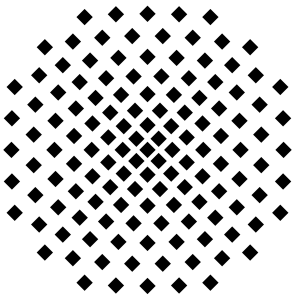


Universität Stuttgart

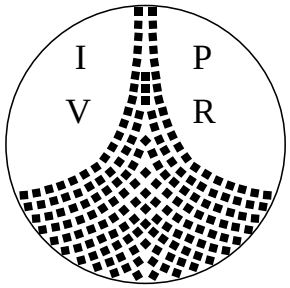
Fakultät Informatik

Prüfer:	Prof. Dr. B. Mitschang
Betreuer:	Dipl. Inf. D. Nicklas
begonnen am:	10.7.2000
beendet am:	10.1.2001
CR-Klassifikation:	C.2.4, D.2.2, H.2.1

Diplomarbeit Nr. 1870

Modellierung der Augmented World in Nexus

Jens Meßmer



Institut für Parallele und Verteilte
Höchstleistungsrechner (IPVR)
Abteilung Anwendersoftware
Breitwiesenstr. 20-22
70565 Stuttgart

AS

KURZFASSUNG

In unserem täglichen Leben sind wir es gewohnt Informationen wahrzunehmen, die an physisch existierende Orte geknüpft sind. Orts- und raumbezogene Applikationen für mobile Benutzer ahmen diesen Umstand nach, indem sie die Organisation ihrer Daten geografisch handhaben. Das Ziel von NEXUS ist es, eine offene und globale Plattform zu entwickeln, die als Middleware für diese Applikationen dient. Da die Informationen an physisch existente Orte gebunden sind, müssen wir die reale Welt simulieren, um die Verbindung zwischen Information und räumlicher Umgebung aufzuzeigen. Zusätzlich haben wir zur Aufgabe die Informationsdaten, die zu den real existierenden Orten in Beziehung stehen, in die simulierte Welt zu integrieren. Folglich benötigen wir ein Weltmodell, welches durch virtuelle Information angereichert ist, um all die Daten darzustellen. Dieses Weltmodell wird Augmented World Model genannt. Es bildet einen zentralen Teil unserer offenen globalen Plattform.

In dieser Diplomarbeit beschäftigen wir uns mit der Entwicklung eines Augmented World Models für NEXUS. Hierfür müssen wir mehrere Teilaufgaben bewältigen: Etablierung einer Terminologie, welche die gängigsten Begriffe definiert und konsolidiert, Untersuchung der Anforderungen an solch ein Weltmodell, Betrachtung von Problemen und Möglichkeiten im Zusammenhang mit dem Augmented World Model, und ein Beweis der Funktionsfähigkeit des erstellten Augmented World Models.

ABSTRACT

In our daily life, we are used to perceive information that is associated to physical locations. Location- and spatial-aware applications for mobile users emulate this fact by organizing data in a geographical manner. The goal of NEXUS is to develop an open global platform serving as a middleware for these applications. Since the information is bound to physical locations we have to simulate the physical world in order to point out the connection between information and spatial environment. Additionally, we have to integrate the information data that is related to these physical places into the simulated world. Thus, we are in need of a world model augmented by virtual information in order to represent all the data. This world model is called Augmented World Model. It forms a central part of our open global platform.

In this thesis we deal with the development of an Augmented World Model for NEXUS. For this purpose we have to perform several tasks: establishing a terminology that provides definitions for those technical terms that occur most often, examining the requirements to such a world model, considering problems and options in connection with the Augmented World Model, and proving the workability of the generated Augmented World Model.

ACKNOWLEDGEMENTS

I would like to thank Professor Mitschang for his remarkable support concerning technical and technological issues. His ideas have given me new impulses several times.

I also want to thank Daniela Nicklas for the fruitful discussions and the feedback. Her outstanding helpfulness has made her support me even during her holidays.

Last I would like to express my sincere thanks to the whole NEXUS research group, especially to Martin Bauer, Steffen Volz, Alexander Leonhardi and Matthias Großmann, for their help and their technical comments.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

One doesn't discover new lands without consenting to lose sight of the shore for a very long time. *(Andre Gide)*

This chapter informs about the objectives of the thesis, its structure, and the motivation.

1.1 PROJECT DESCRIPTION

The research team of the project group NEXUS develops an open global platform for spatial and location-aware applications for mobile users. Hereby the Augmented World Model is of special interest (see section 1.2: “Motivation and Objectives”). The Augmented World Model is a representation of physical existent and virtual objects, which can be static or mobile and connected to external information such as web pages.

The objective of this master thesis is the modelling of the Augmented World. Its objects (e.g. buildings, mobile users, vehicles, virtual information towers, etc.), attributes (position information, representational data, object data, etc.) and functions have to be described with suitable languages (such as UML).

Since NEXUS is built by a research team, several groups are working on the project, which results in different requirements to the Augmented World Model. Consequently, one important task of the master thesis is to communicate with the several groups and to integrate the miscellaneous requirements into one common model.

The necessary design steps are:

- *Analysis of requirements:* What information must be contained by the model
- *Examination of various modelling languages:* Selection or development of a suitable language for describing the objects
- *Conception of the Augmented World Model’s classes*
- *Ascertainment of the model’s feasibility:* The feasibility has to be shown by an exemplary scenario of an example application

1.2 MOTIVATION AND OBJECTIVES

In our environment, there is a lot of information that is associated with spatial actualities or that is organized in a geographical manner. Objects realize people advancing, which results in actions intended to support the persons (traffic lights change to green, doors open, lights turn on, etc.). Also possible are objects that display information on certain occasions (displays on a station’s platform which announce the next train). We are used to perceive environmental information organized in a location-bound way.

But up to now, only few information system applications take into account organizing information in a location- or spatial-aware way. [HOHL ET AL. 1999] claims that typical areas of application were navigation, fleet management and map-oriented geographical information systems. Reminders that are triggered by objects and locations (a mobile user that passes a shoe shop is reminded of buying a pair of shoes), tourist guides, etc. can be other possibilities. Since we are not able to envi-

sion all the possible applications, extendibility should be guaranteed. Therefore an open system is the aspired goal of NEXUS.

We need a world model that describes all the objects occurring in the physical existent world which are relevant to the applications for implementing location and spatial-aware information systems. But a world model that describes solely the real world's perceivable objects is not sufficient because we also need the information that is connected to them. Therefore we need a world model describing the real world, augmented by additional virtual objects describing connected information. So the term of an "Augmented World Model" came into existence.

The applications (navigation, etc.) already mentioned consequently have to contact the Augmented World Model for receiving the data they have to manage. Now we want to enable all applications to use the same data originating from the same world model. Therefore we need not only an open but also a global platform. Thus the applications are able to share the data contained by the world model.

Again we are not able to guarantee the completeness of the Augmented World Model. Possibly applications that are additionally and subsequently implemented need objects not provided by the Augmented World. Two contradictory requirements, the global system on the one hand and the extensibility on the other hand, result in the necessity of finding a compromise. So a definition of an Augmented World's class hierarchy supports standard applications like navigation with necessary classes. This class hierarchy is well-known and enables applications to get global access to all data. But there is also still room left for additional class definitions embedded in the existing class hierarchy so that the system can be extended.

The major task that has to be worked out in the following is consequently the design of such a class hierarchy.

1.3 APPROACH AND OVERVIEW

We started with defining and consolidating terms. We had to acquaint ourselves with the terms and expression regularly used within NEXUS. Additionally, we had to define some terms which were not elaborated then. We also had to understand the system, and hence considered the architecture with regard to the placement of the Augmented World Model. The results of this phase can be seen in **CHAPTER 2:** "Environment".

Next we carried out interviews with members of the NEXUS research team whereby requirements had to be established. In **CHAPTER 3:** "Requirements" we first dealt with conceptional application-specific requirements. The next topic to examine were the consequential needs. In our case, object orientation resulted of application-specific aspects. Finally, we studied the system-driven requirements mostly based on the components' demands.

In **CHAPTER 4:** ”The Augmented World Class Schema” the design of the class hierarchy is shown. We came to the presented design by paying attention to already existing concepts concerning navigation, object-oriented design, and the design of world models with regard to the previously established requirements.

Afterwards we investigated implementation aspects such as distribution and resulting problems, consistency, and the complexity of the navigation network. We also examined possibilities for exchanging data by using an Augmented World Modeling Language, whereby possible existent exchange formats (XML, GML, VRML) were considered. On the basis of established selection criteria, one format was selected and examined more precisely for proving its suitability. Additionally, we worked out a small example scenario serving as an illustration for the interaction between the Augmented World Model and the NEXUS system components. The results of these topics can be gleaned from **CHAPTER 5:** ”Implementation Aspects of the Augmented World Model”.

CHAPTER 6: ”Example Scenario” illustrates the workability of the classes situated on a higher level in the hierarchy by dealing with diverse small scenarios based on the NEXUS use cases. The way for obtaining a class hierarchy that works was made up of a cycling design process, changing the design of the example scenario objects for the design of the class hierarchy and vice versa. By using this iterative process, we tried to reduce the number of errors to a tolerable value.

CHAPTER 7: ”Conclusion” offers a retrospective view over the work and concludes the thesis by evaluating its usefulness. It also provides a list of issues to be done in the future.

CHAPTER 2

ENVIRONMENT

Deep in the human unconsciousness is a pervasive need for a logical universe that makes sense. But the real universe is always one step beyond logic. *(Frank Herbert, Dune)*

In this chapter the project´s environment is presented. It can be understood as a summary of already defined and known issues before the design of the Augmented World Model takes place. Here we want to define some terms and expressions that are commonly used in NEXUS. Then we take a look at the system components and the fields they are involved in. Afterwards the architecture with regard to the Augmented World Model is presented.

2.1 IDEA BEHIND NEXUS

The goal of NEXUS is to provide an open global platform for location- and spatial-aware applications with mobile users [BAUER 2000], [HOHL ET AL. 1999]. Since many existing services have to be re-implemented for each application that uses location- and spatial-awareness, we want to provide a generic platform enabling a cooperation among different applications. The world model frames the central unit of this common platform. Generally speaking, the world model is an assembly of projected regions of the physical world, which are augmented by virtual objects. Thus, the term "Augmented World Model" comes into existence.

Additionally, NEXUS provides us with the possibility of presenting the well-known physical reality in a virtual format however not as perfect as the real world. But if we are able to emulate any real world object, fictions of a nearly perfect virtual reality as described in [WILLIAMS 1998] for instance, are becoming closer and closer to realization.

2.2 TERMS AND DEFINITIONS

For a further work we want to define the most basic technical terms used in NEXUS. Regularly used terms are Augmented Area and Augmented Area Model, Augmented World and Augmented World Model, Augmented World Standard Class Schema, and NEXUS object.

2.2.1 AUGMENTED AREA MODEL

In [HOHL ET AL. 1999] an Augmented Area is defined as a spatially limited region, e.g. the area of a city, the inside of a building. It consists of virtually and substantially existent objects. Augmented Areas may be overlapping or even include each other. Augmented Area Models are data representations of Augmented Areas that are machine-readable in a uniform format (see Figure 2-1). Augmented Areas also include information on the mobile user situated in the appropriate area.

Another characteristic of Augmented Areas is stated in [HOHL ET AL. 1999]. There, the term of a "living model" is used. That means that changes in the state of an Augmented Area are automatically propagated to the corresponding models. The other way round, changes in the state of an Augmented Area Model might trigger some actions that change the real world state (see Figure 2-2).

As a matter of course, the above articulated characteristics are still valid. But continuing thoughts make it necessary to re-phrase the definition taking into consideration recent studies.

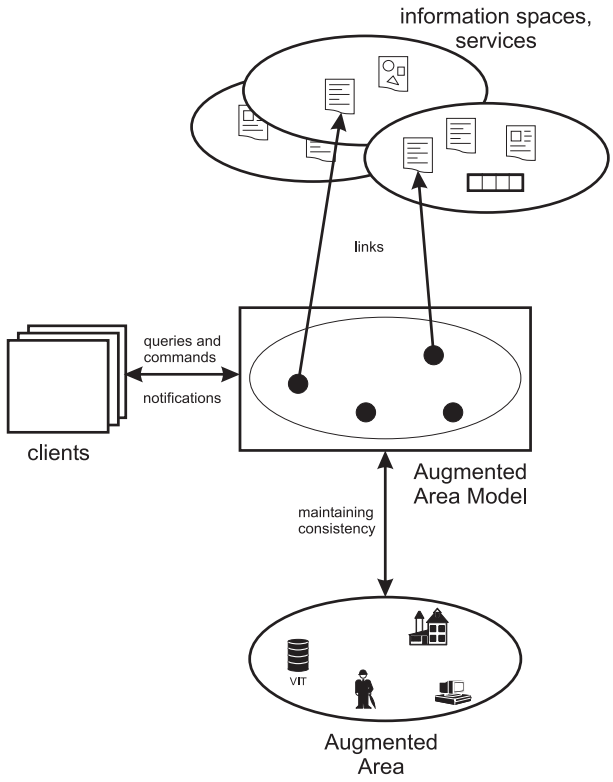


FIGURE 2-1: Augmented Area Model (taken from [HOHL ET AL. 1999])

An Augmented Area is a spatially limited region, that consists of virtual and physical existent objects. Areas can be overlapping and include each other. An Augmented Area Model might be the projection of an Augmented Area. In special cases it might also be an assembly of logically coupled objects, such as 3D representations of mobile objects. Changes in the state of an Augmented Area's physical objects should result in an automatic update of the model's objects. Changes in the states of augmented world objects might trigger events that attend to alter the states of the corresponding substantial objects. The data an Augmented Area Model contains, is always consistent to itself,

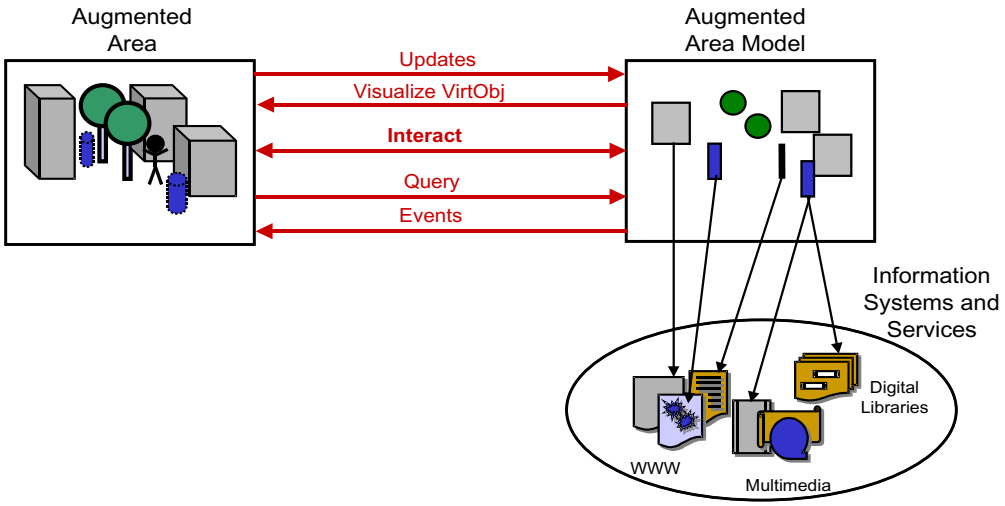


FIGURE 2-2: Interaction between Augmented Area and Augmented World Model (taken from [COSCHURBA ET AL. 2000])

whereas consistency among several Augmented Area Models cannot be plainly guaranteed.

2.2.2 AUGMENTED WORLD MODEL

In [HOHL ET AL. 1999] and [BAUER 2000] the Augmented World Model is defined as a global integration of several single Augmented Area Models. The models are comprehended as distributed representations of regions of the physical world

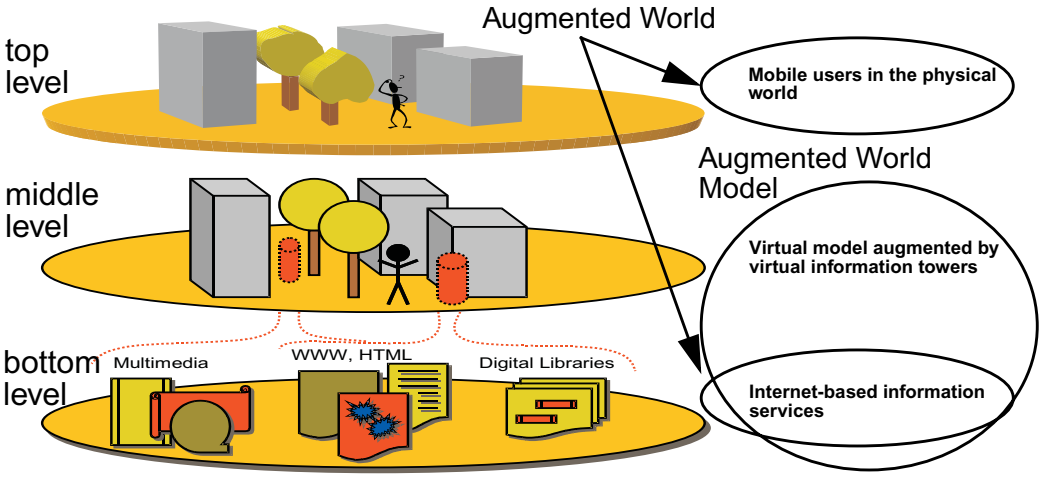


FIGURE 2-3: **The Augmented World's Composition**

whereby each model is described in a uniform format. Models can be augmented by virtual objects. The uniform description of the Augmented Area Models helps to extend the Augmented World Model and to integrate new models similar to the integration of a new WWW-Server into the Web.

Now we want to define this term more precisely for this thesis. Firstly, we have to distinguish between Augmented World and an Augmented World Model. An Augmented World includes the physical world augmented by virtual objects, formed by so-called situated information spaces and services. The Augmented World Model is the projection of the physical world into a virtual one. This mapping is performed by describing physical objects in a computable and uniform way: the Augmented World Modeling Language. Since situated information spaces and services are virtual by now, there is no need to transform them in a machine-understandable format again. They are already part of the Augmented World Model in their machine-readable manner.

In Figure 2-3 the top level pictures the physical world. The middle level shows the projection of the physical world which can be called a virtual model. The bottom level depicts situated information spaces. Thus, a union of the middle and the bot-

tom level can be comprehended as the Augmented World Model whereas a union of the top and the bottom level can be perceived as the Augmented World.

2.2.3 AUGMENTED WORLD STANDARD CLASS SCHEMA

The Augmented World Standard Schema is formed by the classes defined in this thesis. Additionally, there is the Augmented World Extended Schema consisting of classes yet undefined. The Extended Schema offers the possibility of developing the existing model. Take an organization that wants to build their own NEXUS application implementing their internal telephone network composition. In the existing Standard Schema there are no telephone classes provided. But these classes prove to be necessary for implementing the organization's project. So they have to extend the existing schema by inheriting from StaticObject, for instance, and by creating their own application that does the desired activities like automatic call diversion for example.

We have to anticipate some of the requirements if we want to explain the necessity of an Augmented World Standard Class Schema.

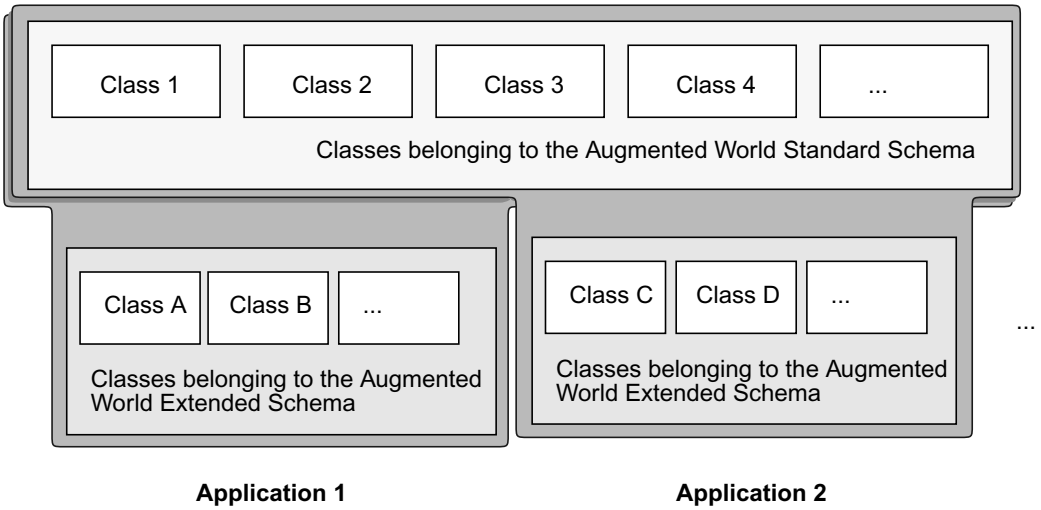


FIGURE 2-4: **Applications Sharing Classes in NEXUS**

The Augmented World Model consists of instantiated NEXUS object classes. An accumulation of instantiated classes constitutes an Augmented Area which describes a self-consistent subset of the Augmented World. As a result we have a distribution of classes in the Augmented World Model. Applications can share classes that are already defined in this work. Classes additionally set up can only be used by applications knowing of their existence. Applications can only rely on the

existence of the Standard Augmented World Schema classes and consequently only use them (see Figure 2-4). Classes, additionally defined by third party organizations, can only be used by extended or re-implemented applications.

2.2.4 NEXUS OBJECT

In NEXUS an object is an instantiated class of the Augmented World Class Schema. It may be a member of the Augmented World Standard Class Schema or a member of the Extended Class Schema, respectively. An object consists of attributes, but it has no methods or operations in NEXUS. The class type NEXUS Object is the uppermost class within in the Standard Class Schema. Any other object used in NEXUS has to be at least transitively inherited[1] by NEXUS Object.

A NEXUS Object embodies an element of the Augmented World Model. Thus it can be either a representation of a physically existent object or a virtual object. Derived, i.e. inherited objects, differ in their type and can be divided in virtual or real objects, spatial or non-spatial, mobile or static.

In the following we want define the characteristics mentioned last stronger:

2.2.4.1 SPATIAL OBJECT

Spatial objects consists of bodies, planes, lines or points and can be either virtual or real. Since they exist in virtual reality as embodied objects they must be represented in some way. The representation data can be based on the object´s spatial extension, or it can be an image. Spatial objects have a position which can be either fixed (when they are static) or variable (when mobile).

