine stick-e note ein, so wird sie mit den relevanten aktuellen Kontextdaten versehen und abgespeichert. Wenn er nun wieder in einen Zustand kommt, der mit dem Kontext der gespeicherten Note übereinstimmt, wird sie automatisch aktiviert und dem Nutzer angezeigt. Mehrere stick-e notes können zu einem *stick-e document* zusammengefasst und abgespeichert werden. Beispielsweise kann also ein Rundgang durch ein Gelände dadurch abgespeichert werden, dass an bestimmten Wegpunkten eine stick-e note erstellt wird, die den weiteren Weg beschreibt (z.B. „Von hier an rechts den Hügel hinauf"). Diese notes können dann zu einem document zusammengefasst werden, durch welches der Nutzer Punkt für Punkt den Weg geleitet wird.

Weitergehend ist der Ansatz von Tarumi et al. [19]. Hier wird nicht ein Modell der realen Umgebung durch einzelne virtuelle Objekte angereichert, sondern es wird eine komplette virtuelle Welt neben der realen modelliert. Diese Welt besteht (ähnlich einer virtuellen Welt in Computerspielen) aus virtuellen architektonischen Objekten und virtuellen Agenten, welche sich bewegen und Nachrichten austauschen können. Die Schnittstelle zwischen der realen und der virtuellen Welt besteht darin, dass beide Welten dasselbe Koordinatensystem verwenden. Die virtuelle Welt wird also in einem so genannten *Overlaid Virtual Model* über die reale Welt gelegt. Bewegt sich der Nutzer in der realen Welt, so verändert er dabei auch seine Position in der virtuellen Welt, die er auf seinem Gerät sehen kann. Dadurch kann er beispielsweise bei einem Rundgang durch eine historische Stätte ständig auf seinem Bildschirm sehen, wie sein aktueller Ort in der Vergangenheit ausgesehen hat.

3. Indizes zur räumlichen Speicherung

Neben einem Kontextmodell benötigt eine kontextbewusste Anwendung auf tieferer Ebene auch Speicherungsstrukturen, mit denen auf Objekte möglichst effizient anhand von (räumlichen) Kontextinformationen zugegriffen werden kann. Analog zu herkömmlichen Datenbanken geschieht dies über Indexstrukturen. Mit ihrer Hilfe ist es möglich, schnell (das heißt mit logarithmischem Aufwand) Anfragen zu beantworten, die sich auf räumliche Daten beziehen. Beispiele solcher Anfragen sind:

- Welche Objekte befinden sich am Ort X?
- Welche Objekte sind im Umkreis von 100 Metern von Objekt Y vorhanden?

Gerade bei geometrischen Kontextmodellen ist ein effizienter Zugriff auf Objekte anhand ihrer Koordinaten besonders wichtig, da sie oft einen sehr großen mehrdimensionalen Wertebereich annehmen können (z.B. bei GPS), der in annehmbarer Zeit nicht sequenziell durchsucht werden kann. Herkömmliche Indexstrukturen wie etwa B-Bäume eignen sich hier nicht, da sie nur auf eindimensionalen Attributen arbeiten. Hierarchische Kontextmodelle dagegen benötigen oft keine zusätzlichen Indexstrukturen, da hier effiziente Zugriffspfade bereits durch die Hierarchie vorgegeben werden.

In diesem Abschnitt sollen einige Indexstrukturen vorgestellt werden, mit denen effizient auf räumliche Koordinaten zugegriffen werden kann.

3.1. Quad-Tree

Quad-Trees wurden schon 1974 von Finkel und Bentley [5] als Erweiterung der B-Bäume auf zweidimensionale zusammengesetzte Indizes vorgestellt. Da ein in zweidimensionalen Koordinaten angegebener Ort auch als zusammengesetzter Index (mit den einzelnen Koordinaten als Attribute) gesehen werden kann, eignen sie sich auch zum Zugriff auf räumliche Daten.

Die Funktionsweise eines Quad-Trees ist sehr ähnlich zu der eines binären Suchbaumes. Dort hat jeder Knoten zwei Bäume als Nachfolger, in denen alle

Indexwerte kleiner (links) bzw. größer (rechts) sind, als der eigene Indexwert. Beim Suchen oder Einfügen eines Knotens wird der zu suchende (oder einzufügende Index) mit dem Wert der Wurzel verglichen und diese Suche daraufhin rekursiv rechts (falls der gesuchte Wert größer ist als der Wert der Wurzel) oder links (falls der gesuchte Wert kleiner ist) fortgesetzt, bis die gewünschte Position im Baum gefunden ist.

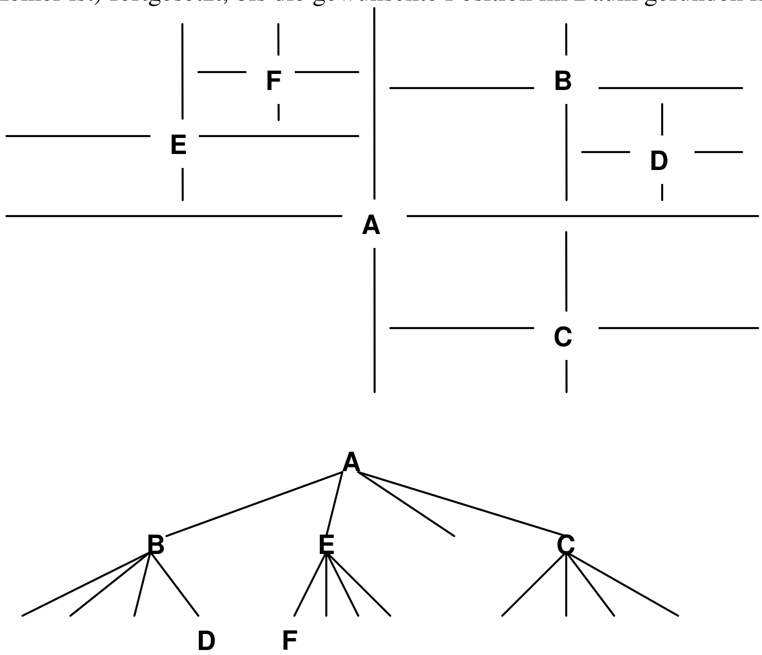

Fig. 7. Quad-Tree und von ihm repräsentierte Struktur (nach [5])