2.2.4.2 STATIC OBJECT

A static object is a spatial object and can be either virtual or real. It is not supposed to move around frequently because of its static property. The positions of static objects are fixed. The position can depend on another object if stuck on it (post-it on car) or placed on it (table on ship), whereby the other object can be mobile. In this case the static object´s position is fixed to a position relative to the moving object, nonetheless.

2.2.4.3 MOBILE OBJECT

A mobile object is a spatial object with a variable position and can be either real or virtual. Mobile objects in NEXUS are equipped with sensors sending their current position so that the Location Service is able to localize them.

1. “Transitively” in the sense of “not directly”.

2.2.4.4 REAL OBJECT

A real object is part of the physically existent Augmented World (see Figure 2-3). In NEXUS this implies that the object has to be spatial.

2.2.4.5 VIRTUAL OBJECT

A virtual object is a non-visible and non-physical object of the real world. Spatial objects, that are not part of the physical world but simulations of objects usually occurring in the real world, are also virtual objects, e.g. virtual information towers and virtual post-its.

2.3 EXAMPLE SCENARIO

In the following we want to present an example scenario to give the reader an impression of NEXUS' functionality.

In our scenario we imagine a childless family called Fisher. Mr Fisher works as a manager in a global firm. His wife does the household and is engaged in local politics.

Usually Mr Fisher gets up at 7.00 am in the morning to go to work. But on this special morning he has an appointment with a colleague working for another company. His company is 500 miles away in another town Mr Fisher does not visit very often. Therefore Mr Fisher has to travel on the motorway, and eventually, after arriving at the town, he has to find a route to the office where the meeting takes place. For this purpose a portable NEXUS client can be very helpful. It calculates a way to the place Mr Fisher is going to meet his colleague and displays it on a map. Additional textual directions assert that Mr Fisher always takes the right course.

When he meets his colleague they want to take lunch first. So they have to find a restaurant close to their present position. NEXUS delivers them a list of nice restaurants in their surroundings. It is even able to present them a menu. When Mr Fisher and his colleague agree upon a restaurant, NEXUS is again able to direct them to their destination.

Near the restaurant the back of an old mansion catches Mr Fisher's interest. Wondering about its history and today's function he points at the building with his telepointer. According to the NEXUS client, once the building was property of a well-known citizen and industrialist. But today it is a small but comfortable hotel with 25 rooms. Sample rooms can even be viewed by a web page that is also connectable by NEXUS. For Mr Fisher does not know how long his business affairs will take, he precautionary orders a room for tonight. And another time the NEXUS client assists him by making the reservation and debiting the money from his account.

In the evening after the meeting Mr Fisher wants to go out. He has no idea what to do at this place. At this very moment he receives a message on his NEXUS client about remaining tickets for the local theatre. Consequently, he reserves a ticket and prepares for the evening event.

This scenario provides an insight in the abilities of NEXUS. Of course NEXUS is capable of many more things but in this scenario only a selection of important functions has been presented. The main characteristics we are able to deduce from the scenario above are as follows:

- *navigation:* navigation to the office and to the restaurant
- *location- and spatial-aware queries:* queries for finding the closest restaurant or for retrieving information about the old mansion
- *accessing external information sources:* connecting the hotel's web page
- *virtual services:* reserving the hotel and the theatre tickets
- *geographical addressed messages:* message about left-over theatre tickets

Now we are aware of NEXUS' abilities but we do not know anything about the functionality and the structure of a NEXUS platform. Thus, we deal with the system components and the architecture before we address ourselves with the design of a world model.

2.4 THE NEXUS SYSTEM COMPONENTS

Before we are able to describe the architecture we should first characterize the single components. So we want to introduce the reader to the tasks and objectives of the system components in the following subsections.

2.4.1 LOCATION SERVICE

The Location Service is responsible for tracking mobile users [LEONHARDI & KUBACH 1999]. It periodically updates the position, speed and direction by using location sensors. The Location Service is distributed among several Location Servers, whereas each Location Server manages the data of a specified area. So, the data of users entering the area of a certain Location Server A are handled by it, whereas leaving users are handed over to a neighboured Location Server B.

2.4.2 SPATIAL MODEL SERVICE

The Spatial Model Service provides data for representing spatial objects. It also operates queries for functions like “does object A touch object B” or “lies object A within the extensions of object B”. In other words: the Spatial Model Service stores the spatial data for NEXUS objects. Just like the Location Service the Spatial Model Service is distributed among various Spatial Model Servers. Each Spatial Model Server handles the data for a specific region. But it might also be responsible for a group of objects that are not spatially adjacent, such as scale drawings for ships, trains and planes. So it is thinkable that all trains of a certain type are managed by one Spatial Model Server.

2.4.3 AREA SERVICE REGISTER

Basically, the Area Service Register is in charge for assigning areas to objects, i.e. an object type returns a list of Augmented Areas containing objects of this type. The other way round it can also assign object types to areas (see Table 2-1). An Augmented Area is returned when entering an object’s position, whereas an input of an

given position → Augmented Area
Augmented Area ↔ contained object type

TABLE 2-1: **Assignments of Area Service Register**

Augmented Area results in a list of contained object types.

2.4.4 NAME SERVICE

The Name Service maps the names of important objects to object identifiers. Nonetheless this will prove to be a very important function since object identifiers in NEXUS enclose information about the area they are located in. Thus we are able to return an area to a given object name. In the case of ambiguous names a list of identifiers is returned.

2.4.5 GLOBAL FEDERATION

The Global Federation offers a way to uniformly access the world model, namely the Augmented World Query Language. Using the Area Service Register, it integrates the data from Spatial Model Servers, the Location Service and hypothetical other external information sources.

2.4.6 EVENT SERVICE

The Event Service [BAUER 2000] constitutes another noteworthy component within NEXUS. Here events are used for triggering actions by sending notifications to clients. Usually clients have to register for an event before they are able to receive a notification. Clients can be mobile users, components or applications. Applications and components might even be able to change an object's state.

2.4.7 HOME REGISTER

In general, data for mobile objects is stored on the location server. But usually the spatial model server has to deliver the data for queries where attributes like the extension or the graphical representation of mobile objects are needed. In most cases queries on mobile objects have to be repeated frequently due to their changing position. A special server called "home register" is used to prevent the spatial model server from overloading. It stores the data that would have been stored on the spatial model server, if the object had been a static object.

2.5 NEXUS APPLICATIONS

Applications are used by clients utilising the services of NEXUS. Applications are modules that communicate via the Global Federation with components. They should be executable on a NEXUS server and on a client, respectively. Due to the extendibility of NEXUS it must be possible to integrate applications additionally offered by third party organizations besides the existing standard applications like navigation.

Clients can be portable (notebooks, personal digital assistant, wearable computer) or static. Due to the main intention of NEXUS to support mobile users, we have to take care that all important applications are runnable on portable clients. Since portable clients might be restricted concerning memory capacity and processing time, each application must be able to work with the limited abilities of those clients. [SCHÜTZNER 1999] provides a classification of portable clients.

2.6 THE AUGMENTED WORLD MODEL WITHIN THE NEXUS-ARCHITECTURE

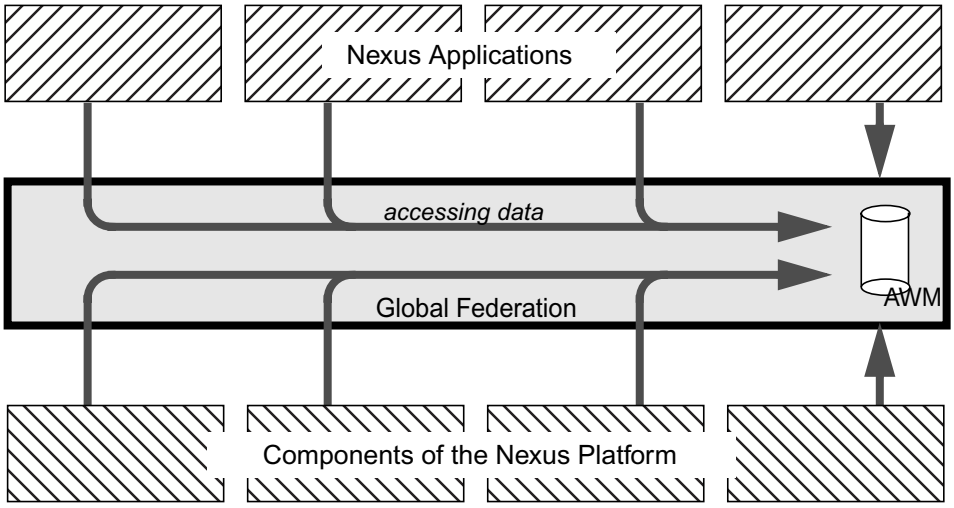


FIGURE 2-5: **The Logical Place of the Augmented World Model**

The world model in NEXUS is referred to as Augmented World Model. Considering the Augmented World Model as a non-distributed data set, containing only one model, it would be sufficient to place the Augmented World Model on the same level as the central component responsible for interactions of applications and components accessing data (see Figure 2-5). Thus, the communication will be performed mainly by the central component called Global Federation.

The intention of NEXUS was to build an extendable and distributed system. Institutions, organizations and governments should be able to provide Augmented Area Models of data concerning their interests. These models may be incomplete and overlapping to other models. It is mainly up to the provider to avoid inconsistencies between models that might occur due to overlapping data.

Regarding the components there is the Location Service that is responsible for position data of mobile objects. The Spatial Model Service stores the data of static objects. It handles all the 2D, 2.5D[1] and 3D information of static objects. The exact graphical representation that is needed for map creation for example, is completely stored on a distributed Spatial Model Service. That means, when a user starts a query for a route from Stuttgart Hauptbahnhof to Killesberg Messe (compare [USE CASE 2000]), the Location Service transfers the localization information to the Glo-

1. 2.5D means models with two dimensions whereby objects may have an additional height information.

bal Federation demanding the corresponding mapping data from the Spatial Model Service.

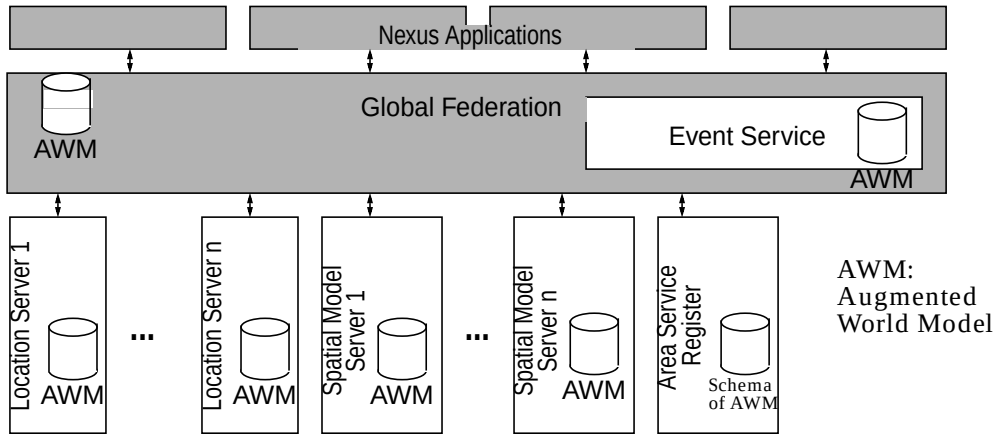


FIGURE 2-6: Distribution of Different Augmented World Models over the NEXUS Platform

Assuming that there are lots of one-way streets on the way from Stuttgart Hauptbahnhof to Killesberg Messe and that our navigation tool should be able to deliver a textual route description as well, we definitely need a place where traffic and transportation information is stored. We also need data for representing buildings, streets, and all the other things one needs for map creation. The Augmented World Model is able to deliver such information. But the query for this information has to be started by the Global Federation (the global interface between all components), representational data for handling the query will mainly be retrieved from the Spatial Model Service. The Area Service Register is responsible for assigning an appropriate Spatial Model Server to a given query from the Global Federation. The data stored in the Spatial Model Service is distributed to several Spatial Model Servers, where every server manages the data for a certain region or subject group, such as the region of Stuttgart West or the climate data of Middle Europe. Therefore, the Area Service Register accesses the Augmented World Model, too, because areas are just objects with attributes like extension, position, and a responsible Spatial Model Server.

Other possible use cases (e.g. 2, 3) implicate the necessity of querying Augmented World Model information from the Location Service component. The best solution is to distribute the Augmented World Model information to the responsible components (see Figure 2-6).

2.7 SUMMARY

In this chapter we have provided an introduction to the terminology of NEXUS. We have learned more about the components and their cooperation. We have also shown that the Augmented World Model is partitioned to Augmented Areas that are distributed among several components. The interface for accessing data from the Augmented Areas is delivered by the Global Federation.

CHAPTER 3

REQUIREMENTS

Everything in software changes. The requirements change. The design changes. The business changes. The technology changes. The team members change. The problem isn't change, per se, because change is going to happen; the problem, rather, is the inability to cope with change when it comes. (Kent Beck, Extreme Programming Explained)

Now that we are familiar with the terminology and the architecture, we are able to address ourselves to the demands. So this chapter discusses the requirements of the Augmented World Model. Therefore the concepts of object oriented design, the NEXUS architecture and underlying components are examined. The requirements are the fundament for generating an Augmented World Standard Class Schema.

3.1 CONCEPTIONAL REQUIREMENTS

Conceptional requirements can be raised by examining the intention and the objectives of NEXUS. Apart from the results that stem from application-specific ideas we also consider requirements that arise from consequential concepts such as object orientation.

3.1.1 APPLICATION-SPECIFIC REQUIREMENTS

The most important necessities emerge from the objectives of the NEXUS project. The purpose of spatial-aware applications is to alleviate the daily actions of mobile users in the real world. Changes in the real world should also be visible in the virtual world, and changes in the virtual world should be visible in the real world. The last statement seems to be confusing at first glance, so we want to give an example: If a user switches off a light in a room by using a NEXUS interface, the light should also go out in the real world. So the first and most important demand is to provide the possibility of modelling all elements that appear in the real world. The second prerequisite comes along with the first one: changes in the state of either the real world or the virtual world have to be projected in the real or virtual world counterpart, respectively.

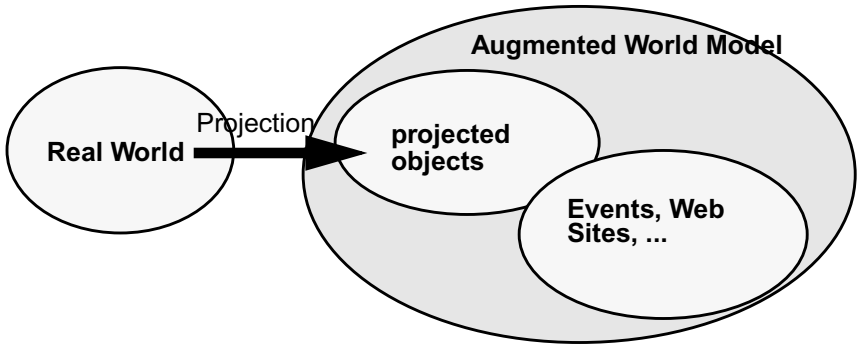


FIGURE 3-1: **Augmented World Objects**

Additionally, there are objects we want to access in NEXUS, too, and that are not part of the real world such as web sites, events, etc. These objects combined with the projected objects result in the Augmented World. So we also have to implement these additional virtual objects.

But there would be no use of all these data if no applications processed it. The applications are able to query databases that contain the Augmented World Model. In

NEXUS four kinds of queries can be performed (see Table 3-1): ordinary queries, spatial-aware queries, time-aware queries and combined queries.

kind of query	description	example
standard database queries	Select all contacts that are born before January 1, 1980.	`select * from CONTACT where date_of_birth < "1980-01-01"`
location- and spatial-aware queries	Select all Chinese restaurants in my surroundings. The example on the right is based on the specifications in [OGC 99-049].	`select object.name, object.id from ChineseRestaurant AS object where object.position within PolyFromText('circle around current_pos', SRID)`
time-aware queries[a]	Select all ice hockey games that will take place in the next three days.	`select * from GAMES where (date < addDay(CURRENT_DATE, 3)) and (date > CURRENT_DATE)`
combined queries	Select all tubes in my surroundings that will leave in the next five minutes.	`select * from Tubes AS object where (object.position within PolyFromText('circle around current_pos', SRID)) and (departure < addMinutes(CURRENT_TIME, 3)) and (departure > CURRENT_TIME)`

TABLE 3-1: Different Kinds of Database Queries in NEXUS

a.time-aware queries will be possibly implemented in future releases

As we already know the Augmented World Model is distributed on Augmented Area Models. Augmented Areas can be created by different people or organizations. The data within the Augmented Area Models are shared by a variety of NEXUS applications which might have been implemented by different people or organizations as well. We need a definition of a standardized class hierarchy to guarantee the interoperability between all offered data objects and applications. Since this class hierarchy will never be complete, we also have to provide the opportunity of extending classes by inheritance. The last requirement leads us to the idea of using an object oriented-concept.

3.1.2 REQUIREMENTS DUE TO OBJECT ORIENTATION

In almost all cases concerning simulations and representations of real world objects the appropriate design approach is object orientation. In NEXUS we use an Augmented World Model that consists of all kinds of data objects. It is an abstract formation of hierarchically structured objects. Similar to their real world counterparts, they possess attributes and varying instances.

Significant characteristics for objects are the concepts of inheritance, hierarchical dependencies as a consequence, and the existence of attributes whose values depend on the state of the object instance. Additionally, we need identifiers for determining objects uniquely within the database representation of the Augmented World Model.

3.1.2.1 Classes and Instantiation

The class concept is essential for object oriented approaches. A class is a model from which objects are instantiated, and attributes and operations are identical to all objects of this class. They only differ in the states that are determined by the values stored in the attributes.

3.1.2.2 Unique Object Identifiers

Usually we need a globally unique identifier (or universally unique identifier) for differentiating between data objects or elements which can have the same attribute (or field - in the database meaning) values. In object oriented programming we need identifiers for objects in order to refer to them and for setting up inheritance. In location- and spatial-aware applications both concepts are combined: identifiers refer to objects and they serve as primary keys in the database.

3.1.2.3 Information Hiding

In several programming languages access restrictions to objects are implemented. That means that some information stored in attributes might be private to the containing object itself and cannot be provided to other objects. There are also public attributes that can be accessed by any other object. In NEXUS the introduction of this concept seems to be valuable, too.

3.1.2.4 Inheritance and Polymorphism

Similar to object oriented programming, attributes are inherited from superclasses. It might be possible that certain rules are becoming necessary for controlling reasons. Probably some inherited methods (operations) of superclasses do not fit for their subclasses and should be redefined. Therefore redefinition of operations (and thus polymorphism) should be possible. The common concepts of inheritance include the question about multiple inheritance. By using multiple inheritance, a subclass can inherit from two super classes. Multiple inheritance comes along with several problems and complications and is not unconditionally and definitely necessary. In most cases it is possible to redesign a multiple inheritance so that only the common inheritance remains [KHOSHAFIAN & ABNOUS]. That is why we do not use multiple inheritance at all.

3.1.2.5 GENERICITY

A very important feature of location- and spatial-aware applications is the genericity of its objects. Genericity in this sense allows us to use generic classes for lots of subclasses. The difficulty is to produce a class-tree structure being that generic that any real world object which is relevant to possible NEXUS applications fits into a place in the resulting tree.

3.2 SYSTEM-DRIVEN REQUIREMENTS

Another worthy source for finding requirements is offered by the NEXUS system architecture or by its components, respectively. Of course, the analysis is strongly dependent on the NEXUS architecture.

3.2.1 DISTRIBUTIONAL ASPECTS

The intention of NEXUS is to build an extendable and distributed system. Institutions, organizations and governments should be able to provide Augmented Area Models of data concerning their interests. We already discussed the preconditions that are due to distribution in an application-specific context. Now we want to consider architectural intricacies. The propagation of changes from the real world to the virtual world and vice versa makes it necessary to keep duplicated data within models consistent. Duplication of data comes along with overlapping area models. As a matter of course, we should at least offer data consistency for objects that might propagate changes. We have to know more about the organization of NEXUS classes in order to resolve inconsistencies. In section 5.1.3: “Consistencies within Functional Dependencies” we attend to inconsistencies in more detail.

3.2.2 STRUCTURE OF THE LOCATION SERVICE

The location service is responsible for storing the actual positions of all mobile users, that agree with revealing their location. It is also able to estimate movement vectors and future positions of mobile users. Therefore additional data like way points is required, the communication network must not be overcharged, and the sensory devices must provide the necessary data, i.e. a tuple as described in section 3.2.4: “Sensory Components”.

Difficulties arise when the placement of data is taken into consideration. What happens to objects that are neither throughout static nor mobile? A virtual post-it for example can be stuck on a mobile user, and consequently, would be mobile, too. Nevertheless, the object itself is not able to move on its own, and seems to be rather

static. As a consequence, it is sufficient to link the post-it to the mobile object and to observe the linked object. A similar question is dealing with the representational data of mobile objects. Originally, it was not clear whether to save this information on a Location Server or on a Spatial Model Server. An agreement was obtained to store the data on a special server called Home Register.

According to use case 17, there are also mobile objects with large extensions like ships, trains, or plains. Within this mobile objects navigation (use case 4) should be possible as well. For achieving this goal, we intend of using an extra Augmented Area for these objects. We attach a coordinate system to such an Augmented Area Model (which is handled as object, too). If we want to determine the position of mobile users within this mobile object in absolute (global) coordinates and in relative (to the object) coordinates, we have to have to know the absolute position of an vector of the mobile object. Therefore we need at least two sensors within the mobile object, where the sensors are fixed points[1] on the local object coordinate system. We also need an origin for the object coordinate system. The position of the sensors must not be changed without changing the positions in the Augmented Area of the mobile object, because then any calculations of mobile users aboard will not be correct.

3.2.3 Structure of the Spatial Model Service

A spatial model is a copy of a real world environment with static objects like buildings, streets, trees, parks, etc. The main task is to make all these objects visible to its users. But a model without any life in it would be boring and not as interesting as a virtual world with representations of its users. It is the task of the Location Service to localize mobile users, but the representation of all data (static or mobile) is on the Spatial Model Service.

3.2.3.1 Tasks of Spatial Model Service

Generally speaking, the Spatial Model Service is responsible for spatial objects regardless of them being virtual, real, static or mobile. Important is the evaluation of spatial queries like checking the predicate of the spatial event “user enters shopping centre”. Typical actions that are performed by the Spatial Model Server are spatial analysis (including the evaluation of spatial queries), navigation (e.g. routing within a city), querying attributes (receiving information about a touristy sight for instance), manipulation of spatial objects (changing attributes), registering[2] to spatial events, and handling of graphical representations. For more information see [Schützner 1999].

1. In GIS these points are also called tie points.
2. “Registering” means that a service has to specify the events it wants to offer. Only offered events are available for clients requesting data from the corresponding service (see [Bauer 2000]).

The use of one or more coordinate systems is a necessity for locating mobile objects in a spatial environment. Using a uniform coordinate system enables the collaboration of Location Service and Spatial Model Service for determining the position of mobile users in an Augmented Area. In NEXUS we use WGS84 for global or large-scaled models, respectively, and Gauß-Krüger for local (small-scaled) models at the moment. It might be possible that other coordinate system will be used later on. Therefore the Augmented World Model must offer a generic concept for coordinate systems.
Besides the coordinate systems already mentioned, a local cartesian coordinate systems referring to single objects is necessary. These coordinate systems are used for buildings or large-scaled mobile objects like ships.

3.2.3.2 DATA ORGANIZATION

The spatial data can be described by using the Augmented World Modeling Language that is an XML derivation. In the case of the Spatial Model Service, it mainly builds up on the Geography Markup Language (another XML derivation) as described in [OGC 00-029]. All the spatial data is stored in a special database that can be accessed by using the Augmented World Query Language. In later phases of NEXUS the Augmented World Query Language can be extended or transformed into an SQL92- or SQL99-like syntax [OGC 99-049]. The OpenGIS Consortium does already provide an object-oriented structure for storing graphical representations of objects (see Figure 3-2). This serves as a basis for modelling spatial structures like extensions of buildings, streets etc.

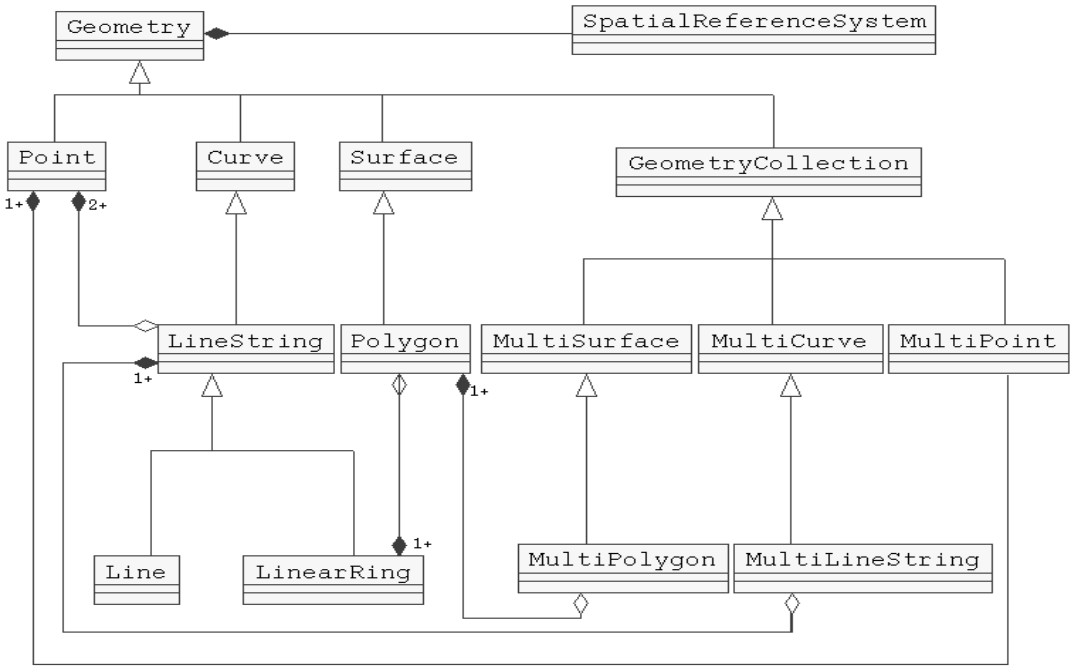


FIGURE 3-2: **Geometry Class Hierarchy**

3.2.4 SENSORY COMPONENTS

Sensory components in NEXUS are dealing with real world data that is projected into the augmented world. Since data produced by sensors is often dependent on the sensor's position, all sensors in our system are treated as an object for being able to determine its position (the position can be given by the Augmented Area provider - it is not meant that all sensors are able to localize themselves). Therefore every sensor object has a position attribute. Additionally, we need certain attributes, such as coordinate systems for determining positions, degrees for measuring temperatures, photoreceptors for recognizing light intensity etc.

3.2.4.1 DIFFERENT TYPES OF COORDINATE SYSTEMS

At the moment we are using three kinds of coordinate systems (shown in Table 3-2) in NEXUS. Because of the extensibility of the NEXUS system, other coordinate systems should be possible, too. The scale of a coordinate system is not initially determined, for the scale is only important for the representation and not the localization of objects.

The first of the three types of coordinate systems is the object coordinate system. The scope of such coordinate systems is restricted to one object. This object can be a city, a building, a mobile object or whatever. But object coordinate systems make only sense for small objects. They are mainly for indoor-use.

Regional coordinate systems are used for regions, cities and states. In Germany, the coordinate system mainly used is Gauß-Krüger for regional areas. But regional coordinate systems differ from country to country, therefore other systems should also be possible. This is no problem for the modelling of the Augmented World. We simply store a coordinate system for each object that is used within this object. A difficulty might arise when a mobile user moves within an object with a special coordinate system, and a NEXUS application using a different coordinate system wants to know the position. Then we have to find the global position of that user. If we want to determine the global position of a mobile object within a special coordinate system we only need a function that converts the coordinate data from one schema into the other.

The global coordinate systems are referring to the whole world. By using them, we can determine any point on the world with a three tuple of longitude, latitude and altitude.

object coordinate systems	regional coordinate systems	global coordinate systems
(x, y, z)	Gauß-Krüger for small scaled Augmented Area Models like regions (x, y, height)	WGS84 for large scaled models (longitude, latitude, and altitude (not NN))
...	...	...

TABLE 3-2: **Different Coordinate Systems Used in NEXUS**

3.2.4.2 SENSORY INFORMATION FOR MOBILE OBJECTS

The accuracy of sensory components is very important for location awareness. The quality of the localization information will not meet all of the NEXUS requirements without being able to determine the position of objects accurate to the metre indoors, 10 or 20 metres outdoors. Of course, it is nearly impossible to determine the position of mobile objects accurate to a metre without using an immoderate effort nowadays. In [WANT ET AL. 1992] an infrared based active badge system is used in an in-building environment. And mobile users can be located accurate to rooms. Additional to the location information one receives a likelihood for finding somebody at the given location in the form of a percentage.

In [BAHL & PADMANABHAN 1999] radio frequency instead of infrared based systems is used. Their system produced a mean error distance of about 3.5 metres for tracking mobile users. Thus, the accuracy of radio frequency based systems is likely similar to infrared ones. It is not possible to determine the position more exact than to rooms.

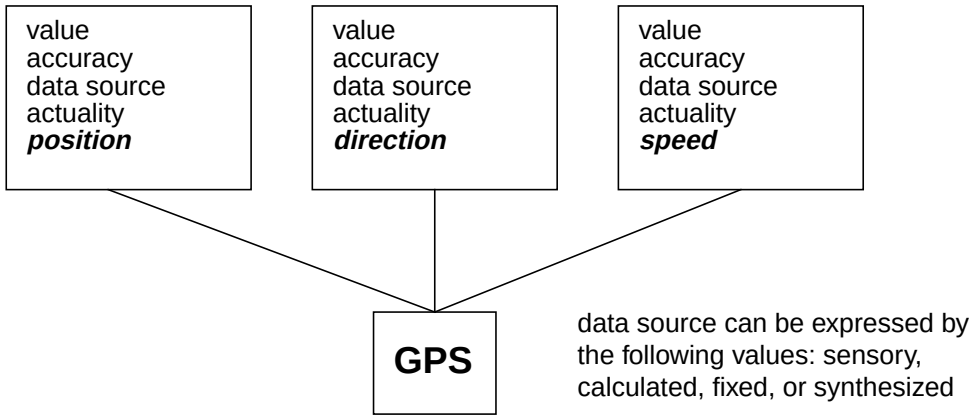


FIGURE 3-3: **Attributes for Position Information**

Outdoors, the Global Positioning System, short GPS, is the common used localization system. A research about error behaviour or GPS and DGPS can be seen at [DANA 2000]. There, the accuracy for DGPS is stated between two and five metres and between 12 and 30 metres for GPS[1]. In NEXUS, we use GPS or DGPS, respectively, for outdoor positioning. Considering a mobile object, we need the quality of the provided positioning information. Three parameter arrays are necessary (see Figure 3-3), i.e. the medium accuracy of the corresponding sensory component, the type of data source, which can be “sensory”, or “calculated”/”synthesized”/ ”fixed” for non-sensory sources, respectively. Non-sensory sources refer to calculated or estimated positions. Updating the position every few seconds would be

1. Originally the accuracy of GPS was between 40 and 100 metres. But on May, 2nd 2000 the “Selective Availability” was turned off and GPS became more exact by one third. “Selective Availability” (S/A) was controlled by the US Department of Defense to limit accuracy for non-U. S. military and government users.

very time consuming and is not necessary, for the position of a mobile object can be estimated (calculated) by using the direction, velocity, and actual position [LEONHARDI 1999]. Moreover, a time-stamp of the last update (actuality) is needed for calculating the next update time.

Synthesized sensory sources draw their data from two or more sensors. This can be a combination of a pedometer and GPS for example. The results are expected to be more exact by using synthesized sources. Fixed sources define that the value has been entered manually and is neither calculated nor estimated nor sensory determined.

3.2.5 ORGANIZATION OF THE EVENT SERVICE

An important service provided by location- and spatial-aware applications forms the event management system. It is responsible for delivering event notifications following event occurrences.

In NEXUS events are described in form of predicates, filters, and actions (see [BAUER 2000]). Predicates are conditions that have to be fulfilled for triggering the execution of the action. Filters are some additional conditions that may be linked to queries to the NEXUS platform (e.g. “send me a message for every fifth visitor entering my shop”, or “I am interested in the event only between 10 am and 11 am”). The action part is going to be used for information about how and where the event notification is going to be delivered.

The Event Service does not have a fixed, predefined set of events. Instead, new events can be integrated dynamically, and the availability may differ between objects and areas. Thus, two Location Servers may offer a different set of events.

Predicates are instances of predicate templates. Attributes of a predicate template are a name, a service (Location Service, Spatial Model Service, etc.), the server offering the service, a variety of variables (e.g. extension of area) for handling the conditions, and delivery information (e.g. who gets the message, quality of service). Besides the predicate template, we also need a notification object. Here, all the delivery information for the registered client is stored, namely a reference to the corresponding event, the event name (onEnter, onLeave), the server, the point of time of the occurrence, the scope period of the notification and some additional information the client wants to have (attributes of entering or leaving objects).

3.2.6 COMMUNICATION INTERFACES

In NEXUS communication for mobile users is accomplished by using radio technologies, such as UMTS, wireless LAN, and GPRS (based on GSM). Considering UMTS, for example, different nets emerge depending on the net provider that can offer dif-

ferent services. Every network is divided into cells which will deliver their data with different qualities. Regarding multi-story buildings or scyscrapers, there are multiple cells bedded one upon another. Thus, a three dimensional coordinate system is needed in order to describe the position of cellular base stations. When users are in border regions of overloaded cells, a handover to neighboured, less used cells, is performed automatically. Therefore, the workload behaviour of the cells is dynamically organized.

For mobile users, predictions about movement might be done by analysing the actual direction and position and comparing it with previously acquired behaviour patterns. The communication components are only able to deliver the position, the mentioned predictions have to be done by the Location Service.

Besides the predictions about user movements, other interesting evaluations can be performed. The current workload of a cell is a useful indicator for high- respectively low-quality transfer rates. Therefore, the workload can be estimated by watching the routing queues or monitoring other connection parameters.

The communication interfaces can be accessed by an API offering a consistent methodology. Functionalities of singular net types that might be missing can be compensated.

An important fact is the location awareness for NEXUS. The Communication interfaces are responsible for providing this data. On the other hand, users might not want to make their current position public. Therefore, security as an important topic leads us to the idea of distributing localization data on various servers and of encrypting this information. Nevertheless, it is doubtful whether all users will trust NEXUS and are willing to reveal their position. It is of course possible to switch off the positioning transfers for privacy reasons. So mobile users cannot be localized.

3.3 SUMMARY

In this chapter we considered the requirements emerging for the Augmented World Model. Sources for establishing essentials were conceptional on the one hand and system-driven on the other hand.

Conceptional requirements dealt with the intentions and purposes of NEXUS as a location- and spatial-aware application. There we found out that we have to provide a possibility for modelling all objects that exist in the real world augmented by virtual data objects. We also remarked that changes in the state of either the real world or the projected copy must be performed in the corresponding counterpart, too. Another main intention of NEXUS was the purpose of location- and spatial-aware applications: to perform location- and spatial-aware queries on a database. The next conceptional requirement was the creation of a set of standardized classes

situated on a higher level in the hierarchy. This led us to the idea that object orientation might be an appropriate approach for designing the Augmented World Model.

The system-driven necessities dealt with distributional aspects. One major problem is consistency, which emerges due to the distribution of the Augmented World. We noticed that we were not able to discuss this topic at once because we have to define the class schema first. Then requirements attributed to the system components were established.

CHAPTER 4

THE AUGMENTED WORLD CLASS SCHEMA

There are two ways of constructing a software design: One way is to make it so simple that there are obviously no deficiencies and the other way is to make it so complicated that there are no obvious deficiencies. *(C.A.R. Hoare)*

This chapter defines a set of classes that all belong to the Augmented World Standard Schema Classes. These classes are dedicated to provide users with a standardized top-level set of classes to model Augmented Areas for NEXUS applications. The definitions are composed in the Unified Modeling Language. Additionally, the describing text clears up obscurities and ambiguities.

4.1 APPROACH FOR THE DESIGN

For setting up the classes as described in the sections of this chapter we applied the following procedure: First of all, the requirements had to be defined. For this purpose, we interviewed NEXUS project team members, asking them about their ideas regarding the realisation of the NEXUS system. The interview included questions that were already focused on an object oriented approach. Thus we asked for classes that are likely to be necessary. Along with these classes, the appropriate attributes emerged. Additionally, we wanted to know for both, classes and attributes, how important the elements might be for the implementation of the planned applications. These declarations helped us to decide in conflicting cases, which class or attribute was to be realized.

The question about functional dependencies between classes was also important. This topic will be discussed in section 5.1.3: “Consistencies within Functional Dependencies”, for it already deals with implementation aspects. The results were marked down on a list which served as a first base. Moreover, [USE CASE 2000] gave us an additional impression of the system’s intended objectives.

The requirements led us to the current design whose main characteristics are briefly summarized for a recollection: First, there is the object-oriented approach which makes sense if one thinks of the intention of NEXUS: reconstructing the world in a digital form. The easiest approach to describe the real world is naturally object orientation because the physical world is composed of objects. Second, the distribution of the Augmented World Model makes the definition of a set of a classes’ upper layer necessary (see section 2.2.3: “Augmented World Standard Class Schema”).

The class definitions were developed using the Unified Modeling Language (see [PAGE-JONES 1999] for further reading). Parallel to this process, we tried to confirm the correctness of the emerging classes by implementing example scenarios (see chapter 6: “Example Scenario”). Similar to a software life-cycle, changes in exemplary objects led to changes in the underlying classes and vice versa. Difficulties arose from the resulting complexity of object hierarchies of rather simple example scenarios. Even simple tasks such as the navigation of mobile users that have to be directed around the corner proved to be very complex. With regard to implementation aspects like execution time, we added solutions that reduced the demanded objectives to a minimum but also required only a fraction of the objects previously needed. Here, the declarations concerning the importance of attributes proved to be very helpful for the reduction. An example for complex and fast navigation can be found in section 6.2.2: “Navigation to Defined Places”.

4.2 NEXUS CLASSES

In this section we want to define the NEXUS Augmented World Standard Schema classes. Class definitions and generalizations are shown in the following figures. By definition, attributes of superclasses are inherited by their subclasses. Therefore they are not specified in the subclasses anymore. Whenever the type of an attribute can be a list of expressions, we use the following notation: attribute: (value1 | value2). The declaration <<optional>> means that the corresponding attribute is not required and can be left out. NEXUS classes all have in common that they do not define any operations or methods. They are only data objects, i.e. only methods defined in applications are able to change the attribute values.

4.2.1 NEXUS OBJECT TYPES

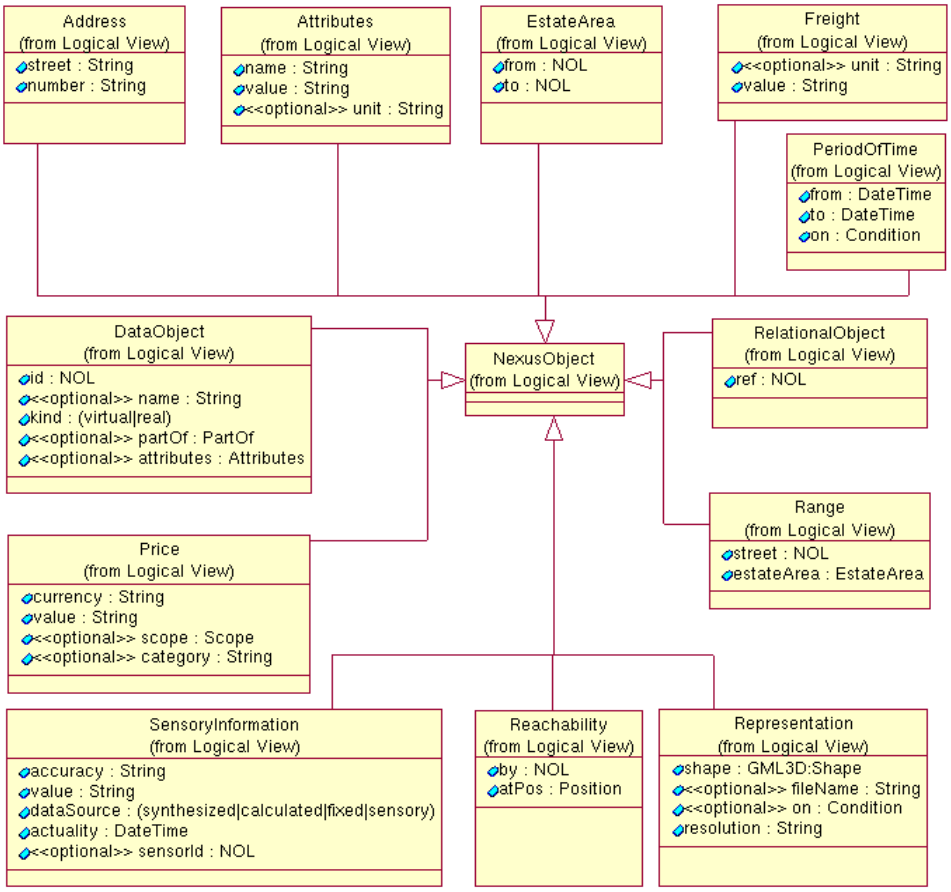


FIGURE 4-1: NEXUS Object Structure

NEXUS object types build the top layer of NEXUS classes. Since these classes are mostly not used as actual instances we call them types for now. They have in com-

mon the absence of an identifier. That means that they are attributes or attribute types in the Augmented World Model with one exception: DataObject, which builds the top level class for all inheriting objects that are ultimately no types (see section 4.2.2: "NEXUS Data Objects").

4.2.1.1 NEXUS OBJECT

The NEXUS Object is the parent object of the Standard Class Schema. Any other object that might be used by NEXUS applications must inherit from this class. Since this class has no attributes, there are no restrictions to possibly inheriting classes. Of course one should be aware of the disadvantages of inheriting from objects at the top levels of the class schema. The more standard attributes can be inherited by an extended class the more interoperability to existing applications is guaranteed. We suggest to inherit from classes at the most distinctive level.

4.2.1.2 ADDRESS

Address is an address type for buildings where a *street* name and a house *number* can be specified.

4.2.1.3 ATTRIBUTES

Attributes permit attribute values for other objects to be defined. Usually a *value*, a *unit* for the value (tons, centimetre, etc.) and an *identifier* for the attribute are needed. *Unit* is not a required attribute.

4.2.1.4 COORDINATESYSTEM

We need an identifier in order to refer to coordinate systems that can be either regional, global or related to objects. Coordinate systems can be named, which makes sense for well-known global and regional systems like Gauß-Krüger or WGS84. Object coordinate systems mostly refer to buildings or large mobile objects in which navigation is needed. Ships, planes and trains are examples of those large-scaled objects. Assuming objects in a building whose positions have to be determined, the *coordinate system* attribute of the *position* attribute, which is again an attribute of the assumed object, determines the position by referring to the underlying coordinate system non-ambiguous.

4.2.1.5 ESTATEAREA

EstateArea defines a range for streets. By using this class type, subsets of navigation edges are able to declare which buildings or estates, respectively, are adjacent to them. In any case the *from* and *to* attributes have to be NOLs (NEXUS Object Locators) of the type Estate or of its subclasses.

4.2.1.6 FREIGHT

Freight is an important attribute for a reservation service. With this class type the kind of service represented by ServiceObject can be specified more precisely. The intention of Freight is to store the maximum capacity of transportation vehicles like buses, trains, ships, etc. for certain freight types like luggage, people, goods, etc.

4.2.1.7 PERIODOFTIME

PeriodOfTime (see Figure 4-2) contains classes that are dependent on a period of time. These can be temporary blocked objects (streets by traffic jams or road works for example) or scopes of objects (lifetime of predicates).

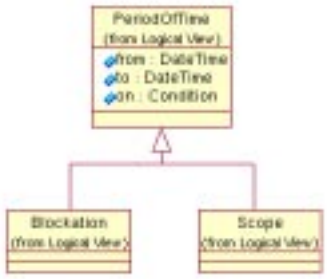


FIGURE 4-2: **DateTime Types**

4.2.1.8 PRICE

Something like a *currency* and the amount (*value*) of money have to be specified for NEXUS applications that are engaged in transferring money. Therefore Price can be used. Since some prices are dependent on seasonal demands, a *scope* can be declared. Categories are another possibility for defining prices (e.g. zones in transport compounds).

4.2.1.9 RANGE

Range is part of the NavigationEdge class and defines the represented street and the range of adjacent estates to it. The type of street has to be a NOL of Road (Road is the NEXUS class personating a street). It facilitates the search algorithms for buildings if it is known which buildings are connected by a navigation edge. Then it is not necessary to examine each connected building object for the *address* attribute but it is sufficient to search the entries of navigation edge for a requested address.