Ein Quad-Tree ist ebenfalls als Baum aufgebaut, bei dem jeder Knoten aber vier Nachfolger hat: Jeweils einen nordöstlichen, nordwestlichen, südwestlichen und südöstlichen. Analog zum binären Suchbaum liegen im nordöstlichen Nachfolger alle Indexwerte mit einem größeren 1. und 2. Indexattribut, im Nordwesten alle Werte mit einem kleineren 1. und größeren 2. Attribut, im Südwesten alle mit kleinerem 1. und 2. Attribut und im Südosten alle mit größerem 1. und kleinerem 2. Attribut. Sollen als Index nun zweidimensionale Koordinaten gespeichert werden, liegen beispielsweise im nordöstlichen Nachfolger eines Knotens alle Knoten mit größeren x- und y-Koordinaten und im südwestlichem Nachfolger alle Knoten mit kleineren x- und y-Koordinaten.

Wird nun ein Objekt mit den Koordinaten X und Y abgespeichert, so wird eine Referenz auf das Objekt mit diesen Koordinaten in den Quad-Tree eingefügt. Dazu werden die Koordinaten des Objektes mit der Wurzel verglichen um danach, je nach Ergebnis des Vergleiches, in einen der vier Nachfolger abzusteigen. Ist dieser Nachfolger leer, so wird der neue Knoten dort eingefügt, andernfalls wird das Einfügen rekursiv mit dem Nachfolger fortgesetzt.

Werden beispielsweise nacheinander Objekte mit den Koordinaten A(100, 100), B(200, 200), C(200, 50), D(300, 150), E(50, 150) und F(75, 200) eingefügt, ergibt sich daraus die in Abbildung 7 gezeigte Struktur.

Sollen nun die Objekte an einem bestimmten Ort gesucht werden, so wird analog zum Einfügen solange rekursiv im Baum abgestiegen, bis entweder das gewünschte Objekt gefunden wird oder ein leerer Knoten erreicht wird (was bedeutet, dass kein Objekt an diesem Ort vorhanden ist).

Finkel und Bentley zeigen in [5] auch, wie mit geeigneten Balancierungsalgorithmen der Quad-tree so optimiert werden kann, dass die Höhe des Baumes und damit der Aufwand zum Suchen stets logarithmisch ist.

Der vorgestellte Quad-Tree eignet sich in dieser Form nur zum Zugriff auf zweidimensionale Ortsangaben. Jedoch ist es sehr leicht, das Prinzip auf dreidimensionale Koordinaten auszuweiten, indem in einem so genannten *Oct-Tree* jeder Knoten acht Nachfolger hat, die die jeweiligen Richtungen im dreidimensionalen Raum repräsentieren.

3.2. R-Tree

Eine weitere Möglichkeit zur räumlichen Indexierung von Objekten stellt der von Guttman [8] vorgestellte *R-Tree* dar. Mit ihm können im Gegensatz zum Quad-Tree nicht nur der Ort, sondern auch die Ausdehnung von Objekten adressiert werden und somit auch Anfragen wie etwa die Überlappung von Objekten beantwortet werden. Dies wird dadurch erreicht, dass jeder Knoten des Baumes nicht nur als Punkt, sondern als Rechteck (bzw. Quader im dreidimensionalen Fall) gespeichert wird.

Der Aufbau eines R-Trees ähnelt dem eines B*-Baumes: Referenzen auf gespeicherte Objekte sind in den Blättern des Baumes enthalten. Dabei sind für jedes Objekt die Eckpunkte eines Rechteckes angegeben, welches das Objekt gerade noch vollständig umschließt. Zu jedem Knoten des Baumes ist ebenfalls ein Rechteck angegeben, welches alle Rechtecke der untergeordneten Knoten (bzw. referenzierten Objekte bei Blattknoten) minimal umschließt. Hierbei können sich verschiedene untergeordnete Rechtecke auch überlappen, so dass es (im Gegensatz zu B*-Bäumen) nicht immer eindeutig ist, in welchem Unterbaum eines Knotens ein bestimmter Ort gespeichert ist.

In Abbildung 8 ist ein Beispiel eines R-Trees gegeben. Hier sind 8 Gebäude der TU Kaiserslautern mit den zwischen ihnen verlaufenden Gängen G1-G4 modelliert. In diesem Beispiel liegt Gebäude 13 in beiden übergeordneten Rechtecken R8 und R9, könnten also in beiden Unterbäumen gespeichert sein. Hier liegt es unter R8.

Wird nun ein Objekt an einem bestimmten Ort gesucht, so wird die Suche (analog zur Suche in B*-Bäumen) an der Wurzel des Baumes begonnen, indem die gesuchten Koordinaten mit allem dort gespeicherten Rechtecken verglichen werden. Diese Suche wird nun rekursiv mit allen Unterbäumen fortgesetzt, deren Rechtecke den gesuchten Ort umschließen. Wird ein Blattknoten erreicht, so werden alle Objekte zurückgegeben, deren Rechtecke den gewünschten Orte enthalten.

Da sich die Rechtecke überlappen können, muss die Suche gegebenenfalls in mehreren Unterbäumen fortgesetzt werden, ist also aufwändiger als die Suche in einem Quad-Tree. Dafür kann hier auch sehr einfach eine Suche nach allen Objekten in einem bestimmten Gebiet durchgeführt werden: Anstatt jeden Knoten auf Rechtecke zu überprüfen, die einen bestimmte Punkt enthalten, wird nun nach Rechtecken gesucht, die sich mit dem bewünschten Bereich überschneiden.