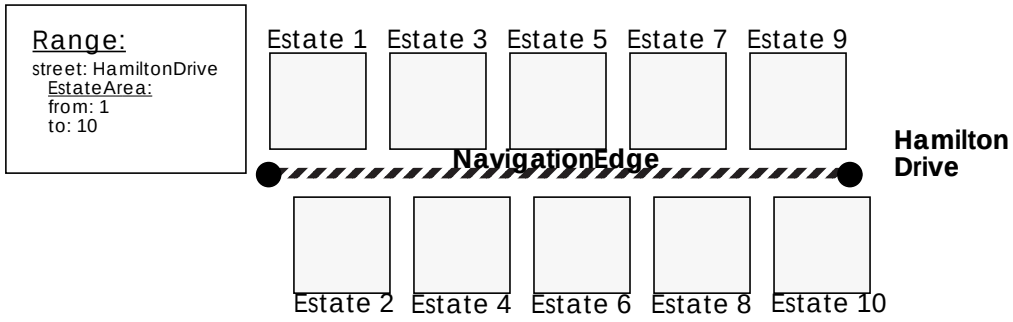


FIGURE 4-3: **Range Example**

4.2.1.10 REACHABILITY

Reachability is very important with regard to the Augmented Area view. Consider a provider of an Augmented Area that describes a city. Then, a citizen wants to add his home to the Augmented World Model. He has to establish connections to the surroundings that are represented by the city's Augmented Area for integrating his home. Since it is very likely that the owner does not have the user permissions for changing the navigation system (NavigationNodes and Edges), he has to find another way to connect his house to the surroundings. He can define a reachability with the Reachability class for his own building that is hosted by his own server and managed by his own Augmented Area.

The *by* attribute in Reachability defines the NavigationEdge by which the building shall be reachable and *atPos* defines the position on the NavigationEdge.

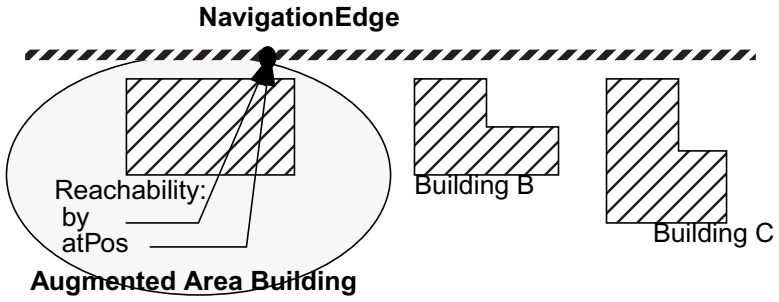


FIGURE 4-4: **Reachability Example**

4.2.1.11 RELATIONALOBJECT

RelationalObject is a superclass of relations in general. PartOf, BelongsTo, HeldBy and SticksOn inherit its attribute, namely *ref*. *Ref* specifies the reference to the parent object of a relation and can be of any NOL type. RelationalObject gives users the possibility to inherit from it and to extend the existing class schema.

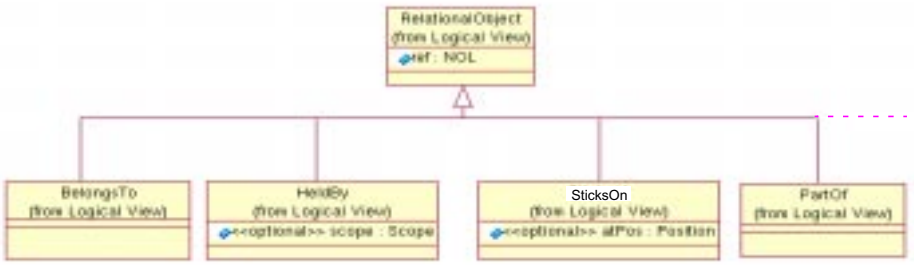


FIGURE 4-5: **Relational Objects**

Anticipating section 5.1.2: “Augmented Areas: Cooperating versus Competitive” we have to take care of providing hard-coded links between identical objects. RelationalObject can easily be used for inheriting a class called IsIdenticalTo that is responsible for defining objects representing the identical real world object. Since we would like to support only cooperative Augmented Area Models we did not implement IsIdenticalTo in the following.

4.2.1.12 PARTOF

PartOf is one of the most powerful NEXUS classes. Any hierarchical part-of structure has to be built with this relation class. To give an example: building is part of its estate, which is part of a lot, which is part of a city, which is part of an area like a state for example and so on. Universe should occur at the uppermost end of each partOf relation, whereby Universe has to be the same for all objects that are connected in any way. PartOf can also be a reflexive relation for some classes. NavigationEdge for example can be divided into a subset of NavigationEdges.

4.2.1.13 BELONGSTO

BelongsTo expresses the ownership in contrast to possession that is expressed by HeldBy. The person who rents a house is referenced by HeldBy for instance, whereas the person who owns it by law is referred to by BelongsTo.

4.2.1.14 HELDBY

Since HeldBy, contrary to BelongsTo, depends on a period of time (usually things are not rented for a lifetime), *scope* is an additional attribute of HeldBy.

4.2.1.15 STICKSON

Some special items like post-its, virtual or not, can be stuck on other objects. The stuck objects do not have a position themselves! The position has to be calculated by referring to objects they are sticking on.

4.2.1.16 REPRESENTATION

Representation is a very important class for NEXUS applications because it is responsible for a graphical representation of objects. Each object can have more than one representation. For determining which representation has to be used for which occasion, a *condition* and a *resolution* can be declared in the attributes. The *shape* attribute generates the graphical representation which has to be defined in GML3D[1]. GML3D is also able to include textures so that pictures in jpg or gif-for-

1. GML3D is still not available. At the moment one has to be content with the 2D capable Geographic Markup Language [GML 1.0].

mat can be displayed as well, by simply defining a rectangle and filling it with the desired picture.

4.2.1.17 SensoryInformation

SensoryInformation describes values that emerge from data measured by sensors. The *accuracy* attribute stores the mean accuracy of the sensor, *value* contains the value last measured. *DataSource* can be either “fixed” for positions of static buildings for instance, or “sensory” for a result of a single sensor, “synthesized” for results of several sensors, or “calculated” for estimated values (see section 3.2.4: “Sensory Components”). *Actuality* is the timestamp of the last measuring whereas *sensorId* delivers the NOL of the responsible sensor.
Direction and Speed are mostly “calculated”, i.e. estimated values and needed as attributes for mobile users.

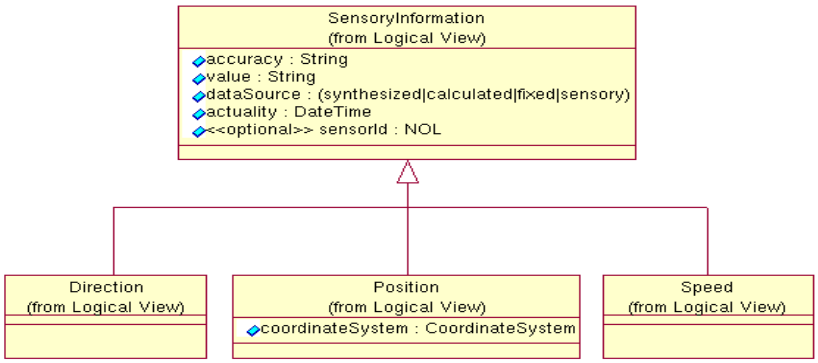


FIGURE 4-6: **Sensory Objects**

4.2.1.18 Position

The position is an essential information for local- and spatial-aware applications. Most of the NEXUS standard classes posses a *position* attribute. In most cases (namely all static objects) the *dataSource* attribute value will be “fixed” because no sensor was responsible for the position information but a user that entered the data manually. In this case the *accuracy* will be very exact and the *actuality* has to be the entry date. The optional value *sensorId* will be left blank in this case.

4.2.2 NEXUS Data Objects

DataObject is the superclass of all NEXUS objects that can be referred to by an identifier. Consequently, they can be parents of other NEXUS objects that refer to them via relations. All subclasses of this class can use the partOf relation. *Name* is an optional attribute and does not have to be unique. It is recommended to use only names when it makes sense to do so. Examples are the names of objects or buildings like a

restaurant's name or the term football.

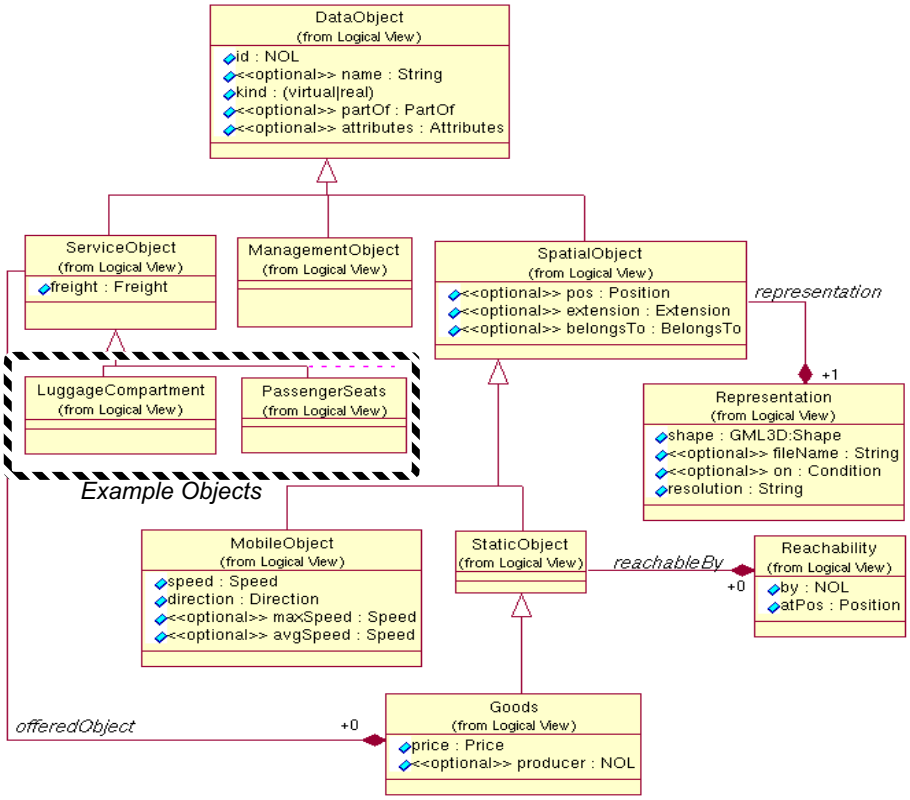


FIGURE 4-7: Data Objects Part 1

The *kind* attribute declares whether the object is real or virtual. There is no other difference between virtual and real objects because it makes no difference for NEXUS applications. Eventually, they are virtual representations of either real counterparts or virtual appearances. *Attributes* is something like an abstract attribute that can contain yet unspecified attributes. Therefore, existing classes are more flexible to small changes. So existing applications have to be only slightly modified by adding functions for the new attributes. Perhaps there are already predefined abstract functions for these generic attributes. Another solution would be the extension and inheritance of the necessary class.

At first glance, the class company does not seem to be important enough to be part of this work. But we want to create a standard that already contains the most used and the most important classes. Whenever one wants to represent ownerships in

NEXUS where natural persons do not suffice, we have to use a legal entity which will be, in the most commercial cases, a company.

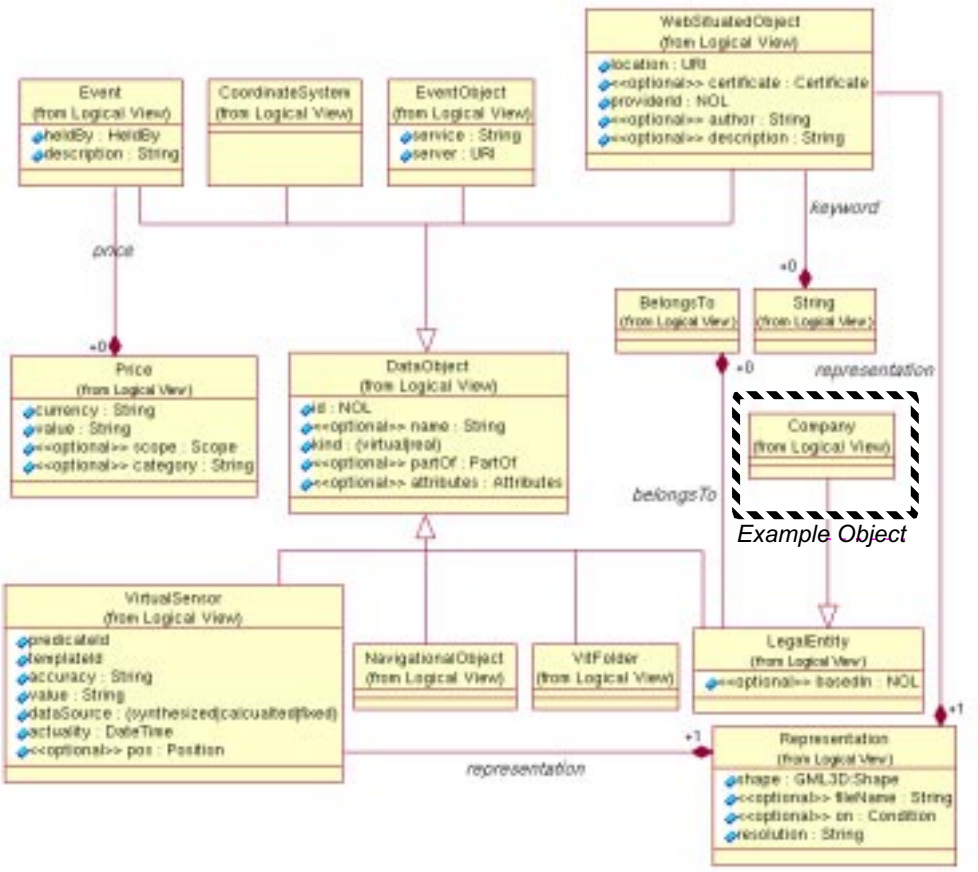


FIGURE 4-8: **Data Objects Part 2**

VitFolder is part of a VirtualInformationTower and usually contains VitPosters. For this thesis we had to change the originally suggested fixed hierarchy [VILIS2000] structure for adapting it to the more general partOf structure. This will be discussed in more detail in section 4.2.4.1: “VirtualInformationTower”.

Event does not have to be mistaken for EventObject, which describes the computer-scientist view of an event. The event we want to offer here is a commercial or organizational event, such as an opera performance or a business meeting. Events can have an organizer which can be modelled by HeldBy, they might cost some money (depending on categories in an opera performance), and they can be described.

4.2.2.1 EVENTOBJECT

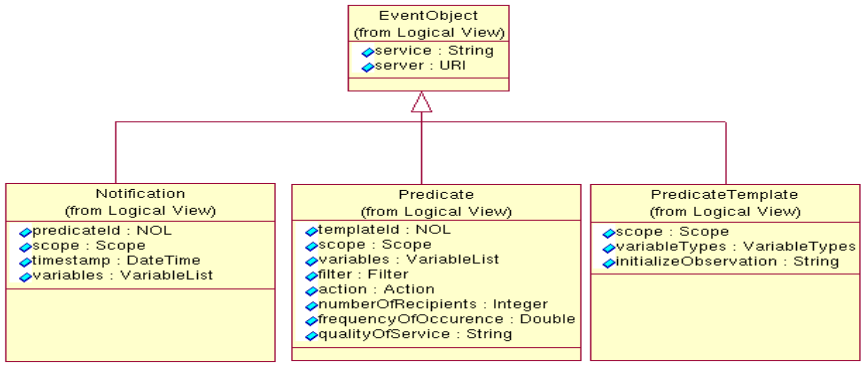


FIGURE 4-9: **Event Objects**

EventObject is the superclass of all relevant objects concerning event handling. *Service* and *server* are the attributes of EventObject. Server indicates the instance of the service that registered the event.

4.2.2.2 PREDICATE, PREDICATETEMPLATE AND NOTIFICATION

More information concerning event handling in NEXUS can be found in [BAUER 2000]. The attribute *scope* declares the lifetime of the event. *Variables* contains yet unspecified attributes but there is an example in chapter 6: “Example Scenario” dealing with these variables. *Filter* provides the possibility of filtering the persons triggering an event, by e.g. defining to act on an event only every fifth time. *Action* contains information on the actions that have to be performed as soon as the event has occurred. The attributes *numberOfRecipients*, *frequencyOfOccurence*, *qualityOf-Service* are assigned values at runtime and are used for statistical purposes.

4.2.2.3 MANAGEMENTOBJECT

ManagementObject is a container for objects that are dependent on other objects. They are used for managing timetables, price-lists and other things that could be stored in databases.

Menu stores the offer of a restaurant including prices, and therefore enables users to compare prices. BankAccount is used for money transactions involving both, natural persons and legal entities.

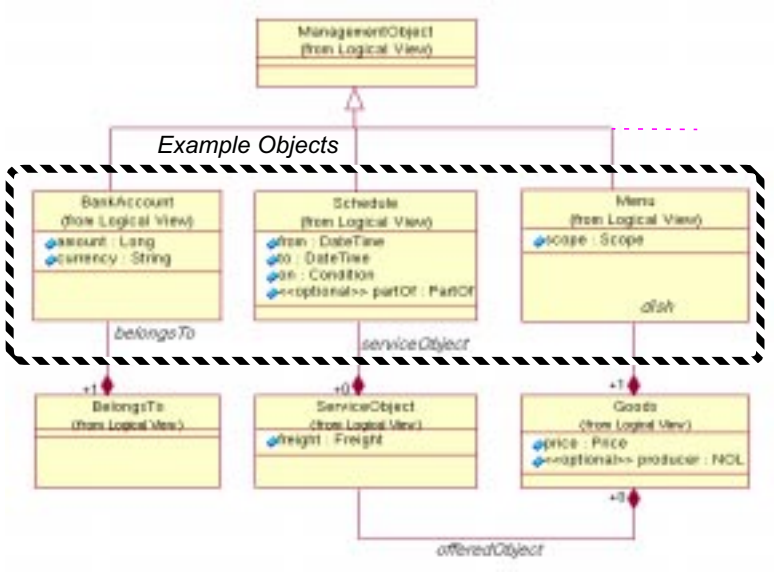


FIGURE 4-10: **Management Objects**

4.2.2.4 SCHEDULE

Schedule is an alternative to querying databases for timetables of trains, planes, buses and so on. The time of departure and arrival can be determined with this class. The *on* attribute can be used for specifying a condition like “this train goes daily” or “this train goes only on weekdays”.

4.2.2.5 NAVIGATIONALOBJECT

NavigationalObject is the superclass of all classes used for navigational purposes. In NEXUS, navigation is realized by using pathfinding algorithms like Dijkstra on a weighted directed typed graph. The type of navigation nodes or edges defines the kind of mobile object that may use the edge or node. In other words: type determines a restriction for the navigation element to a set of mobile objects. Therefore we call this attribute *restriction*. *Weight* announces the mean time that one needs to pass a navigation edge. For the duration to travel along an edge depends on the used mobile object, *weight* is an attribute of restriction. Edges have to be directed for determining the *direction* that mobile users can head for. One-way streets for instance have to be defined by a directed edge. Way is only needed if a transporta-

tion line for subways, ships, airlines, etc. is to be built. With the class Way, schedule entries can be implemented.

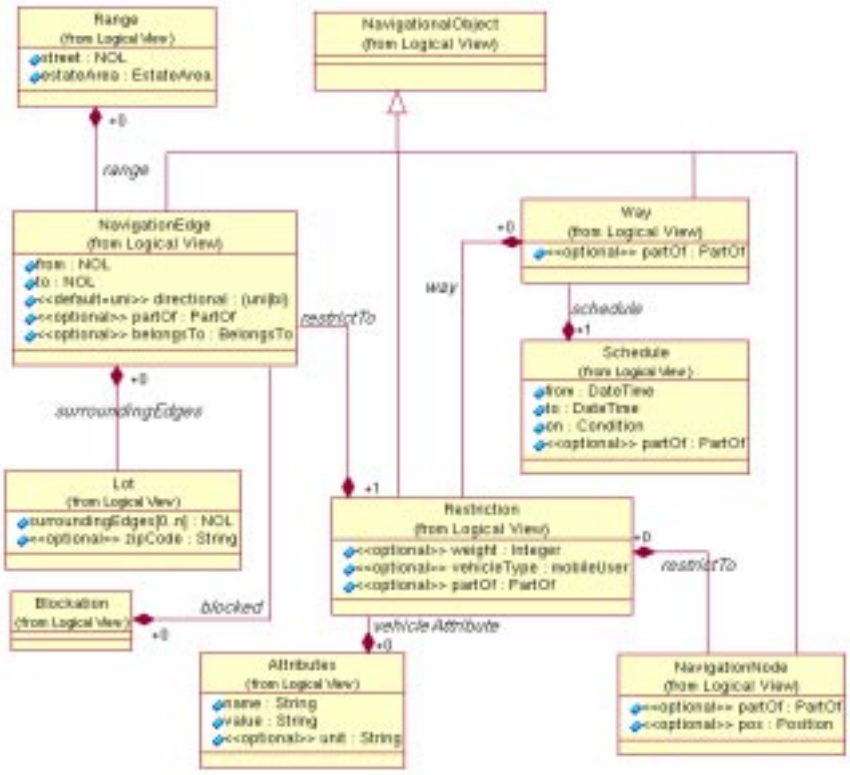


FIGURE 4-11: **Navigational Objects**

4.2.2.6 NAVIGATIONNODE

NavigationNode is the class that forms all nodes of the navigation graph. A NavigationNode has a position, is part of an area or a building and can be restricted. The restriction can be calculated by generating the union set of the restriction of the adjacent navigation edges. As a rule, navigation nodes are placed on crossings or in rooms, but they also make sense as marks for important objects in huge halls or on big squares.

4.2.2.7 NAVIGATIONEDGE

Navigation edges are the connectors between nodes that are declared by the *from* and *to* attribute. Edges can be blocked temporarily by road works, traffic jams, etc. It might also be the edge of a road where tolls have to be paid for using it. Then the *belongsTo* relation enables the modeller to set up an owner with a bank account where the toll can be paid into. *Range* is used for facilitating navigation to buildings (see section 4.2.1.9: “Range”). NavigationEdge can be divided into a set of subedges. If a motorway is modelled, it can be implemented by only one edge that

ranges from one motorway junction to the next. All gateways on this edge can be modelled as nodes connected by edges that are part of the motorway edge.

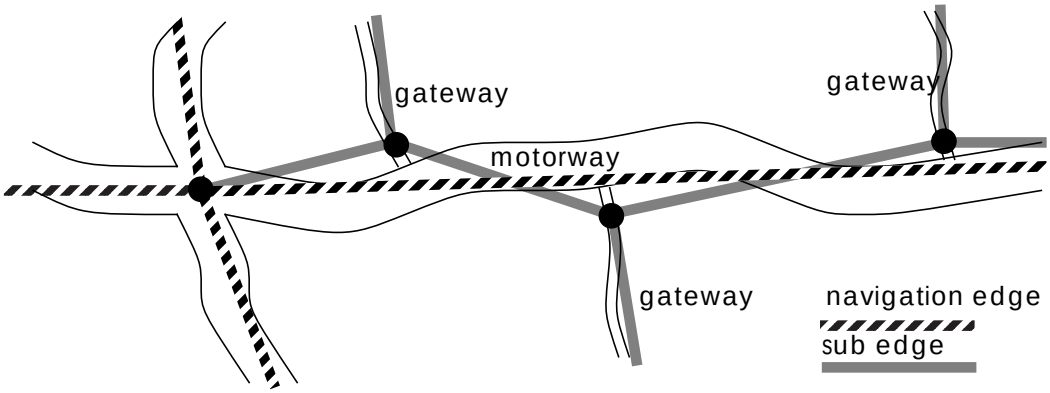


FIGURE 4-12: **Dividing a NavigationEdge in Sub-NavigationEdges**

4.2.2.8 RESTRICTION

Restriction defines which mobile objects can travel along the edge. Some edges can be stairs for example and therefore they are not usable for wheel chairs. Others might be roads that may only be used by vehicles that weigh less than 5 tons. Thus, Restriction also contains an attribute called *vehicleAttribute* where attributes of the mobile objects can be specified that continue restricting the restriction. Usually, Restrictions are part of NavigationEdge and NavigationNode.

4.2.2.9 SCHEDULE

Schedule is used as a replacement for external databases where timetables of public transports are stored. Each Schedule object defines exactly one departure from the node specified in NavigationEdge's *from* and one arrival in NavigationEdge's *to* attribute. Whenever Schedule is used on a navigation edge it has to be unidirectional. The *on* attribute is an optional condition where one can specify that the transport only plies at special times (like on weekdays or on holidays). ServiceObject declares the objects that might be transported with the vehicle.

4.2.2.10 WEBSITUATEDOBJECT

WebSituatedObject is the superclass of all classes that are situated on servers and can be addressed by a URI (Uniform Resource Identifier). Webpages and databases can be accessed by applications and are therefore modelled in the Augmented World Model.

Applications are important in the following case: a person using a NEXUS application enters an area where NEXUS is not available anymore. By having the network cells and the coverage modelled in the Augmented World Model it is predictable

when the necessary data cannot be sent anymore. Then the required data and the appropriate application can be downloaded before leaving the covered area, and the application can be used off-line as long as the user does not perform any unpredictable actions. The process of fetching information that can be used later is called information hoarding.

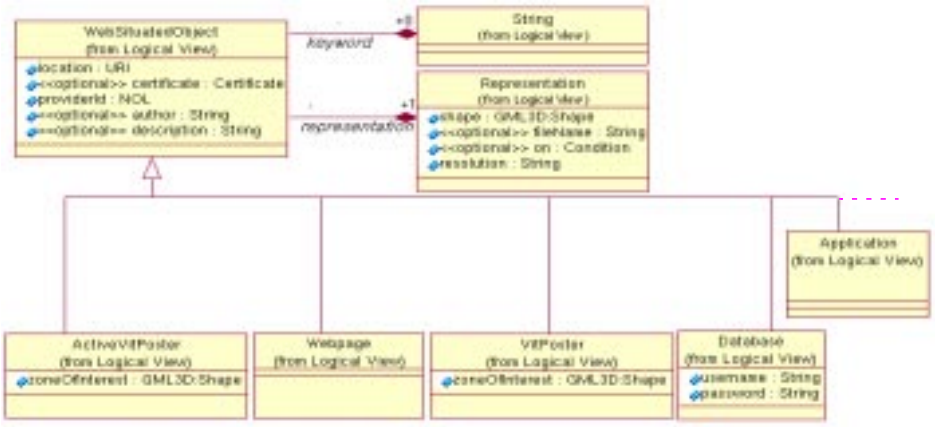


FIGURE 4-13: **Internet Objects**

ActiveVitPoster and VitPoster are part of the Vilis-project. They are objects that present web pages on a virtual information tower (see [VILIS2000]). The *zoneOfInterest* attribute is a sub-area of the *zoneOfInterest* defined for virtual information towers (see section 4.2.4.1: “VirtualInformationTower”).

4.2.2.11 SERVICEOBJECT

ServiceObject specifies the kind of services that are offered by companies like airlines, railways, malls, hotels etc. The attribute *offeredObject* references the object itself that can be for rent or for sale, respectively. LuggageCompartment and PassengerSeats are examples for an airline but it would also be sufficient to use ServiceObject instead.

4.2.2.12 VIRTUALSENSOR

VirtualSensor can be used for combining synthesized values of distributed sensors. The result of the synthesized values are stored in the SensoryInformation class that refers to the virtual sensor. The distributed sensors are all connected to the virtual sensor via their partOf relations. VirtualSensor is also needed for observing functional dependencies and for observing events if one of the involved dependent classes changes its values.

4.2.2.13 SPATIALOBJECT

SpatialObject is an object that is situated in a space. Thus it has a position, an *extension* and a *representation*. Depending on the resolution and on the invoking application the representation can differ.

4.2.3 MOBILEOBJECT

Mobile objects are able to move. They were either constructed to move or they are able to move on their own. The attributes *speed*, *direction*, *maximum speed* and *average speed* are for internal use and can help estimating future positions with a certain probability. Post-Its that are sticking on mobile objects are not mobile. They have to be modelled by using the SticksOn relation. The position of the transported objects can be calculated consequently.

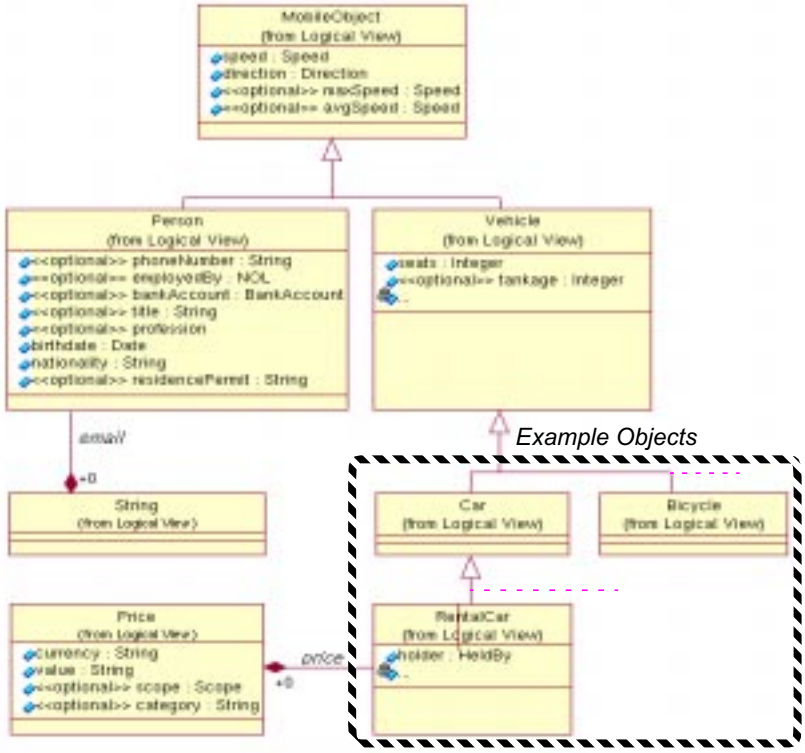


FIGURE 4-14: **Mobile Objects**

4.2.3.1 PERSON

A person is a mobile object, too. Since this class is in most cases a user of NEXUS, it will need dozens of attributes depending on the applications that are using person

as a class. So we propose to extend person and to customize it to the individual needs of each application.

4.2.3.2 VEHICLE

Vehicle contains all mobile objects that are built by humans. This includes vehicles with motors and without, such as bicycles, wheel chairs, skate boards, tea-carts, trolleys, mobile beds in hospitals, cars, vans, etc.

4.2.4 STATICOBJECT

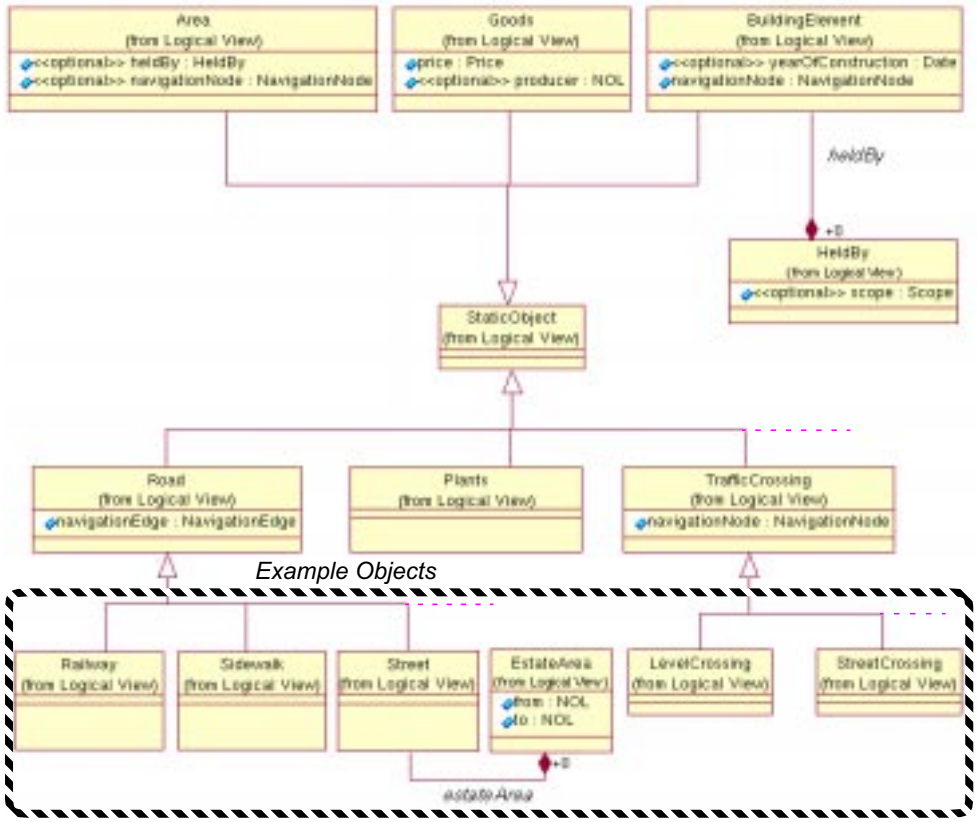


FIGURE 4-15: **Static Objects Part 1**

In contrast to mobile objects static objects have a fixed position. Whenever they are moved (tables or other furniture for example) the Augmented Area that contains the moved elements has to be updated by an editor[1].
InfoStations can be used for hoarding. An information station offers a broad band-width for exchanging data. PostIt is a static object that can stick on another mobile

1. An editor is someone who is allowed to change (update) objects.

or static object and contains a note. Plants is used to be the superclass of all plants that can be inserted below that class if needed.

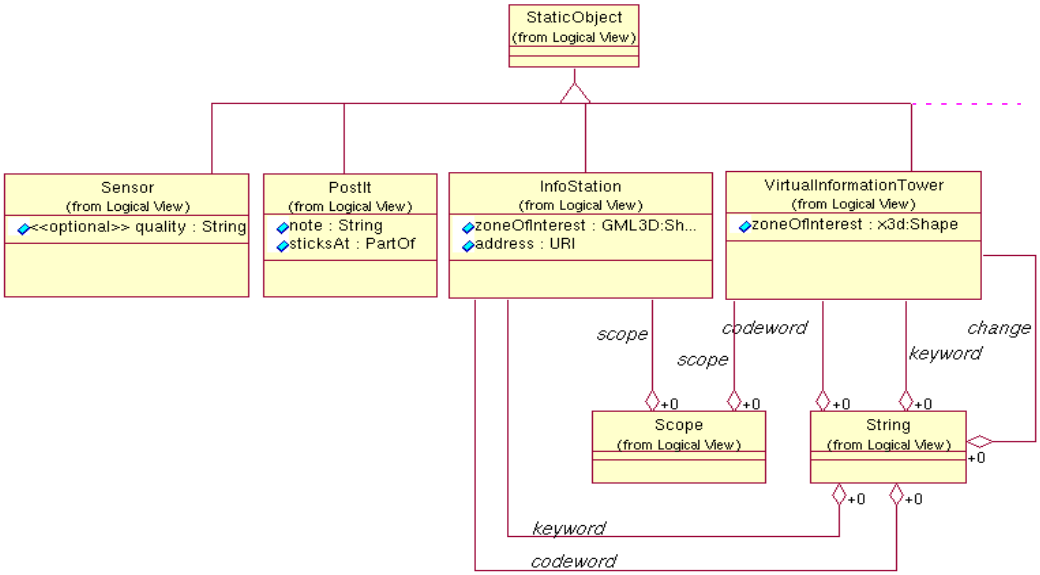


FIGURE 4-16: **Static Objects Part 2**

4.2.4.1 VIRTUALINFORMATIONTOWER

A virtual information tower contains posters with information on it (see [VILIS2000]). Posters can be active and are displayed to passing users interested in the information tower's content without demand. There can be only one ActiveVitPoster per virtual information tower. The amount of VitFolders and VitPosters is unlimited. Regular VitPosters have to be requested by navigating through a VitFolder. We changed the [VILIS2000] definitions so that the former cylindrical visibility property is transformed into a *zoneOfInterest* of type GML3D which is conformable to other three-dimensional bodies defined in this thesis. The owner property is described by the *belongsTo* relation. The hierarchy of contained posters and folders can be implemented by using the *partOf* relation, which is more conformable to the NEXUS object hierarchy system than the solution suggested in [VILIS2000]. ExpiryDate is changed into *scope* that provides a start and an end date.

4.2.4.2 SENSOR

Sensor is an object that represents real-world sensors like thermometer, light barriers, photo sensors and so on.

4.2.4.3 TRAFFICCROSSING

TrafficCrossing is the superclass of all crossings. These can be railways, streets and other important traffic junctions. For navigation applications, traffic junctions are represented by navigation nodes. For presentation purposes TrafficCrossing implements a representation.

4.2.4.4 ROAD

Similar to TrafficCrossing, Road implements the presentation data of navigation edges.

4.2.5 GOODS

Goods are used for modelling the transaction of goods. Transactions can be the sale or leasing of objects. The distinction of Figure 4-17 into Food, HardwareGoods and

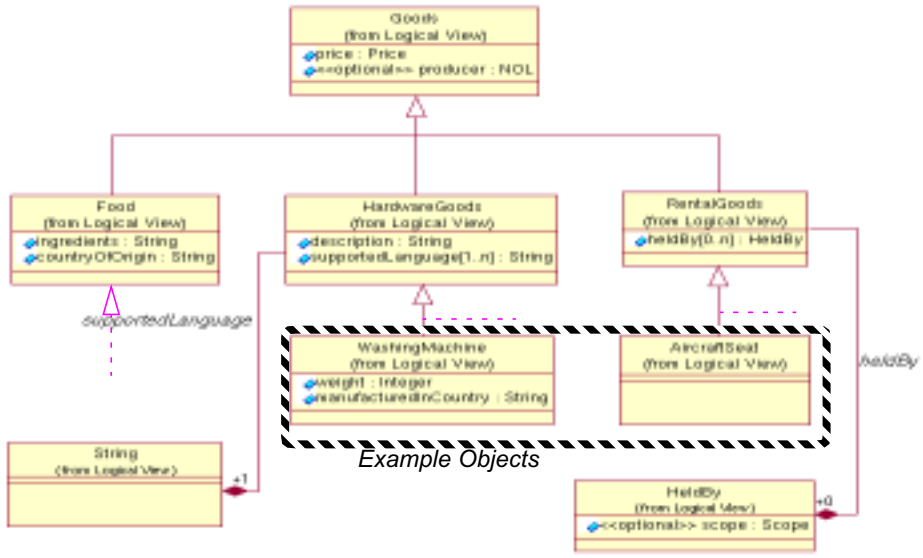


FIGURE 4-17: **Commercial Objects**

RentalGoods seems to be impure at first glance. So RentalGood contains possibly objects out of HardwareGoods. But usually it makes no sense to rent food, so that RentalGoods does contain only hardware goods. Consequently, we thought HardwareGoods to be goods that can be sold, and RentalGoods to be hardware goods to be rented. Admittedly, the design can be slightly improved by inserting RentalGoods as a subclass below HardwareGoods.

Food is the superclass of all goods that can be eaten. It is used for modelling menus of restaurants. HardwareGoods is the superclass of all goods that can be sold. Rent-

alGoods is used for objects that are for rent. HeldBy specifies the user who rents the object. RentalGoods can also be seats in planes or opera buildings.

4.2.6 AREA

Area includes all classes implementing an area, such as country, city, estates, etc. Universe is the uppermost class for all partOf relations. Theoretically, there can be more than only one Universe, but then there would be more than one Augmented World Model, whereby multiple Augmented World Models are not connected to each other by links.

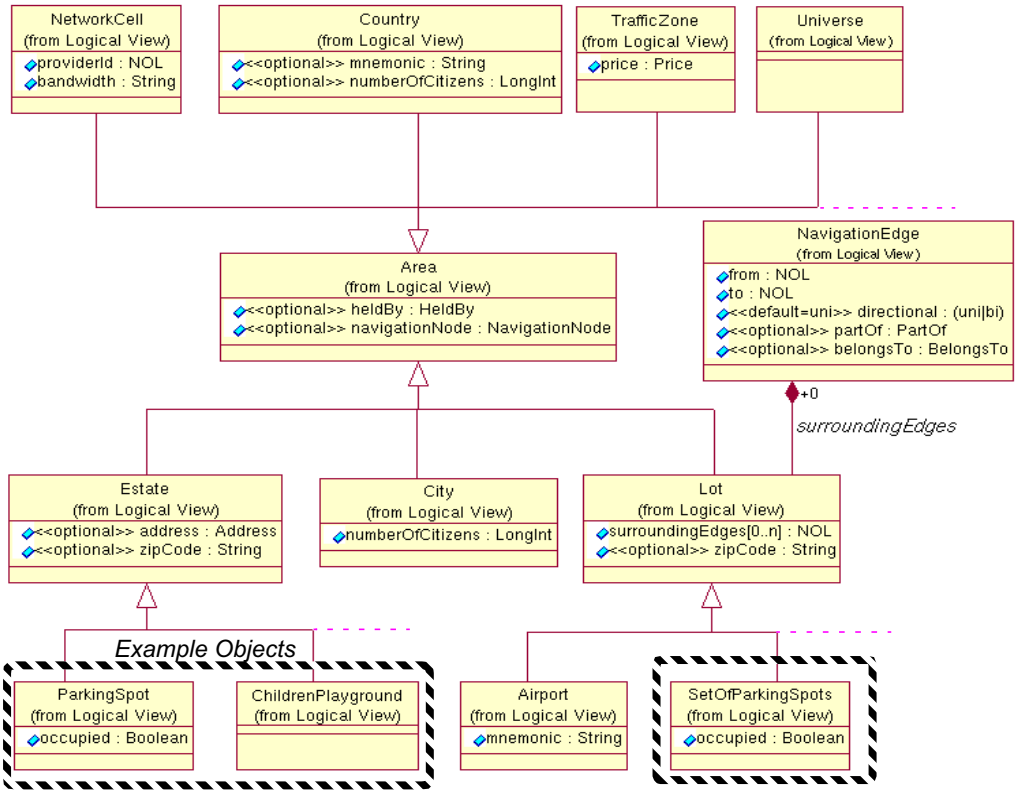


FIGURE 4-18: **Area Types**

4.2.6.1 CITY AND COUNTRY

City implements the uppermost class for all elements of a city like buildings, streets, lots, estates and navigation elements. Country is the top-level class for all objects belonging to a state. Assuming that City or Country form an Augmented Area, the area size of the modelled city or country might be too big. Therefore, Country could also be a sub-area of a state (state in the meaning of an independent form, e.g. Germany, Switzerland, etc.) and the object City a quarter of a city. This can be used for distributing large-scaled areas in smaller and more feasible regions.

4.2.6.2 ESTATE

Estate is the plot of land where one building or a set of functionally dependent buildings (e.g. two buildings of a hospital) is placed.

4.2.6.3 LOT

Lot is the next upper instance of estate in the object hierarchy. A lot is an area that is surrounded by streets. In planned cities lots are usually rectangular or quadratic[1]. Lot is mainly an organizational unit for dividing a city in smaller sub-areas.

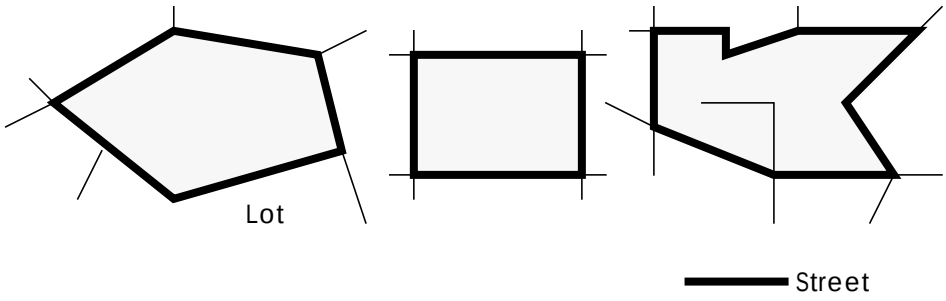


FIGURE 4-19: **Examples for Lots**

4.2.6.4 AIRPORT

Airport can be imagined as a lot, too. It is the area where buildings, runways, terminals etc. of a typical airport are built on.

4.2.6.5 NETWORKCELL

NetworkCell implements the net coverage and the bandwidth for areas within NEXUS. With this class it is possible to estimate where the bandwidth for certain applications will not be sufficient so that information hoarding has to be done.

4.2.6.6 TRAFFICZONE

This class implements the different zones of public transportation systems whereby calculations for the prices have to be performed by evaluating the extensions of the zone and comparing it with the positions of the start station and the end station.

1. In the US they are usually called block. Here, we avoided this term because we do also inherit other objects from lot such as parking lot, building lot, etc. Therefore a somewhat more general term than block seemed to be advisable.

4.2.7 BUILDINGELEMENT

BuildingElement is something like a construction kit for elements that can all be parts of buildings. Here one finds the classes Room, Flat and Building. Flats, buildings and rooms can be rented by users and do therefore have a HeldBy attribute.