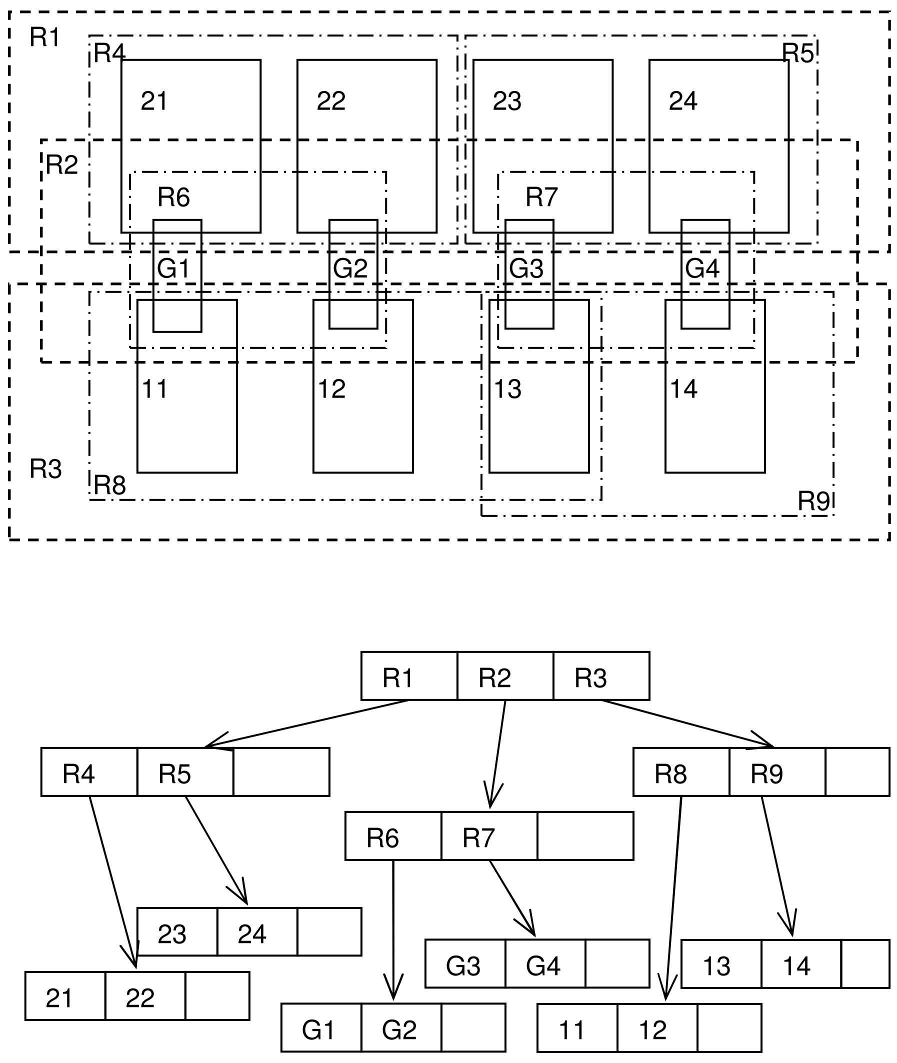

Fig. 8. Beispiel einer Gruppe von Gebäuden der TU Kaiserslautern und sich daraus ergebender R-Tree

Einfügen und Löschen sind in einem R-Tree recht komplexe Operationen, da hier unter Umständen Knoten geteilt oder zusammengeführt (wie bei einem B*-Baum) und auch die Größe von übergeordneten Rechtecken angepasst werden müssen: Wird ein neues Objekt in den Baum eingefügt, so wird zuerst in einer Suchoperation der

Knoten identifiziert, in dem die Objektreferenz gespeichert werden soll. Ist der Knoten bereits vollständig gefüllt, so muss er aufgeteilt und die Teilung eventuell rekursiv nach oben fortgesetzt werden. Wenn das umschließende Rechteck des neu eingefügten Objektes über die Grenzen des übergeordneten Rechtecks hinausgeht, so muss dieses vergrößert werden und auch diese Anpassung rekursiv fortgesetzt werden.

Beim Löschen eines Objektes muss zuerst der zugehörige Knoten identifiziert und die Referenz gelöscht werden. Unterschreitet nun der Füllgrad des Knotens den Mindestgrad, so müssen Knoten analog zum Vorgang in einem B*-Baum zusammengeführt werden. Anschließend muss die Größe der umschließenden Rechtecke angepasst werden, so dass sie wieder minimal werden.

Genauso wie der Quad-Tree kann auch der R-Tree sehr leicht auf mehrere Dimensionen ausgeweitet werden.

3.3 UB-Tree

Die bisher vorgestellten Indexstrukturen sind die am weitesten verbreiteten und bekanntesten für räumliche und mehrdimensionale Schlüssel. Darüber hinaus gibt es aber noch viele weitere Forschungsansätze für mehrdimensionale Indexstrukturen, von denen ein Beispiel im Folgenden dargestellt werden soll.

Der *UB-Tree* (Universal B-Tree) wurde von Bayer und Markl (siehe [12]) als Erweiterung des B*-Baumes vorgeschlagen, die auch mehrdimensionale Schlüssel unterstützt. Das Prinzip hierbei ist recht einfach: Der zu indizierende mehrdimensionale Schlüssel wird einfach durch eine bijektive Abbildung in eine eindimensionale Zahl umgerechnet, welche als Schlüssel eines B*-Baumes verwendet wird. Dadurch kann die übliche Implementierung von B*-Bäumen weitestgehend unverändert übernommen werden.