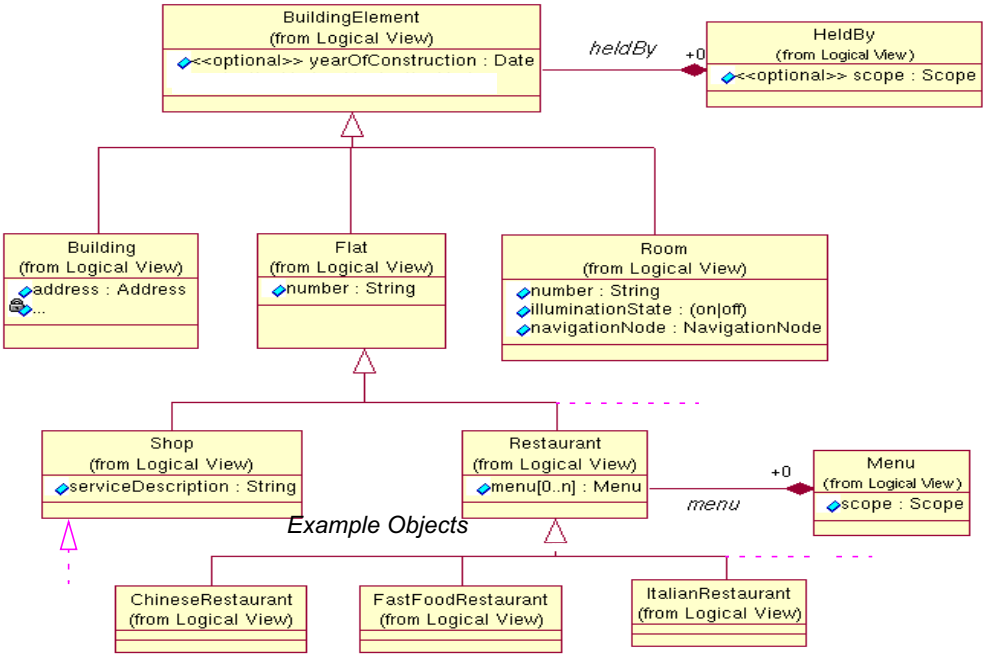


FIGURE 4-20: **Building Types**

4.2.7.1 SHOP AND RESTAURANT

Shop and Restaurant are subclasses of Flat. Shops can sell HardwareGoods and Restaurant sells Food that is described by the *menu* attribute. For further specifications of the restaurant type, Restaurant should be inherited such as shown in the example illustrating FastFoodRestaurant, ChineseRestaurant and ItalianRestaurant.

4.2.8 ROOM

Room is the superclass of all rooms and subsets of rooms. Rooms can be illuminated or not. The real illumination state can be checked by sensors whereby applications are able to change the *illuminationState* modelled in Room.

Office can be a box within a big room or a simple desktop and consequently is a part of a room whereas OfficeRoom implements a whole room with several work places in it.

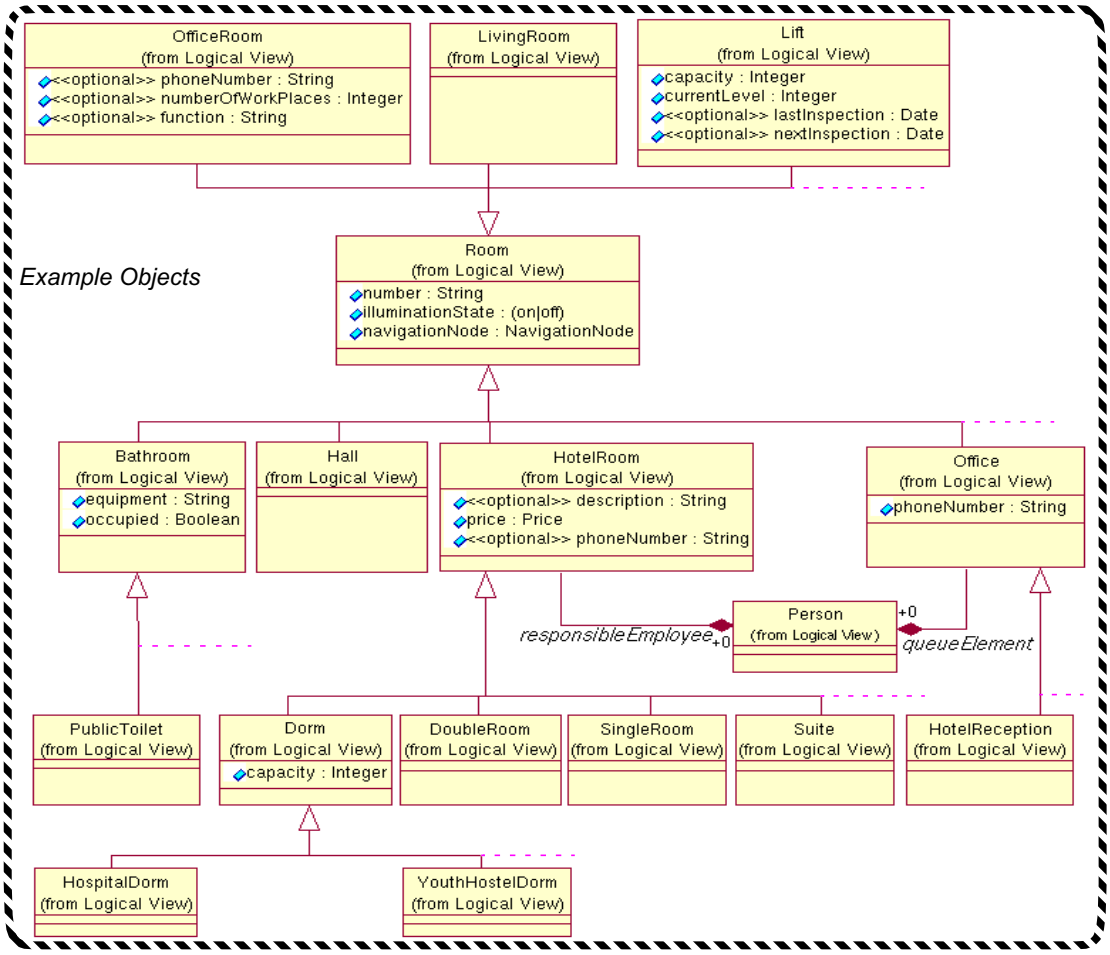


FIGURE 4-21: **Room Objects**

4.2.9 BUILDING

Building implements all kind of buildings. Hotels, Malls, Hospitals, EventBuildings and StationBuildings are subclasses. The attribute ward of hospital defines the wards that are part of the hospital. Alternatively, this could also be realized by mod-

elling a ward class inheriting from legal entity that is connected by a partOf relation to hospital.

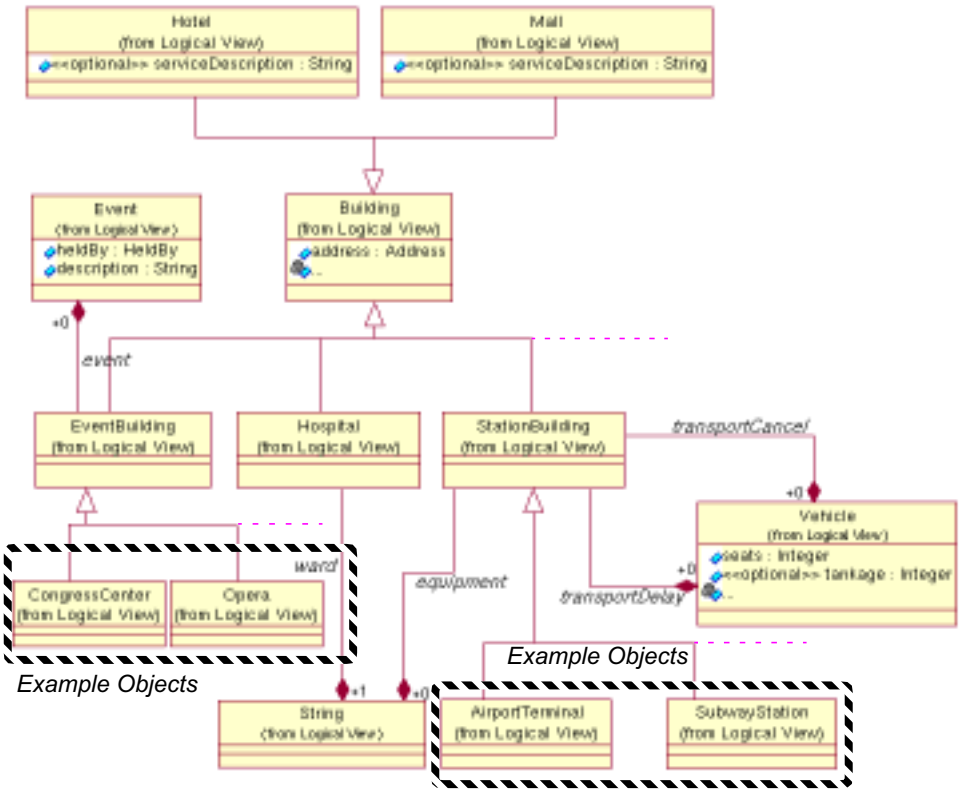


FIGURE 4-22: **Building Objects**

4.2.9.1 MALL

Malls are conglomerations of shops in one building. The attribute *serviceDescription* offers the possibility of delivering a short description of services offered by the mall's shops.

4.2.9.2 EVENTBUILDING

EventBuilding is something like an opera, a theatre etc. There, events can take place.

4.2.9.3 STATIONBUILDING

StationBuilding is the superclass of all buildings containing navigation nodes and edges that use schedules, i.e. airport terminals, underground stations, bus stations, train stations etc. For providing information on delays or cancelled transports the

appropriate attributes *transportCancel* and *transportDelay* are defined. They refer to the vehicle of a line (e.g. a flight) that is delayed or cancelled.

4.3 SUMMARY

In this chapter we have tried to define a superset of important classes for implementing an Augmented World Model. We have created the necessary data classes to implement the itemized applications with regard to the requirements as stated in chapter 3: “Requirements”.

Of course, the feasibility of the class schema has never been tested and thus cannot be guaranteed to work as envisioned.
On the one hand there are implementation aspects like data consistency problems due to the distribution of the data on several Augmented Areas. Another problem might occur with the feasibility of the complex structure of partOf relations. Search algorithms that have to perform searches within the partOf network (it is not necessarily a tree) will take considerable time to find the demanded object.
On the other hand, the functionality of the designed classes is still unproven. It is not guaranteed that all necessary attributes and classes are existent. Moreover, the amount of classes is considerable and not all of them might really come into use later on. So there might be classes in the standard schema that are nearly useless and therefore should not be part of the Augmented World Standard Schema Classes.

Since we want to assure ourselves as far as possible that all the modelled data is useful and correctly designed we have to check it in an example scenario, which is done in chapter 6: “Example Scenario”.
As we already know, classes have been defined to ensure the operability of the requirements but the technical implementation has only been of secondary importance until now. Thus, these aspects, namely the querying of objects in our complex partOf structure and the consistency within functional dependencies, have to be examined further in chapter 5: “Implementation Aspects of the Augmented World Model”.

CHAPTER 5

IMPLEMENTATION ASPECTS OF THE AUGMENTED WORLD MODEL

As long as there were no machines, programming was no problem at all; when we had a few weak computers, programming became a mild problem, and now we have gigantic computers, programming has become an equally gigantic problem. In this sense the electronic industry has not solved a single problem, it has only created them - it has created the problem of using its products.

(Edsger W. Dijkstra, The Humble Programmer)

This chapter deals with implementation aspects. Here we discuss the distribution in NEXUS concerning the creation of object identifiers, cooperation between Augmented Areas, and consistency of data. Furthermore, we examine possibilities of exchanging data, which results in the suggestion of using an Augmented World Modeling Language. We also consider the complexity and difficulties of navigation networks within NEXUS. Last, we deal with an example that shows the functionality of navigation networks by using system components and the Augmented World Model.

5.1 DISTRIBUTIONAL ASPECTS

In NEXUS, distribution is a very important aspect. As we already know, distribution is necessary for fulfilling the criteria to provide an open global platform for spatial-aware applications (see section 1.2: ”Motivation and Objectives”). In the following sub-sections we want to examine some issues that can be attributed to the distributional character. First we deal with the creation of unique object identifiers. Then we want to take a look at consistency issues. There are two facets we have to consider. The distribution separates Augmented Areas from each other, so that consistency among Augmented Area Models cannot be guaranteed. Therefore we want to analyse the possible behaviour of models to one another. Last, functional dependencies going through various Augmented Area Models are going to be studied.

5.1.1 GENERATING UNIQUE OBJECT IDENTIFIERS

Usually, we need a global unique identifier (or universal unique identifier) for differentiating between data objects or elements. In object oriented programming we need identifiers for objects in order to refer to them and for setting up inheritance. In location- and spatial-aware applications both concepts are combined: identifiers refer to objects and they serve as primary keys in the database. The problem we usually have when creating unique identifiers in a distributed system, namely not to assign two identical identifier at the same time, will not be treated in this thesis. [UUID 1998] suggests a solution for this problem. We want to consider a possibility of integrating more information into the identifier, so that we are able to determine the area the object is situated in when examining the identifier.

The creation of a standard unique identifier is described in [UUID 1998]. There, a 128 bit long key is generated. The first part consists of a time stamp and a clock sequence, that is activated whenever the system clock might have been changed (e.g. while the system was powered off). The last bits of the identifier are formed by a node identifier, which is either derived by the MAC (Media Access Control) address or by a randomly generated number.

Implementing identifiers, there are three basic approaches:

- *Identifiers are unique to their real world object’s representations:* Using this approach, identifiers would not be unique within the system. Different data objects representing the same real world object have the same identifier. However, this would be an inconsistence the data management system could not handle. So an additional identifier suffix is needed. The base identifier is extended by additional two bytes for differentiating between diverse representations of the same real world object. Two bytes suffice for 65535 different representations of one real world object.This approach has advantages and disadvantages. Advantageous is the fact that one can determine two identical objects from the beginning and that consistencies are more likely being kept. Imagine the case that there is an Augmented Area Model describing a town. Then, a competitive model is set up

which describes the same town, the same buildings, streets and so on. Both models are offering different services: the first is responsible for navigation, the second is used for evaluating pipe bursts of the urban sewer system. Links between both Augmented Area Models would be useful if an assigned building firm wants to navigate from its office to the place where the pipe burst took place. But if identical identifiers are not mandatory and the competitive models have not been linked manually by using references, the system will simply not know that both models are representing the same town.The disadvantages are obvious, nevertheless. Setting up new objects representing the same real world objects is very expensive. When one wants to create a new representation, one has to query the number of existing ones, first. Then, a system-wide lock for the real world object is necessary that allows only one new representation to be created (for avoiding a parallel creation of identifiers, that would be identical). Afterwards, the object has to be unlocked.

- *Identifiers are unique to objects representing a real world object:* If we use this approach, any new instantiated representation of an object is getting his very own identifier that is unique within the whole system. In this case nothing more than the 128 bits for the identifier are needed. The identifier is easy to create. This task can be done locally without contacting any other servers for identical objects. The only disadvantage is that one has to care for the links to other models and objects on his own for each object.
- *Combinations of identifiers are unique:* In the last case, a combination of identifiers is unique for the whole system. The idea behind this approach is as follows: consider an Augmented Area as an object, thus it has an identifier. Each Augmented Area Model consists of several objects. Therefore an object identifier has to be unique within a model and a model identifier has to be unique globally. The conclusion is the consideration of using an identifier pair (area id, object id) for referencing objects globally [GROßMANN 2000].

The last proposal leads to the idea of using an internet-protocol-like design: the identifiers can be replaced by names that are unique in a machine-wide or model-wide context, respectively, and can be accessed by the following way: nexus://<machine name>/<model name>/<object name>. Be aware of the fact that the <machine name> is necessary because the system-wide uniqueness of <model name> is not mandatory in contrast to the model identifier. Of course it would also be possible to access an object by only providing the following information: nexus://<machine name>/<model id>/<object id>, where <machine name> is not a necessary information. Leaving out the object information, i.e. nexus://<machine name>/<model name>, results in referencing a certain model and is called Augmented Area Model Locator. Providing the full information including <object name> (as in example above) gives us the NEXUS Object Locator (NOL).

5.1.2 AUGMENTED AREAS: COOPERATING VERSUS COMPETITIVE

In the Augmented World of NEXUS, there are a lot of data objects representing real world objects and virtual objects. An Augmented World object is always a data object with attributes that can be other objects (compositions or aggregations of objects) or simple values. Logically, objects belong to Augmented Area Models or, in other words, to an Augmented World Model. Models are collections of (normally adjacent) static objects, either virtual or real (e.g. buildings) where consistencies between objects within a model are guaranteed. Models can also be collections of logical related objects, like mobile objects of type train or car. Considering hierarchical structures of objects (i.e. the partOf relation), hierarchical structures of models are conceivable, too. But then Augmented Area Models are treated as objects belonging to a model managing Augmented Area Models. This is important for avoiding problems with non-ambiguous pairs of model and object identifiers, as described in section 5.1.1: ”Generating Unique Object Identifiers”

An Augmented Area Model is provided by an institution that might be educational (mainly in the beginning), governmental, commercial, or personal. In an optimistic view models know each other. The Augmented Area Model of Stuttgart West, for instance, knows about the existence of a model called Stuttgart Ost. Thus, objects within these models representing the same real world object can be correlated easily by using the same identifier for example. Correlating objects of different models is important for realizing queries over distributed Augmented Areas. One example is a query for all Italian restaurants in Stuttgart West whose owners are living in Stuttgart Ost. Augmented Area Models that are correlated are called cooperative models. We can further distinguish cooperative models in directly and indirectly cooperating models, respectively. Directly cooperating Augmented Area Models know each other from the beginning and are using the same identifier for data objects representing the same real world object. Indirectly cooperating models did not know each other by origin and did not use the same identifier as a consequence. Therefore links can be employed to establish a correlation between two objects. Then one cannot quickly identify two data objects representing an identical real world counterpart, but one has to read the complete object information including the links. In a less optimistic view, Augmented Area Models do not know each other, they are not cooperative but competitive. This could happen for marketing strategic reasons or for a much simpler one: the provider really does not know about the existence of the other model. If NEXUS became an application world-wide used, it would likely be possible that providers offering a service for the same real world object do not know each other.

The conclusion of this discussion is the fact, that one has to be aware of the existence of competitive and indirectly cooperative Augmented Area Models. By using directly cooperating models only, one could create an identical identifier for any object of a new model representing the same physical existent object. Assuming that there are also indirectly cooperating models, the premise that a number of NEXUS objects representing the same real world object do have the same identifier, is not true for any case. Therefore we have to check for manually set links parallel to the execution of each query. Of course, there is no guaranty that all the links are up-to-

date. Providers not knowing each other surely will not have set any links. Moreover, there will be error messages like the http 404: “document not found” (object not found, in our case). To prevent errors like that we need a mechanism providing consistency.

5.1.3 CONSISTENCIES WITHIN FUNCTIONAL DEPENDENCIES

First, we want to give an example for a typical functional dependency before continuing with our examination:

An office building is represented by its floor plan in an Augmented Area. It is also part of the Augmented Area managing the data of the building´s surroundings. Now it is planned to demolish the office building. Here we have a functional dependency between the building´s representation on the area map of the surroundings and the building´s ground plan. If one ceases to exist the other one should also disappear.

Let us recall the statement that changes in the physical existent world have to be propagated to the virtual world, and the other way round, and that changes in the virtual world may trigger updates in the physical world. In Figure 5-1 the process of the projection and the update after a transformation of the world’s state is shown. What we want to consider, concerning consistency and updates, are duplicated data objects representing the same real world object. A change in the state of one virtual object that has to be propagated to the physical world must also be propagated to potential other virtual counterparts of the physical objects. Otherwise a query about the represented object would result in two different outputs when asking both virtual objects (presuming that the projection from the physical world to the virtual world cannot be perfectly and completely performed by sensors).

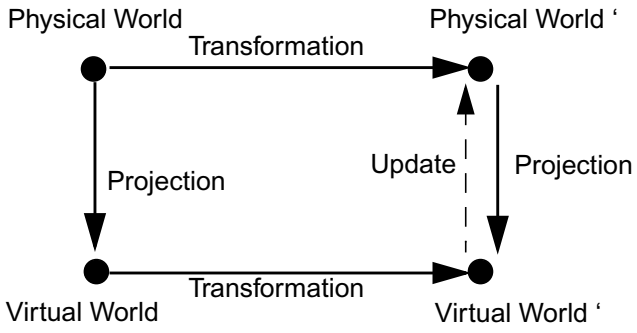


FIGURE 5-1: **Dependencies of Virtual World and Real World**

Thus, we see that we have to offer a functionality for avoiding inconsistencies at least for functionally dependent objects. So where are functional dependencies likely to happen in NEXUS:

- *partOf Relations:* As seen in the example above, rooms which were part of the building must vanish at the same time the building disappears.

- *sticksOn Relations:* Assuming a post-it that stuck on an office door of the mentioned building it has to be removed from the Augmented Area Model as well.
- *IdenticalTo Relations:* Imagine a physically existent room with a light turned on and a mobile user in it whereby two Augmented Area Models represent the room. The user accesses one model and commands via a NEXUS application to turn off the light. However, the other model has not been connected and does not know that the light has to be turned off.
- *navigationEdge Relations:* A navigation edge that connects two nodes which are situated in a building and at the outside must be removed when one of the nodes vanishes.
- *BelongsTo Relations, etc.*

⇒ all relations are able to cause functional dependencies when spread over several Augmented Area Models

Now that we know where functional dependencies might occur, we want to find the possibilities for avoiding inconsistency. The most appropriate way to master this problem is the use of the Event Service [BAUER 2000]. It is conceived to observe changes in the environment and to trigger actions.

1. *Register an event for each object that might be functionally dependent on others:* Initially, we have to define an event that is able to recognize changes in the state of NEXUS objects. Therefore we have to define a predicate for the Event Service. The predicate name is maintainConsistency. Now there are two cases we want to study (see Table 5-1). When an object is registering firstly for maintainConsistency, its own NEXUS Object Locator and the one of the object that is to be observed is stored in variables.selector[1]. In case of a change in state action.recipients[2] delivers variables.selector as a result and consequently sends a notification

Case	Input Parameters	Output Parameter
first subscriber	NEXUS Object Locator of subscriber and of object to be observed	Event Identifier
any further subscriber	NEXUS Object Locator, remaining predicate data of existing event	Event Identifier

TABLE 5-1: **Predicates for Maintaining Consistency**

to the elements of variables.selector. Any further object registering for this predicate will be added to varaiables.selector.