Die Abbildung der einzelnen Dimensionen des mehrdimensionalen Schlüssels in einen einzelnen Wert erfolgt nach dem Z-Verfahren. Hierbei werden die Schlüsselattribute bitweise verschränkt, das heißt die resultierende Zahl hat in Binärdarstellung als niedrigstes Bit das niedrigste Bit des ersten Schlüsselattributes, als zweitniedrigstes Bit das niedrigste des zweiten Attributes und so weiter, formal (mit Anzahl der Dimensionen d, Bitlänge der einzelnen Attribute s, Bit des j-ten Attributs an der i-ten Stelle x_{ij}):

$$Z(x) = \sum_{i=0}^{s-1} \sum_{j=1}^{d} x_{i,j} * 2^{i*d+i-1}$$

Bei zweidimensionalen Koordinaten als Schlüssel ergibt sich damit Tabelle 1.

Diese Abbildung macht auch den Namen des Z-Verfahrens ersichtlich, da die sich ergebenden Zahlen in der Tabelle hierarchische z-förmige Verläufe annehmen.

Durch die bidirektionale Abbildung des Z-Verfahrens kann das Einfügen und Löschen von Objekten genauso erfolgen wie bei B*-Bäumen. Auch die Suche nach Objekten an einem bestimmten Punkt erfordert keine zusätzlichen Algorithmen. Schwieriger ist jedoch die Suche aller Objekte in einem bestimmten Bereich. Hier müssen von den Eckpunkten des Bereiches ausgehend Schritt für Schritt verschiedene Koordinaten des Bereiches umgerechnet werden und dann nach den zugehörigen

Blattknoten im B*-Baum gesucht werden. In diesen Knoten muss schließlich für alle referenzierten Objekte überprüft werden, ob sie im gesuchten Bereich liegen.

Der UB-Tree ist durch das Z-Verfahren für beliebig viele Dimensionen anwendbar, jedoch steigt der Aufwand exponentiell mit der Anzahl der Dimensionen, weswegen sein Einsatz ab einer bestimmten Anzahl von Dimensionen nicht mehr praktikabel ist.

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
0	0	1	4	5	16	17	20	21
1	2	3	6	7	18	19	22	23
2	8	9	12	13	24	25	28	29
3	10	11	14	15	26	27	30	31
4	32	33	36	37	48	49	52	53
5	34	35	38	39	50	51	54	55
6	40	41	44	45	56	57	60	61
7	42	43	46	47	58	59	62	63

Tabelle 1. Umrechnungstabelle zweidimensionaler Koordinaten in eindimensionale nach dem Z-Verfahren

3.4 Vergleich der verschiedenen Verfahren

Stellt man die drei vorgestellten Indexstrukturen einander gegenüber, so unterscheiden sie sich vor allem in ihrer Mächtigkeit und den Kosten der Einfüge- Lösch- und Suchoperationen.

Die geringsten Ausführungskosten liegen bei einem UB-Tree, da hier nach einer einfachen Umrechnung der Koordinaten dieselben schnellen Operationen ausgeführt werden wie bei B*-Bäumen. Allerdings können hier auch nur gezielte Suchen nach bestimmten Koordinaten durchgeführt werden.

In einem Quad-Tree können zusätzlich auch recht schnell Objekte in der Umgebung oder einem bestimmten Bereich identifiziert werden, allerdings sind hier die Kosten für die Operationen schon höher, weil mehrere Dimensionen gleichzeitig betrachtet werden.

Der R-Tree stellt eine noch mächtigere Variante zur Indexierung von räumlichen Objekten dar, da er auch ausgedehnte Objekte adressieren kann. Diese größere Mächtigkeit wird aber durch die höchsten Kosten in den Operationen Suchen, Einfügen und Löschen bezahlt.

Insgesamt hängt die Wahl einer geeigneten Indexstruktur also von den Anforderungen an die Mächtigkeit und den Ressourcenverbrauch einer bestimmten Anwendung ab. Je nach Priorität ist eine andere Indexstruktur optimal.

4. Systeme zur Speicherung räumlicher Informationen

Die bisher gezeigten Beispiele und Anwendungen von räumlichen Informationen und Kontextmodellen beschränkten sich nur auf recht einfache Systeme. In diesem Abschnitt sollen nun komplexe Systeme vorgestellt werden, die Kontextdaten speichernd und verarbeitend können. Diese Systeme enthalten neben Kontextmodellen auch Möglichkeiten, Kontextdaten effizient zu speichern und sie Anwendungen geeignet zur Verfügung zu stellen. Im Folgenden sollen zwei unterschiedliche Beispiele vorgestellt werden. Das eine Beispiel ist eine Spezifikation von Systemen, die geographische Daten speichern, das zweite Beispiel beschreibt ein konkretes System zur Speicherung von Kontextdaten.

4.1 Open GIS

Das *Open GeoInformation System* (OpenGIS [14]) ist eine Menge von Spezifikationen zur Nutzung geographischer Daten. Es wird verwaltet vom Open Geospatial Consortium (OGC), einer Gruppe von über 300 Firmen, Regierungsbehörden und Universitäten. Ziel von OpenGIS ist die Vereinheitlichung der räumlichen Daten und ihrer Nutzung, so dass verschiedenste Systeme, die Geodaten nutzen, miteinander kommunizieren und die Daten untereinander austauschen können. Die entwickelten Standards sind frei verfügbar.

OpenGIS enthält zwei verschiedene Arten von Spezifikationen: abstrakte und Implementierungsspezifikationen. Abstrakte Spezifikationen beschreiben die Zerlegung eines Systems in einzelne Komponenten, deren Funktionalität und die Schnittstellen und Kommunikation zwischen einzelnen Systemen. Von diesen Spezifikationen ausgehend beschreiben Implementierungsspezifikationen detaillierter den inneren Aufbau der Komponenten und Schnittstellen. Die gesamte abstrakte Spezifikation des Systems ist aufgeteilt in 16 Themen, die in Abbildung 9 mit ihren gegenseitigen Abhängigkeiten dargestellt sind.