2. *Register only one event and find potentially affected objects via the Area Service Register:* This approach is more complicated to implement because it is difficult to determine all the kinds of functional dependencies before they are going to eventuate. We have to define the area or the set of objects, respectively, that might be

1. variables.selector defines the set of objects that are able to trigger an event
2. action.recipients defines the set of objects that are going to receive a notification

affected by a functional dependency. It is theoretically possible to demarcate potentially affected objects of another object's modification via its position and spatial functions like "lies within", "touches", "is identical", etc. So we have to define (besides variables.area) variables.function that specifies the appropriate geometric method to evaluate the predicate. Figure 5-2 illustrates the described

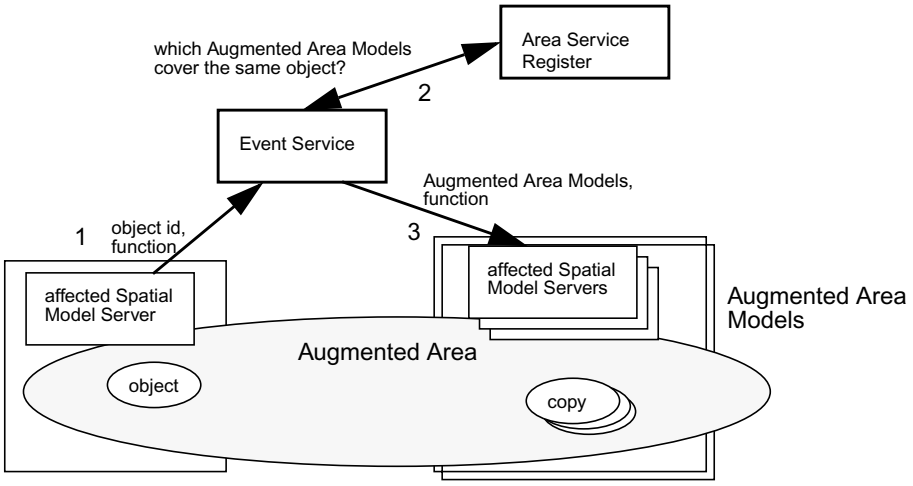


FIGURE 5-2: Declaration of Affected Objects for Determining Functional Dependencies via the Area Service Register

process. First, the Event Service remarks a change in the objects state. It receives the object identifier and a function defining other potentially affected objects (1). Then, the Event Service has to look up other Augmented Area Models situated on Spatial Model Servers that contain the same area like the object (2). The resulting models can be used for filtering affected objects by employing the defined function ("lies within", "touches", etc.) (3).

After considering these approaches we have to deal with the notification. There are several possibilities to realize it, too.

1. *No action of anybody:* The notification contains nothing. It is up to the applications to recognize the necessary actions that have to be performed.
2. *update / delete + Id:* The notification contains information about the action that has to be performed. Actions can be an update or a delete whereby cascaded actions should be possible as well.
3. *object data:* The notification holds the updated or changed object data or an empty set, respectively in case of a vanished object.

The presented event has to be provided by the system components. Desirable would be an additional event that informs the provider A of an Augmented Area Model that another provider B has related one of his objects to one of B's. So they are both aware of the danger of potential inconsistencies and are able to be responsive to it.

The suggested method should only be used in absolutely necessary cases because the equipment of too many objects with observing events could result in a loss of system performance. It is mainly up to the provider to avoid inconsistencies between models that might occur due to overlapping data. This can be traced back to the fact that one Augmented Area has so many duplicated objects that it is infeasible to update every change.

5.2 DATA EXCHANGE FORMATS

5.2.1 INTENTIONS

The Augmented World Modeling Language is responsible for describing the NEXUS objects (including attributes) stored in a distributed database. The distribution is due to the partition to mobile and static objects, hence the division of the data to the Location Service and the Spatial Model Service. Moreover the data is further distributed corresponding to Augmented Areas covered by Location Servers and Spatial Model Servers.

Furthermore we want to design NEXUS as a global and open platform where anybody is able to add and change[1] information similar to the World Wide Web (WWW). Thus the Augmented World Modeling Language is comparable to the Hypertext Markup Language (HTML) with one exception: the data elements look different because in NEXUS a combination of virtual objects (VRML and HTML) and physical existent objects have to be modelled.

5.2.2 THE AUGMENTED WORLD QUERY LANGUAGE

First of all, it should be mentioned that apart from the Augmented World Modeling Language there exists an Augmented World Query Language which can be used for formulating queries. Responses to these queries are delivered in the Augmented World Modeling Language. In the future, the Augmented World Query Language will be a SQL99 derived language. At the moment, however, it is still defined as an

1. Naturally, one has to regard the user permissions for changing, adding or deleting data. Only authorized entities may be able to manage and create certain objects and structures. The openness mainly refers to personal items and properties in the area of physical existent objects.

XML derived construct, which does not allow nested queries and which is restricted to boolean expressions and simply held partOf relations of polygons.

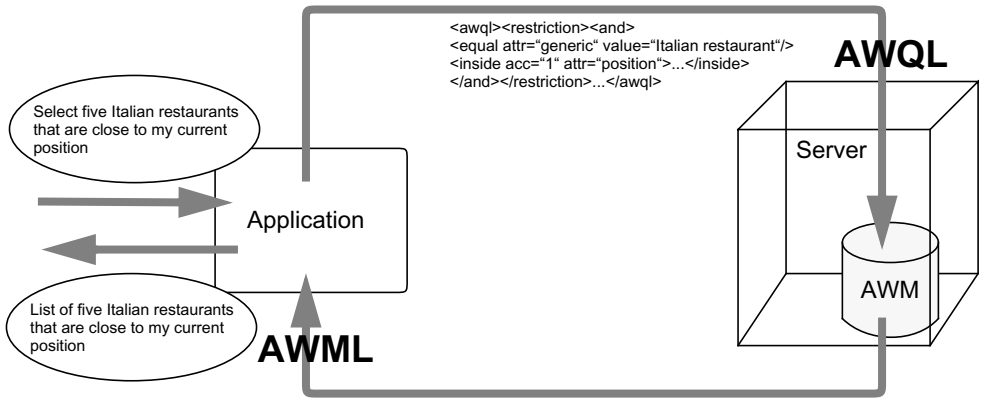


FIGURE 5-3: **Communication Process by Using AWML and AWQL**

5.2.3 SELECTION CRITERIA

Since this section deals with the Augmented World Modeling Language, we want to concentrate on this topic from now on. For selecting a suitable language we have to establish some selection criteria which are to be presented in the following.

1. *little need of memory:* The consumption of memory is dependent on the kind of query. Assuming queries to happen via radio networks, a small memory capacity per object is desirable for enabling fast communication processes.
2. *efficient processing:* The syntactic and semantic analysis of information kept in the modelling language should be efficient. If the information is available in a packed or encrypted[1] state, additional effort for processing the decrypting has to be added.
3. *uncomplicated integration of the Augmented World Model into the system:* The effort for integrating new Augmented Area Models should be small. The implementation for providing efficient processing should be easy to do.
4. *portability of data objects:* Data (especially already existing 3D or 2.5D data) should easily be portaged to the Augmented World Model. Furthermore the portability to other systems must be warranted. Therefore a common and standardized description language should be used.
5. *maintainability of the model:* Changes concerning objects and the structure of objects, respectively, should be executable with little effort during the system's runtime.

1. Due to security aspects an encryption method can be put on the transmission of position information.

6. *extendibility of the model:* Additional objects and structures should be insertable at any time. The Augmented World Model must be extendible.
7. *efficient access:* When processing queries the time needed to access the data must be efficient. This is dependent on point one because a small size of memory per object guarantees fast accessing. It also depends on point two, since the stored data might be transferred encrypted and consequently the access is more inefficient since the data has to be decrypted first.

After establishing the selection criteria, we have to examine the languages to choose from in the following section.

5.2.4 Comparison of Possible Implementations

As we already know, the NEXUS system is mainly characterized by its distributional nature and the object-oriented structure. Furthermore the openness and extendibility is of fundamental significance. This results obviously in requirements to the way of storing data.

First, the data must be simple to read and edit, hence easy to maintain. Eventually, NEXUS Objects should be similar to web pages, generative and changeable by anybody. On the strength of the demand for maintainability a language in ASCII seems to be suitable. The requirements for the openness and the standardized language lead to an XML-derived language such as the Geography Markup Language or to comparable structured languages like the Virtual Reality Modeling Language. As a matter of course we also want to consider XML itself.

VRML: The Virtual Reality Modeling Language is based on the OpenInventor ASCII file format of Silicon Graphics Inc. [OPENINVENTOR 1994]. When developing VRML the emphasis laid on the creation of virtual realities. Different from the Geography Markup Language, VRML supports the rendering of scenes, i.e. animated graphic sequences. Moreover, VRML is almost exclusively used for representing 3D data in Geographical Information Systems.

VRML is formed by a hierarchical structure of scene graphs which consist again of an arrangement of nodes. Nodes can be shape nodes (cylinder, cone, etc.) appearance nodes (FontStyle, material, textures, etc.), transformation nodes (rotations, translations, etc.), camera nodes, light nodes and group nodes [VRML 1.0].

Considering the selection criteria for VRML, no major problems arise. Concerning memory there is not very much to say because it is expected to be rather equal for all, GML, XML, and VRML, which is due to the relationship of the languages among themselves. Therefore the efficient processing will not differ much as well. The facile integration into the system is guaranteed, since W3D.ORG offers the source code of the Cosmo VRML browser. Hence equipped with the appropriate extensions this VRML browser would be easy to integrate into the system. Concerning the portability it can be said that some data originated to geographical informa-

tion systems already exists in VRML format. Programmes for transforming GML to VRML do not exist, since VRML describes mostly 3D representations and GML two-dimensional ones. As a matter of course it is not possible to derive 2D data from three-dimensional shapes. The maintainability should be provided by the standardized data format. Moreover, editors can be used, such as CosmoWorld. The model's extensibility is more difficult to deal with. Admittedly, there is the possibility of defining own data types but this works just rudimentarily, and only data records can be produced. By using X3D or VRML2000, which is an extended VRML version adopted to XML, the definition of new data elements is more facile to handle. More extensive and more complex structures can be described.

GML: The Geography Markup Language is, other than VRML, designed for static data. Here, no animated scenes can be rendered. GML was developed by the OpenGIS Consortium [OGC 00-029] and supports mainly the geometric representations of two-dimensional data, hence surfaces and polygons. Its objective was to provide a uniform description language for data originated from geographical information systems.

Considering the suitability of GML, one topic is the facile portability of the data. GML is the standard data format for exchanging 2D data. Therefore the majority of that data should be present in this format or should be transferrable by converters. GML is not used for visualizing the data. In applications GML data is transported into SVG (Scalable Vector Graphics), VRML or VML (Vector Markup Language). The integration into the NEXUS system can be guaranteed by the open sources for the transformation algorithms from GML into the visualization format. The extensibility is assured by the fact that GML is an XML derivation so that new data elements can be defined via XML.

XML: If GML and VRML had to be extended by XML constructs for embodying all NEXUS objects and structures (GML and VRML are used for representational data only), then we could also use XML at once. The GML data generated by third party organizations can also be described via XML and therefore easily integrated.

5.2.5 INHERITANCE WITH XML

Now we want to show an example of a NEXUS class transformed into XML. Therefore we give the reader an example of the DTD.

XML 5-1: Structure of NavigationNode

```
(0) <!ELEMENT neXus:NavigationNode
      (neXus:Attributes*, neXus:Position,
       nnode:Restriction*)>
(1) <!ATTLIST neXus:NavigationNode neXus:id ID #REQUIRED>
(2) <!ATTLIST neXus:NavigationNode neXus:name CDATA>
```

```
(3) <!ATTLIST neXus:NavigationNode neXus:kind
      (real|virtual) „virtual“ #IMPLIED>
(4) <!ATTLIST neXus:NavigationNode neXus:partOf IDREF>
(5) <!ELEMENT neXus:Attributes ANY>
(6) <!ELEMENT neXus:Position EMPTY>
(7) <!ATTLIST neXus:Position ...>
(8) <!ELEMENT nnode:Restriction EMPTY>
(9) <!ATTLIST nnode:Restriction nnode:weight CDATA>
(10) <!ATTLIST nnode:Restriction nnode:vehicleType
      NMTOKENS>
(11) <!ATTLIST nnode:Restriction nnode:partOf IDREF>
```

As we can see in XML 5-1 Structure of NavigationNode we have to define elements that should have been inherited such as id, name, kind, partOf and attributes. Furthermore, imported attribute types like Position in line (7) must be declared twice. Of course it would be better to have an XML system that supports inheritance and typing. XML Schema [XML SCHEMA 2000] supports inheritance and the definition of own attribute types.

Thus, we decided to use XML Schema to describe the Augmented World Modeling Language. In Figure 5-4 we can see the placement of the Augmented World Modeling Language together with its origins. GML3D is needed to describe three-dimensional data by using the Geography Markup Language. But GML3D has not yet been developed so that we have to use GML at the moment which is restricted to two dimensions.

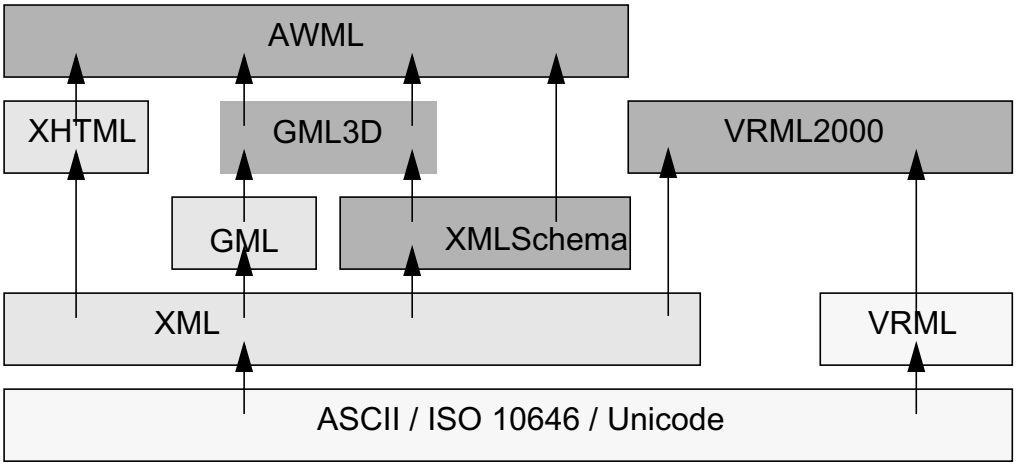


FIGURE 5-4: **Development of the Augmented World Modeling Language**

5.3 THE NEXUS NAVIGATION SYSTEM

This section deals with the development of the navigation system and with problems arising due its complex structure. First, we want to quote the reasons for developing the navigation network. We also try to give some alternative possibilities. Then we take a closer look at the resulting problems.

When we navigate through a town and want to reach a room within a specific building, we have to use two Augmented Area Models: one containing the town and another one containing the ground plan. Comparing the city map with the ground plan of the building within the town, it is very likely that we are not able to make out the wanted building on the city map. But our navigation system has to manage this if it wants to navigate us to the named building. The problem is to find entry points for Augmented Area Models of lower detail level that are derived from models of higher detail level. In our example the hierarchical structure is as follows: building - room - door. Now we want to provide some solutions for recognizing entry points:

1. *Finding entry points by comparing the outlines:* The spatial model service might be able to find entry points automatically if it manages identifying two identical objects (of different detail level) in different Augmented Area Models. A bijective function assures the one-to-one projection of two points that are identical in the real world but belong to different scaled models in NEXUS Figure 5-5. The only information one needs is the parent representation of the current representation which should be the most detailed one, and the scales of the different representations.

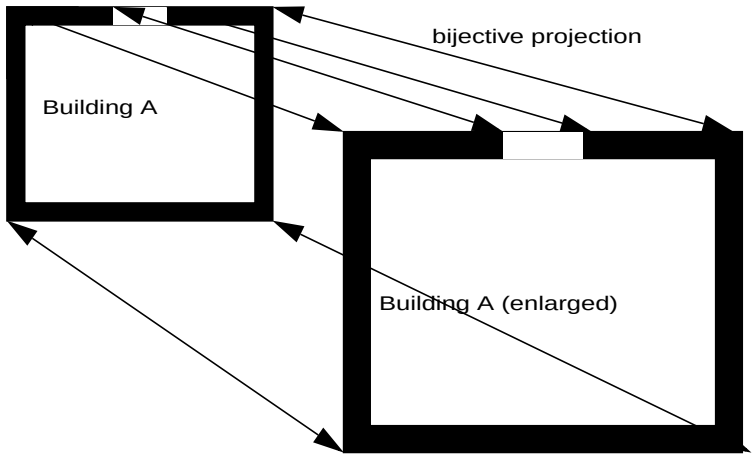


FIGURE 5-5: **One-to-One Projection of a Building with different Detail Level**

Unfortunately, this feature will not work if the automatic identification fails. Additionally, we are not able to define the type of mobile user which is able to use the entry points.

2. *Using the NEXUS Object Locator for identifying identical objects:* A second solution would be to handle links by deducing the kind of relationship out of the NEXUS Object identifier. In this case, the identifier has to be the same for all real world objects as discussed in section 5.1.1: ”Generating Unique Object Identifiers”. Therefore we need either a unique identifier for a model-object-pair or a non-ambiguous key for one real world object extended by a counter for the data instances. But this does not solve the problem concerning the definition of mobile user that may use the links. If we trusted in an Augmented World with only cooperating Augmented Area Models, each model would know any other model and could therefore use the same identifier for objects representing the same real world objects. Of course, this is not the case and we have to find another method, nonetheless (see section 5.1.2: ”Augmented Areas: Cooperating versus Competitive”).
3. *Manually creating links:* Thus we need a way for setting up the links manually. A possible solution is the storage of certain entry points, e.g. the position of a building’s door. That means that one needs an additional link object for building directional or non-directional typed relations: the navigation nodes and edges. A navigation node that is part of the city map model can be connected to a navigation node inside the building (consequently part of the building’s Augmented Area Model). Thus an entry point for navigation is defined.

The solution suggested last is the best one for it works in any case whereas the functionality of the first and the second proposition cannot be guaranteed. However, the solution presented last does also have its difficulties. The complexity of the elaborated navigation network can be very high and therefore path finding algorithms might possibly take a lot of time. So we worked out three approaches to reduce the complexity and consequently decrease the time consumed for performing Dijkstra.

1. *Distribution of navigation network:* The navigation network should be distributed corresponding to the type of their restriction, so that we obtain Augmented Area Models containing only street nodes and edges, for instance. So we are able to pack more information into one Area Model and hence need not switch between different models that often. So the connection time can be minimized.
2. *Establishing short cuts:* As already shown in section 4.2.2.7: ”NavigationEdge”, short cuts between important navigation nodes can be generated so that these nodes are visited earlier when executing Dijkstra [SEDGEWICK 1995]. Nonetheless, the advantage is not eminent because we cannot create short cuts for every node without increasing the complexity. Increased complexity would wreck any advantage gained as a consequence.
3. *Leaving out information:* When reducing the navigation nodes and edges to the connection data only, restrictions and weights need not be evaluated. This helps us reducing time when retrieving information about the edges’ or nodes’ attributes and also forces down the execution time consumed by applications. On the other hand we lose functionality because we cannot determine the type of mobile class that may pass the edges and we cannot predict any duration of journeys.

5.4 Exemplary System Behaviour for a Scenario

Now we want to describe an example for the system's behaviour when mobile users want to book a flight. Imagine two people that want to book a flight from the airport next to their home town to Paris in France. In this case we assume that there is no external database where flight schedules and connections can be looked up so that the objects provided by NEXUS have to be used.

The procedure can be subdivided in four major tasks. We presuppose that all the necessary data has already been entered, i.e. desired departure time, number of seats to reserve and destination airport.

1. *Finding the closest airport to mobile user's current position:* The user's sensor sends the current position to the NEXUS system. Then the Area Service Register is queried for classes of type airport. We remember that the Area Service Register returns a list of Augmented Area Models that contain the requested class. The search for neighboured areas can be improved by using spatial indices ([HÄRDER & RAHM 1999], chapter 9). After receiving the closest airport we have to query the corresponding Augmented Area Model for navigation nodes restricted to planes. We have to remember the identifier of this node for consecutive queries.
2. *Retrieving the necessary identifiers for Paris' airport:* By using the Name Service, we are able to find the identifier of airport Charles-de-Gaules in Paris. The identifier is of type NEXUS Object Locator and consequently contains information about the Augmented Area Model it is situated in. Thus, we know the Augmented Area we have to query for navigation nodes restricted to planes. The resulting identifiers are those of the destination airport and will help finding the navigation edge connecting both nodes.
3. *Getting the navigation edge that connects start and destination airport:* After we got to know the navigation nodes that are connected by the wanted edge, we are able to query the Augmented Area Model containing the navigation network for airplanes. If there is no navigation edge connecting both airports, the starting airport obviously will not offer any flights to the destination airport. Thus we would have to start again at point one finding the next unvisited airport close to the home town.
4. *Looking for available seats on the flight:* If there is a flight offered from the home town to Paris, we have to check the plane for available seats. The objects Schedule, which stores arrival and departure times, and ServiceObject, which specifies the kind of freight to be transported, are both part of the Augmented Area Model managing the navigation network for planes. So we can query this model for a passenger flight at the desired time. ServiceObject also declares the identifiers of the flight's seats. The seats however are part of the Augmented Area Model that manages the flight. Nonetheless, we are able to extract the Augmented Area Model managing the seats out of the identifier and therefore are able to access the seat's data. If there are no seats left for the desired flight, there is the possibil-

ity of proposing another flight or to look for another airport by restarting at point one.

The example above illustrates that we always try to avoid connecting several Augmented Area Models for performing a query, and this is very important. We will give another example: whenever we have to switch a model, we have to establish a connection via TCP/IP - a time consuming effort. Therefore we have to make sure that queries are spread on different Augmented Areas as sparsely as possible. The solution can be done by providing the possibility of distributing data objects. So we can distribute the data of a street on different Augmented Areas whereby each model stores information that might be relevant for one application. An application related to public transportation stores only the street data that is necessary for navigating buses and storing timetables, whereas a car navigation system stores all the data that is relevant for personal navigation. Remember the structure of NEXUS identifiers: data objects representing the same real-world objects have the same identifier. The only difference is the Augmented Area identifier part. In this case, the second part of the navigation node's identifier is the same for the car navigation node and for the bus navigation node, but the Augmented Area part is different. So the navigation node can be identified as representing the same object and nevertheless be distinguished by the Augmented Area Locator.