Das erste Thema *Feature Geometry* legt die verwendeten geometrischen Strukturen fest. Es entspricht der internationalen Norm ISO19107 und definiert Formen wie „Punkt“, „Linie“, „Fläche“ und „Körper“.

Das zweite und dritte Thema spezifizieren die Nutzung von Koordinatensystemen (das zugrunde liegende Kontextmodell ist also ein geometrisches nach der Klassifikation von Abschnitt 3). *Spatial Referencing by Coordinates* definiert Koordinatenreferenzsysteme. Diese sind Koordinatensysteme, die zusätzliche Informationen enthalten, in welchem Verhältnis die einzelnen Koordinaten mit Punkten auf der Erdoberfläche stehen. Darüber hinaus wird die Umrechnung zwischen verschiedenen Koordinatenreferenzsystemen spezifiziert. *Location Geometry Structures* ergänzt dazu weitere spezielle Koordinatensysteme, wie etwa digitale Bilder, die Verzerrungen durch die Projektion aufweisen.

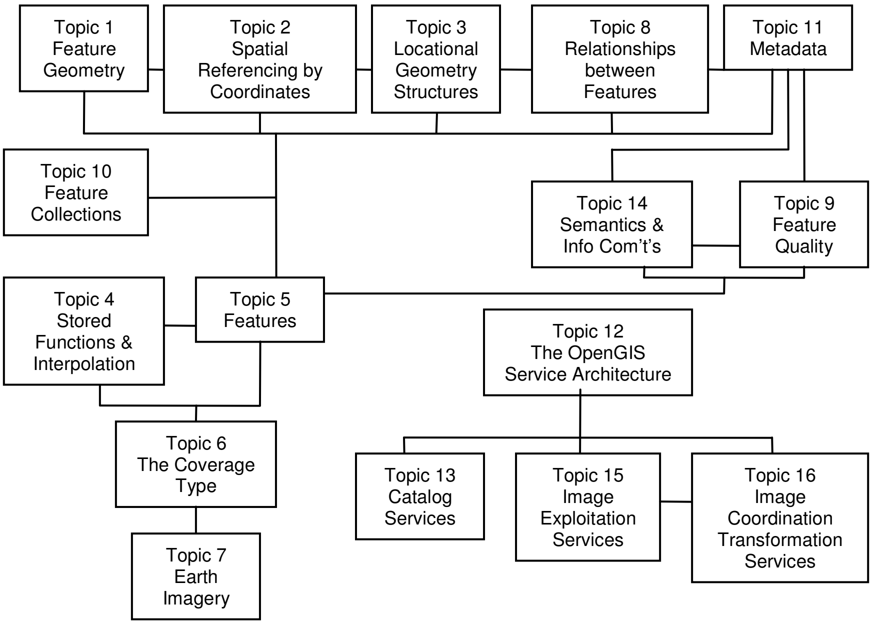

Fig. 9. Abhängigkeiten zwischen den abstrakten Spezifikationen (aus[14])

Die Themen 5 bis 7 beschreiben die Typen von geographischen Informationen, die verarbeitet und ausgetauscht werden. In Thema 5 wird der Grundtyp *Feature* definiert. Ein Feature ist die Repräsentation eines realen oder abstrakten Objektes im Weltmodell des Systems. Grundsätzlich gibt es zwei Arten von Features: Geometrische Features (*Features with Geometry*) entsprechen räumlichen Objekten wie etwa Gebäude, Straßen oder Flüsse. Dieser Datentyp ist mit seinen Attributen (z.B. den Koordinaten) und Operationen in Thema 5 spezifiziert. Eine weitere Art von Feature ist die Abdeckung (*Coverage*), die in *The Coverage Type* definiert wird. Eine Abdeckung beschreibt einen Ausschnitt des Weltmodells, indem sie einem bestimmten Bereich von Koordinaten Werte zuordnet. Ein Beispiel dafür ist eine Landkarte. Ein spezieller Typ von Abdeckung ist ein Bild der Erdoberfläche, der in *Earth Imagery* genauer definiert wird. Thema 4 (*Stored Functions and Interpolation*) spezifiziert Hilfsfunktionen zur Umrechnung zwischen Punkten in einer Abdeckung und einem Referenzkoordinatensystem.

Die Themen 8 und 10 definieren weitere Unterstützungen für die Features. In *Relationships between Features* werden die verschiedenen Typen von Verbindungen zwischen verschiedenen Features definiert. Beispiel einer solchen Verbindung ist etwa ein Feature „Straße" welches ein anderes Feature „Fluss" an einer Brücke kreuzt. *Feature Collection* definiert die Möglichkeit, Klassen von Features mit einem eigenen Schema zu erstellen.

In den Themen 12, 13, 15 und 16 werden die angebotenen Dienste spezifiziert. *The OpenGIS Service Architecture* beschreibt den Grundaufbau der Services. In *Catalog*

Services werden die Dienste zum Zugriff auf die Daten spezifiziert, wie etwa das Einfügen eines neuen Features oder die Suche nach Objekten. *Image Explotation Services* beschreibt die Nutzung von geographischen Bildern, wie etwa die Anzeige, das Anreichern von Bildern mit zusätzlichen Informationen oder das Extrahieren von Features aus Bildern. In *Image Coordination Transformation Services* werden die Dienste zur Umrechnung von Koordinaten zwischen Bildern beschrieben. Dabei muss unter anderem auch die Verzerrung durch unterschiedliche Perspektiven berücksichtigt werden.

Neben diesen abstrakten Spezifikationen existieren noch viele weitere Spezifikationen, die zur Ergänzung der vorhandenen oder als Implementierungsspezifikation vorgeschlagen wurden.

Programme, die anhand der OpenGIS Spezifikation erstellt wurden und diese erfüllen, können sich diese Übereinstimmung vom OGC zertifizieren lassen. Die OGC bietet auf ihren Internetseiten eine Liste der zertifizierten Programme an, die bereits knapp 400 Produkte umfasst.

4.2 Nexus

Das Nexus-Projekt (siehe [16]) ist ein fächerübergreifender Forschungsbereich an der Universität Stuttgart mit dem Ziel, ein gemeinsames Weltmodell für eine Vielzahl verschiedener ortsbasierter Anwendungen zu erstellen. Dazu soll ein offenes allgemeines Kontextmodell erschaffen werden, in das verschiedenste lokale Modelle, sowohl für den Innen- als auch Außenbereich, integriert werden können.

Grundmodell ist hierbei eine dreidimensionale Modellierung der physischen Welt, nach der Klassifikation von Abschnitt 2 also ein geometrisches Kontextmodell. Dieses Modell wird, wie in Abschnitt 2.5 beschrieben, um zusätzliche Informationen und Umgebungsmodelle erweitert. Zentrales Problem ist dabei die Integration verschiedener Umgebungsmodelle, Sensoren und Datenspeicherungen, um das Gesamtmodell skalierbar und auf verschiedenste Anforderungen anwendbar zu halten.

Damit Daten aus verschiedensten Umgebungsmodellen und Datenquellen verwaltet und gespeichert werden können, muss zunächst eine Klassifikation dieser Daten vorgenommen werden, da unterschiedliche Arten von Daten unterschiedlich gehandhabt werden müssen. Diese Daten werden daraufhin aus vielen verschiedenen Datenquellen in ein Gesamtmodell integriert. Die folgenden Abschnitte sollen Ansätze des Nexus-Projekts zu diesen Problemstellungen vorstellen.

4.2.1 Klassifikation von Kontextdaten

Um die verschiedenen Arten von Kontextdaten, die in das Gesamtmodell integriert werden, einordnen zu können, werden von Grossmann et al. [6] zwei grundsätzliche Skalen vorgeschlagen:

- *Update Rate*: Hierbei werden die Daten danach klassifiziert, wie oft sie aktualisiert werden. Auf der einen Seite gibt es statische Daten, die sich nie oder kaum ändern (z.B. die Position eines Gebäudes), zum anderen können sich andere Daten sehr schnell ändern (etwa die Position eines mobilen Objekts).

- *Usage for Selection*: Hier werden die Daten nach ihrer Zugriffshäufigkeit eingeteilt. Informationen wie die Farbe eines Raumes werden beispielsweise viel seltener zugegriffen als etwa die Position eines Anwenders.

Abbildung 10 zeigt die Beispiele von verschiedenen Kontextdaten und ihre Einordnung in die beiden Skalen.

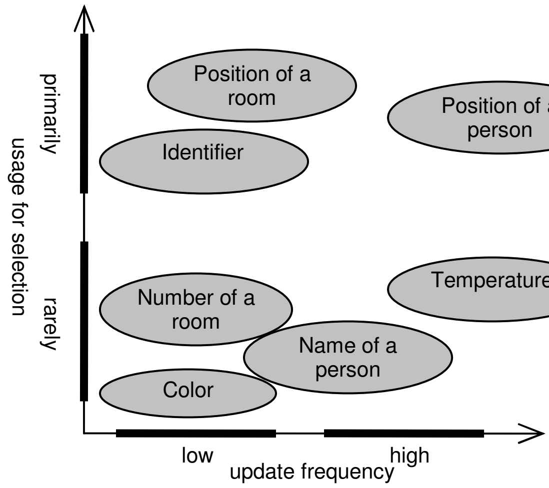

Fig. 10. Beispiele für verschiedene Arten von Kontextdaten (aus Grossmann et al. [6])

Abhängig von der Klassifizierung der Daten gibt es unterschiedlich geeignete Arten, sie zu speichern. Bei Daten, die sehr oft aktualisiert werden, ist es etwa nicht nötig, sie ausfallsicher in einer Datenbank zu speichern (es sei denn, die Daten sollen dauerhaft gespeichert werden, um eine Historie herzustellen). Würde das System abstürzen, würde der Sensor bei einem Wiederanlauf schnell aktuelle Daten liefern, außerdem wären die gesicherten Daten nach einem Neustart höchstwahrscheinlich veraltet.

Sehr oft zugegriffene Daten sollten dagegen so gespeichert werden, dass ein Zugriff möglichst schnell möglich ist. Dies kann durch geeignete Speicherstrukturen geschehen, etwa dadurch, dass die Daten nicht über verschiedene Server verteilt werden. Für derartige Daten eignet sind auch besonders das Anlegen von Indizes.

4.2.2 Integration verschiedener Datenquellen

Da die Nexus-Plattform zum einen viele verschiedene Datenquellen und lokale Kontextmodelle gleichzeitig nutzen und zum anderen auch bei sehr großen Datenmengen skalierbar bleiben soll, unterstützt es die Verteilung der Kontextdaten auf viele unterschiedliche Systeme, so genannte *Context Server*. Ein Context Server stellt einen Lieferanten einer Menge von Kontextdaten dar. Dies kann eine Datenbank oder auch nur ein Sensor sein, abhängig von der Art der Kontextdaten und den Anforderungen an ihre Speicherung (siehe Abschnitt 4.2.1).

Um als Context Server für die Nexus-Plattform zu dienen, muss der Server ein bestimmtes XML-basiertes Format zum Austausch von Daten erfüllen (siehe unten). In [6] wird dargestellt, wie unterschiedliche Komponenten mit Wrappern als Context Server integriert wurden. Beispielsweise wurden sowohl ein Datenserver zur Speicherung großer Mengen von statischen geographischen Daten als auch ein eingebettetes System zur Aufnahme von Sensorinformationen hinzugefügt.

Einen Überblick über die Architektur der Nexus Plattform gibt Abbildung 11.

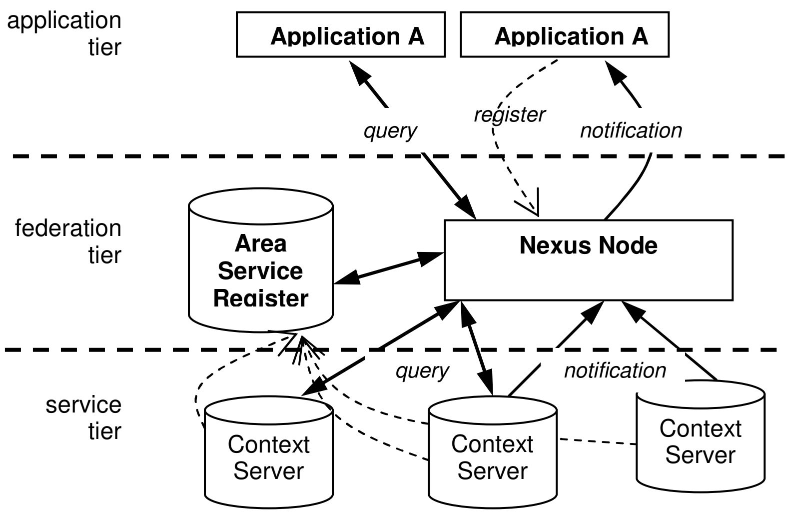

Fig. 11. Die Architektur von Nexus (aus Mitschang et al. [13])

Als Middleware zwischen den Context Servern und den Anwendungen, die auf die Kontextinormationen zugreifen möchten wurde in der Nexus Plattform die Föderationsschicht eingefügt. Sie hat die Aufgabe, Anfragen der Anwendungen transparent zu beantworten. Im Einzelnen bedeutet dies unter Anderem:

- Alle verschiedenen Schemata der Server werden in ein Gesamtschema integriert.
- Bei Anfragen wird entschieden, welche der Context Server die entsprechenden Daten zur Verfügung stellen und die Anfrage an diese Server weitergeleitet.
- Context Server können an der Federation Schicht registriert und abgemeldet werden.
- Die Konsistenz der Daten wird gewährleistet, etwa in dem Fall, dass zwei verschiedene Context Server Daten über dasselbe physische Objekt enthalten.

Der letzte Punkt beinhaltet die Identifizierung von Objekten in verschiedenen Datenbanken, die auf dasselbe physische Objekt verweisen. In Volz und Walter [21] ist der Ansatz beschrieben, wie im Nexus Projekt solche Relationen zwischen Objekten erzeugt und automatisch auf die zughörigen Schemata übertragen werden.

Als gemeinsames Austauschformat für Anfragen und Daten wurden nach Grossmann et al. [7] spezielle XML-basierte Formate geschaffen. Beispielsweise dient zur Übertragung von Anfragen die *Augmented World Query Language* (AWQL). Sie unterstützt räumliche Abfragen wie etwa die Suche nach der Position eines Objektes oder nach allen Objekten in einem bestimmten Gebiet.

4.2.3 Speicherung räumlicher Informationen

Die einzelnen Kontextdaten des Weltmodells liegen bei Nexus in den verschiedenen Kontextservern vor. Diese können wie beschrieben je nach Klassifikation der enthaltenen Daten einfache Sensoren sein, die nur die aktuellen Werte zurückliefern, aber auch Datenbanken, die Daten persistent speichern. Wichtig hierbei ist, dass auch räumliche Informationen gespeichert werden können.

In Schwarz et al. [18] wird ein Beispiel zur Speicherung räumlicher Informationen im Rahmen von Nexus gegeben. Hierbei werden geometrische Daten mithilfe einer Java-Bibliothek gespeichert.

Die Speicherung beruht in diesem Fall auf dem Framework *Java Topology Suite* (JTS, siehe [20]), welches die *Simple Feature Specification* von OpenGIS (siehe Abschnitt 4.1) implementiert. JTS verfügt über eine Reihe von geometrischen Basistypen (Punkt, Linienzug und Ebene) und bietet auch einfache Funktionen an, wie etwa die Berechnung, ob zwei verschiedene Gebiete ineinander liegen, sich schneiden oder disjunkt sind.

Wie schon in Abschnitt 4.2.2 beschrieben, ist ein wesentlicher Aspekt von Nexus die Integration verschiedener Datenquellen und Kontextmodelle zu einem Weltmodell. Jedes der einzelnen Kontextmodelle kann über ein eigenes Koordinatensystem verfügen (z.B. auf Länge und Breite basierende bei GPS-Sensoren und kartesische Systeme in Innenräumen), deshalb muss bei der Implementierung der Speicherung die Möglichkeit vieler verschiedener Koordinatensysteme berücksichtigt werden. JTS kennt nur ein einziges globales Koordinatensystem. Daher wurde in [18] eine Bibliothek entwickelt, die JTS um die Unterstützung mehrerer Systeme erweitert. Hierbei führen alle geometrischen Objekte zusätzlich eine Referenz auf das Koordinatensystem, in dem sie sich befinden.

Sollen nun Berechnungen zwischen Objekten in verschiedenen Koordinatensystemen durchgeführt werden, wird zunächst einmal ein gemeinsames Koordinatensystem berechnet, in welches die Koordinaten der Objekte dann umgerechnet werden (etwa durch Verschiebung, Drehung und Stauchung). In diesem gemeinsamen System werden die Objekte dann an JTS übergeben, wo die benötigten Berechnungen durchgeführt werden.

Zur persistenten Speicherung der Daten wurden ebenfalls OpenGIS implementiert. Sie erfolgt mit den Text- bzw. XML-basierten Standards WKT (Well Known Text) und GML (Geographic Markup Language). In beiden Fällen wurden die Standards um zusätzliche Referenzen erweitert, die zu jedem Objekt das zugehörige Koordinatensystem angeben.

5. Zusammenfassung

In dieser Arbeit wurden verschiedene Zugriffsstrukturen und Kontextmodelle für kontextbewusste Systeme vorgestellt. In derartigen Systemen ist vor allem der räumliche Kontext, das heißt die Position verschiedener fester und beweglicher Objekte im Raum, von Bedeutung.

Kern eines jeden kontextbewussten Systems ist ein Kontextmodell. Dieses sagt aus, welche Arten von (räumlichen) Kontextdaten wie gespeichert werden und wie sie untereinander und mit anderen Daten verknüpft sind. Hierbei gibt es zwei verschiedene Ansätze: Das geometrische Modell, welches Orte über Koordinaten speichert und das hierarchische, welches mit Enthaltensbeziehungen zwischen verschiedenen Orten arbeitet. Darüber hinaus existieren weitere Mischformen, die die Vorteile beider Ansätze zu vereinigen suchen.

Einfache Indexmodelle zum Zugriff auf gespeicherte Daten reichen hier nicht aus, da diese nur eindimensionale Indizes effizient verwalten können, räumliche Koordinaten aber zwei- oder dreidimensional sind. Aus diesem Grunde wurden verschiedene Indexstrukturen zum mehrdimensionalen Zugriff erläutert.

Die Anwendung in räumliche Daten verarbeitenden Systemen wurde anhand von zwei sehr unterschiedlichen Beispielen vorgestellt. Das erste Beispiel beschreibt OpenGIS als Standard für viele verschiedene Systeme, die geografische Daten verarbeiten und darstellen können. Zum Abschluss wurde im zweiten Beispiel Nexus als umfassendes System zur Speicherung und Nutzung verschiedenster Kontextdaten präsentiert.

Literatur

[1] Brown, P. J., Bovey, J. D., Chen, X., *Context-aware Applications: from the Laboratory to the Marketplace*, IEEE Personal Communications, 4(5), 58-64, 1997.

[2] Brumitt, B., Shafer, S.: *Topological World Modeling Using Semantic Spaces*. Workshop Proceedings of Ubicomp 2001: Location Modeling for Ubiquitous Computing, Sept. 2001.

[3] Cheverst, K., Mitchell, K., Davies, N.: *Design of an Object Model for a Context Sensitive Tourist GUIDE*. Proceedings of the International Workshop on Interactive Applications of Mobile Computing (IMC98), Rostock, Germany, November (1998)

[4] Domnitcheva, S.: *Location Modeling: State of the Art and Challenges*. Workshop on Location Modeling for Ubiquitous Computing (2001).

[5] Finke, R. A., Bentley, J. L.: *Quad Trees A Data Structure for Retrieval on Composite Keys*. Acta Informatica 4, 1-9 (1974)

[6] Grossmann, M. et al: *Efficiently Managing Context Information for Large-scale Scenarios* (2005). Proceedings of the Third IEEE International Conference on Pervasive Computing and Communications.

[7] Grossmann, M., Leonhardi, A., Mitschang, B., Rothermel, K.: *A World Model for Location-Aware Systems* (2001) Aus Informatik Informatique 5 2001

[8] Guttman, A.: *R-Trees: A Dynamic Index Structure for Spatial Searching*. Proc. ACM SIGMOD International Conference on Management of Data 1984

[9] Hu, H., Lee, D-L.: *Semantic Location Modeling for Location Navigation in Mobile Environment*. Proceedings of the 2004 IEEE International Conference on Mobile Data Management (MDM'04)

[10] Jiang, C., Steenkiste, P.: *A Hybrid Location Model with a Computable Location Identifier for Ubiquitous Computing*. Aus: G. Borriello and L.E. Holmquist (Eds.): UbiComp 2002

[11] Leonhardt, U.: *Supporting Location-Awareness in Open Distributed Systems*. Dissertation, Department of Computing, Imperial College, London, May 1998.

[12] Markl, V.: *MISTRAL: Processing Relational Queries using a Multidimensional Access Technique*. Dissertation Universität München, 1999

[13] Mitschang, B. et al.: *Federating Location-Based Data Services* (2005). Data Management in a Connected World 2005: 17-35. Springer Berlin / Heidelberg

[14] Open Geospatial Consortium: OpenGIS Specifications (Standards). http://www.opengeospatial.org/standards

[15] Pascoe, J.: *Adding Generic Contextual Capabilities to Wearable Computers* (1998). Proceedings of the 2nd IEEE International Symposium on Wearable Computers

[16] Rothermel, K. et al: *NEXUS. Spatial World Models for Mobile Context-Aware Applications*. Sonderforschungsbereich 627 Annual Report (2004)

[17] Schindler, G., Metzger, C., Starner, T.: *A Wearable Interface for Topological Mapping and Loclization in Indoor Environments* (2006). Aus Hazas, M., Krumm, J., Strang, T. (Eds): Location- and Context-Awareness 2006, LNCS 3987, pp 64-73, 2006

[18] Schwarz, T., Höhnle, N., Grossmann, M., Nicklas, D.: *A Library for Managing Spatial Context Using Arbitrary Coordinate Systems* (2004) Second IEEE Annual Conference on Pervasive Computing and Communications Workshop 2004

[19] Tarumi, H. et al.: *Kotohiragu Navigator: An Open Experiment of Location-Aware Service for Popular Mobile Phones* (2006) Aus Hazas, M., Krumm, J., Strang, T. (Eds): Location- and Context-Awareness 2006, LNCS 3987, pp 48-63, 2006

[20] Vivid Solutions: Java Topology Suite. http://www.vividsolutions.com/jts/jtshome.htm

[21] Volz, S., Walter, V.: *Linking Different Geospatial Databases By Explicit Relations* (2004). Geoinformation Science Journal, Vol. 6, No. 1, 41-49.