5.5 SUMMARY

In this chapter we examined some implementation aspects. First, we considered distribution and discussed the generation of object identifiers in NEXUS. We found out that a combination of an identifier pair consisting of an area model identifier and an object identifier is the best solution. With this method we are able to determine the area an object is situated in by simply viewing the NEXUS Object Locator (the pair consisting of area and object identifier).

Next we have taken a look at Augmented Areas and hinted at problems concerning the cooperation between different models. There are Augmented Area Models that are directly and indirectly cooperative, and competitive, whereby competitive ones do not support each other. Therefore no links between competitive models exist. For providing a cooperation a language construct is needed that expresses that two objects are identical.

Consistency among diverse area models is the last distributional aspect we talked about. Here we used the Event Service for observing objects and notifying applications or objects in case of a modification. But the consequences of a reported modification have to be handled by applications and providers. Consequently, consistency can be supported but not be guaranteed for all data objects. It is still up to the user to avoid inconsistency.

In the next subsection we were discussing an appropriate data exchange format. There were three possibilities for implementing the Augmented World Modeling Language: GML, VRML and XML. Since GML and VRML are used for showing 2D and 3D models without exception, they seemed to be insufficient. Thus we decided to use XML. But XML also proved to be insufficient regarding the concepts of inheritance and type definitions. Consequently, we came to the solution of using XML Schema which supports both, inheritance and the definition of own attribute types.

Another important section within that chapter was the discussion about the functionality of the navigation network. First, we remarked that the automatic generation of a navigation network is difficult and cannot be relied on. In any case the automatic generation misses information about the restrictions concerning mobile users that are able to use the network. Hence, we proposed the manual creation or at least the editing of navigation networks. Whether using the manual or the automatic navigation, in any case the resulting network might be very complex. Therefore we examined some possibilities of reducing the complexity. We found three ways, namely distribution according to restrictions, short cuts and the omitting of information. But each proposed solution had its advantages and disadvantages.

Last, we presented an example showing the functionality of a typical query taking into consideration the system components.

CHAPTER 6

EXAMPLE SCENARIO

A perfect method should not only be an efficient one, as respects the accomplishment of the objects for which it is designed, but should in all parts and processes manifest a certain unity and harmony.
(George Boole, An Investigation of the Laws of Thought)

This chapter illustrates the workability of the Augmented World Standard Class Schema. An exemplary scenario deals with aspects like navigation, events, transactions of money, services and goods, and information hoarding. Therefore the scenario is divided into sub-examples that are examined. Each sub-example provides graphics in Unified Modeling Language that exemplify the instantiations and aggregations of the standard classes.

6.1 DESCRIPTION OF SCENARIO

In the following we want to present an example scenario which fulfils most of the requirements and expectations to an Augmented World Model. It deals with the aspects of pathfinding and navigation and of querying data dependent on the user's location. Moreover, it considers the problems of linking different kinds of data objects with each other, such as timetables for public transportation and the corresponding transportation vehicles.

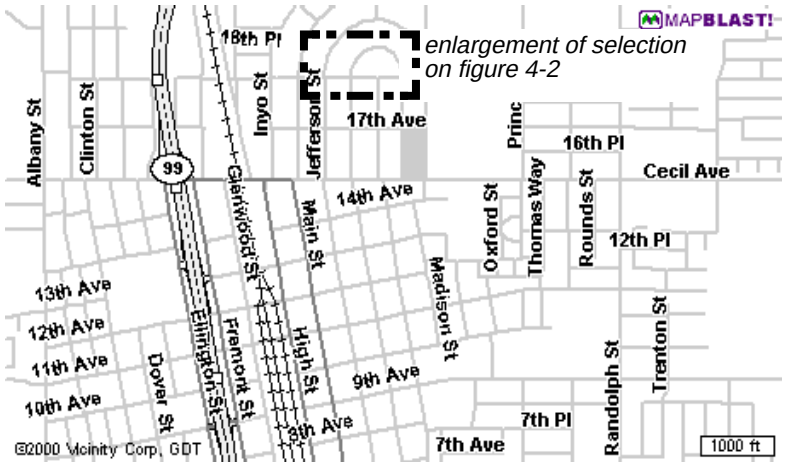


FIGURE 6-1: **Street Map of Fisher's Home Town: Delano, FLA**

Assuming a couple Fisher who wants to visit an economic centre 2000 miles away from their home town Delano on the weekend. The reason for this idea is an appointment of Mr Fisher with his business colleague Mr Andrews on Saturday. This seems to be a good occasion to spend a weekend in BigTown. First they have to book a flight from Delano to BigTown. This is done automatically by the NEXUS application. Furthermore the NEXUS application presents a list of hotels in BigTown where Fishers could stay overnight. Fishers decide to reside in a Hilton near the centre of BigTown. NEXUS does the booking for the hotel and confirms the execution. The same procedure is performed with a rental car.

In the morning of their leave, Fishers turn on their portable NEXUS client in the car for getting the route from Delano (see Figure 6-1) to the airport. Since it is early in the morning there is only little traffic on the highway. Therefore NEXUS is able to take the shortest path from Fishers' home to the airport. Arriving at the airport, NEXUS connects to the airport services which include a management system for available parking spots. After directing Fishers to a free parking spot, Mr Fisher unplugs the NEXUS client from the car for taking it with him. In BigTown Fishers' NEXUS client directs the family to the rented car that is parked for them on the car-park after they have arrived.

Mr Fisher reinstalls the portable client in their new vehicle and connects the navigation control of BigTown. NEXUS calculates the possible paths to the Hilton where they booked the rooms. While computing the potential shortest path from the airport to the hotel Fishers have booked, the application realizes a traffic jam that would obstruct their passage. For avoiding a delay NEXUS reroutes them on an alternative route.

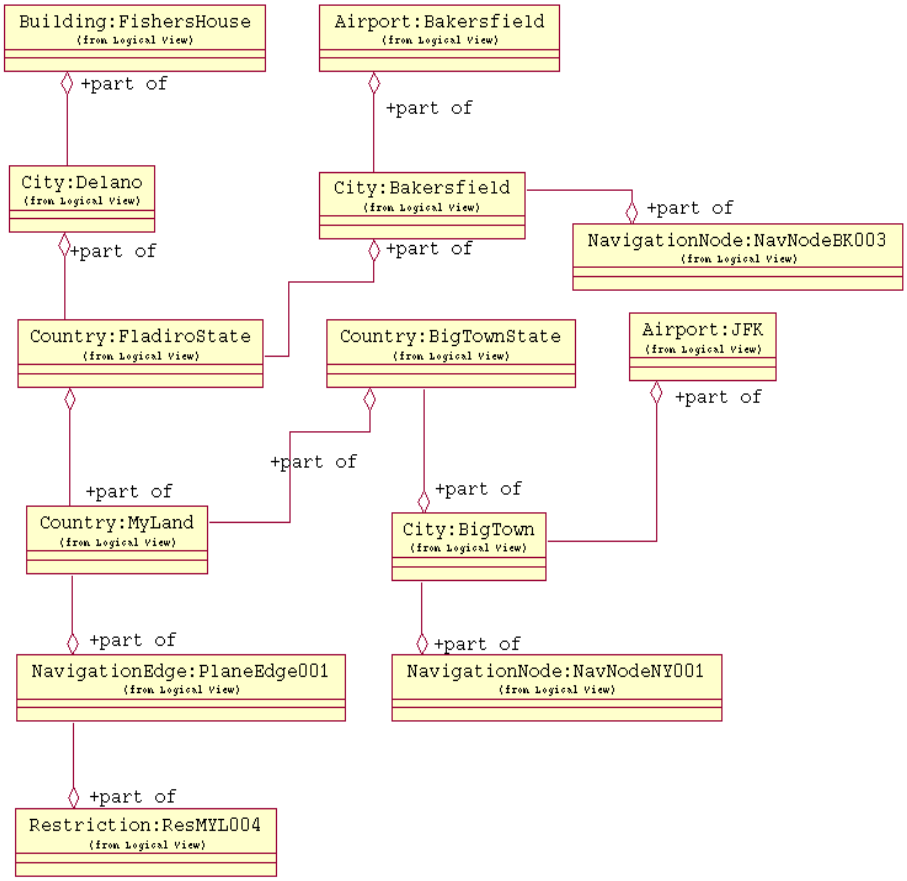


FIGURE 6-2: Overview for Finding closest Airport

Near a traffic light in the centre of BigTown the car in front of Fishers´ crashes into another waiting vehicle. One of the accident cars is equipped with a NEXUS client that immediately sends an emergency signal to the ambulance, informs the local authorities and places a virtual warning triangle for approaching vehicles.

After arriving at the hotel the check-in is done nearly automatically. In the lobby of the hotel there is a virtual information tower which is advertising for the hotel's services. Fishers' NEXUS client realizes it and shows the information that is connected to the virtual information tower. So Fishers get informed that there will be a servant available in the next couple of minutes to show them their rooms. At the same time

the hotel´s NEXUS client perceives the newcomers and informs a servant. This employee gets all necessary data, such as the room number and potential additionally booked services.

FIGURE 6-3: **Bakersfield Airport**

Shortly after Fishers have been brought to their room, Mr Fisher has to leave the hotel for his meeting. Again Mr Fisher is in need for help of the NEXUS system since he has never met his business partner before and does not know how he looks like. So the system supports him in arranging a meeting point with his colleague. NEXUS knows the position of both men and can therefore synchronize their routes. This task has to be done dynamically dependent on the men's current position. Thus, a NEXUS server should be accessible on the route that both men are going to take. Since the way can be estimated beforehand, it is also possible to take into account areas with lower bandwidth. Areas with lower bandwidth might not be able to provide sufficient bandwidth for all the data needed for dynamic navigation. As a precaution information hoarding should be performed and the necessary data should be stored before Mr Fisher leaves the hotel. Fortunately, there is an info-station in the hotel's lobby that offers both, the data and a broad bandwidth.

After eventually coming together, both men decide to look for a restaurant. They would like to negotiate their business affairs while having lunch. Their preference are Italian restaurants, so they query for the closest restaurant of this type. NEXUS navigates them to the restaurant and already transfers their orders after consulting the menu.

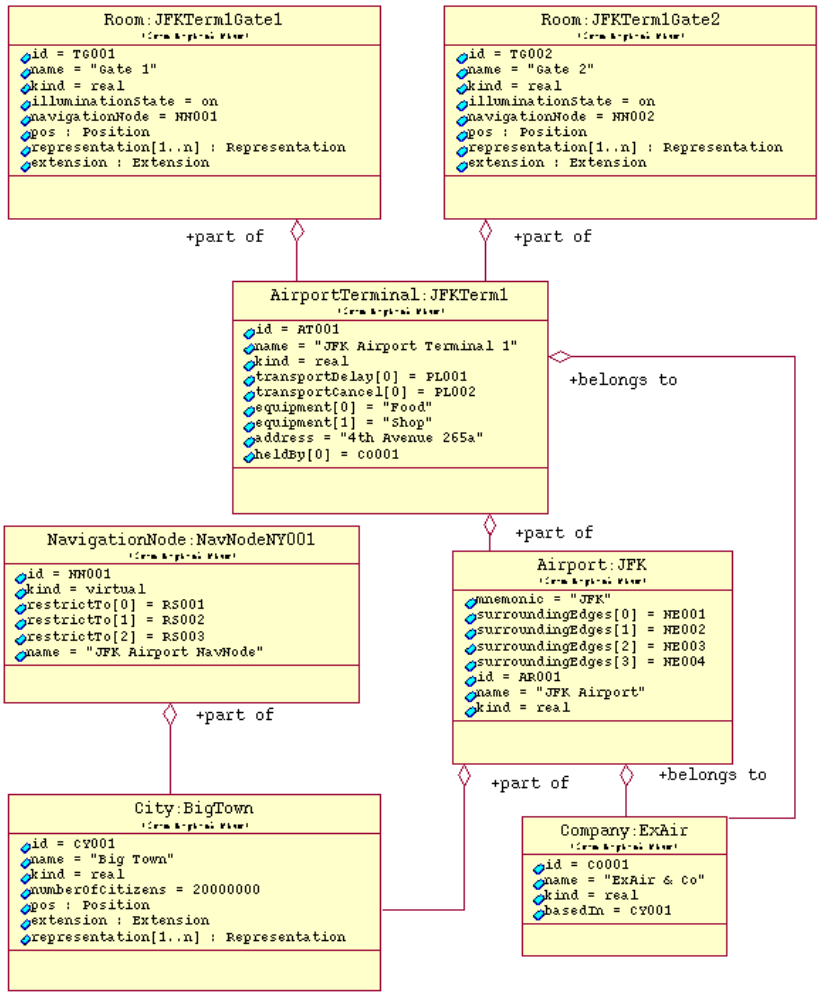


FIGURE 6-4: BigTown Airport

In the meantime Mrs Fisher is on a shopping tour through BigTown. She arrives at a mall where she is tempted to buy a new fur coat. Nonetheless, it should be good value for money and of top quality. The mall offers its current fitting stocks with small pictures, prices and product information. After Mrs Fisher has performed a preselection she examines the remaining articles more precisely. By pointing at the articles with a special device like a telepointer, she gets a more detailed information on the products. After a short while Mrs Fisher settles for a coat that pleases her particularly.

The next day Fishers are planning a sight seeing tour. Unfortunately, their hired car does not restart work, so that they have to use the public transportation. Of course they do not need a tourist guide because they are able to consult NEXUS. The application prepares a route for them, which leads along all sights of interest respectively taking into account the timetables and transportation lines whenever they want to travel forth. If Fishers want to receive more detailed information on sights, they simply have to follow hyperlinks on their display or they have to point at remarkable objects with their telepointer.

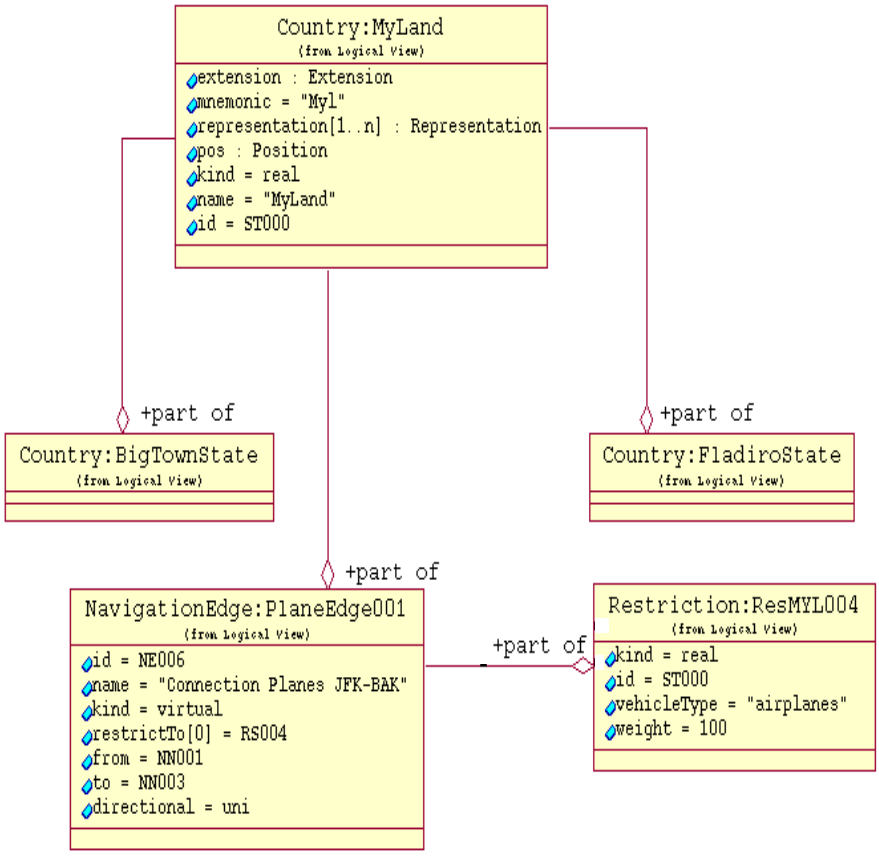


FIGURE 6-5: Connection between Cities

After undergoing a pleasing afternoon Fishers have to pack their bags and prepare for their departure. The hotel charges automatically all used and by NEXUS registered services to their bank account and says goodbye to the couple.

6.2 REALIZATION OF SCENARIO

In the following section we examine the scenario further and treat the emerging applications with more detail. We are able to excerpt nine sub cases or applications out of the scenario.

1. *Reservation and booking of the hotel and the flight:* The first application deals with the booking of the flight to BigTown and the reservation of the hotel and the rental car. This also includes the search for the closest airport that offers the desired flight.
2. *Navigation:* The second task is the navigation from Fishers' house to the airport of Bakersfield and the navigation to a free parking spot.
3. *Event-based accident support:* Whenever an accident occurs NEXUS should be able to inform the local authorities and to reroute the affected vehicles.
4. *Iterative navigation:* Another application dealing with navigation is the routing through BigTown while dynamically occurring obstacles like traffic jams or accidents blockade the preferred way. Thus the paths have to be calculated several times whenever the state of the navigation network changes[1]. This also includes the navigation of two mobile users wishing to meet each other.
5. *Transferring money:* After that we deal with the hotel management system, and the charging of bills on accounts of legal entities via NEXUS.
6. *Virtual information towers:* There is a virtual information tower in the hotel's lobby which offers information on the hotel's services.
7. *Information Hoarding:* An info-station can also be found in the lobby, which is utilized for pre-storing data that can be used by NEXUS applications at a later date.
8. *Querying the database depending on the current position:* The next task is to query a position depending on the attributes of the queried objects or other objects. In this case we query for the closest Italian restaurant.
9. *Virtual shopping:* Now we pay attention to the mall information system that is visited by Mrs Fisher on her shopping tour.
10. *Navigation depending on timetables:* Finally Fishers' sight seeing tour that includes navigation regarding timetables of public transportation will be examined.

1. Naturally a change in the state of the navigation network is either caused by a blockaded navigation edge (traffic jam, accident) or a removed or added edge / node.

6.2.1 RESERVATION AND BOOKING OF SERVICES

BOOKING THE FLIGHT

The queries that are necessary for booking the flight can be separated in 5 sub-queries or actions: the query for the destination airport, the query for the amount of desired seats for the flight, the nearest airport to the current residence which offers flights to the destination, the query for available flights from the nearest airport at the preferred point of time and the reservation of the places along with charging the costs to Fishers' bank account.

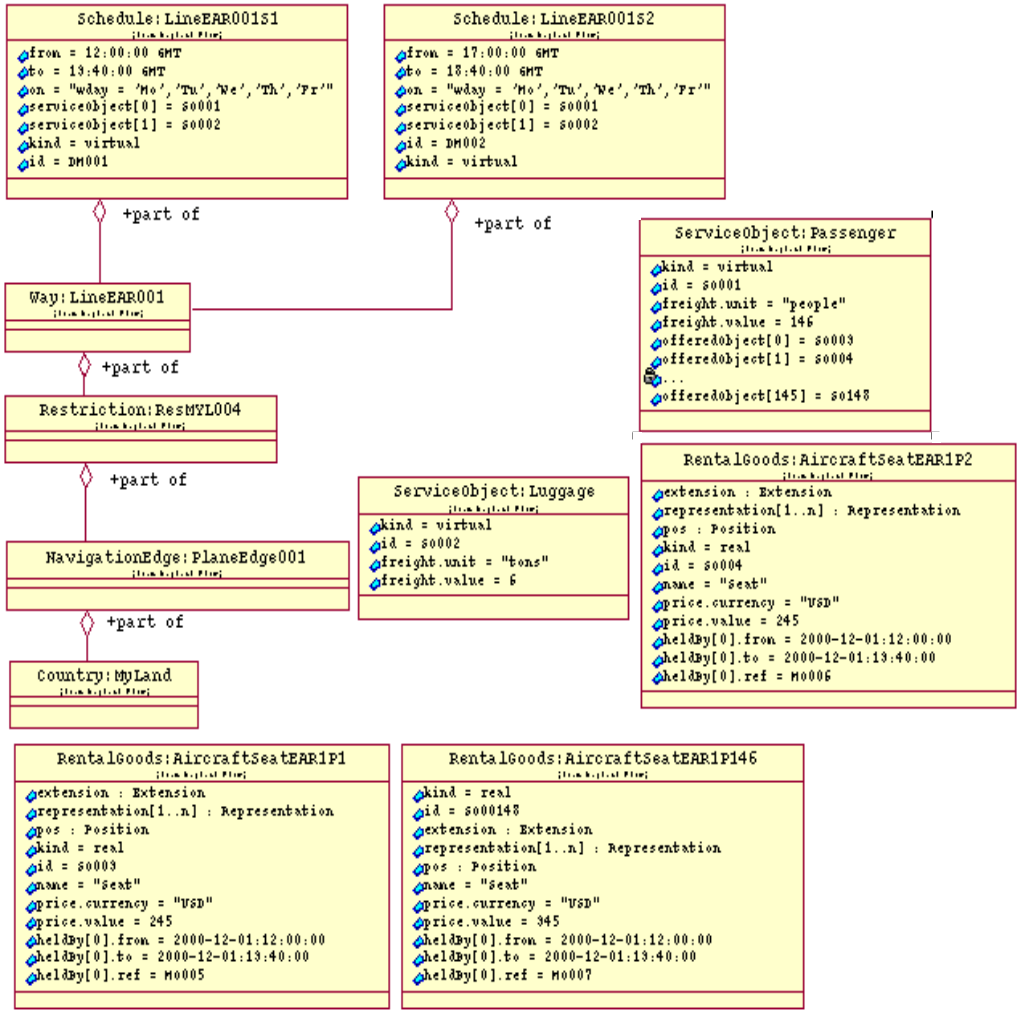


FIGURE 6-6: **Reserving the Seats for the Flight**

Figure 6-2 provides an overview of necessary classes for finding the closest airport that offers a flight to the destination airport. There you can see a part of the regional

surroundings of Fishers' home town including Delano and the closest city that has an airport. Usually the query for the nearest airport can be done by comparing distances between airports and Fishers' home town.

If we want to be sure that the closest airport offers a connection to the destination airport we have to model these connections as well. Therefore we use navigation nodes and edges just like nodes and edges in graph theory. Edges can be directed or not (one-way-streets, flights) and weighted, where the weight expresses the estimated time for passing the edge with a corresponding vehicle (can also be a wheel chair or on foot). Obviously, the weight is dependent on the kind of vehicle or movement type. Thus, we specify several restrictions to each edge whereby each restriction expresses the vehicle or movement type. We also offer some attributes that can restrain the restriction further in addition to the weight that is part of every restriction. So we can restrict edges to trucks that weigh less than five tons and are lower than 10 feet.

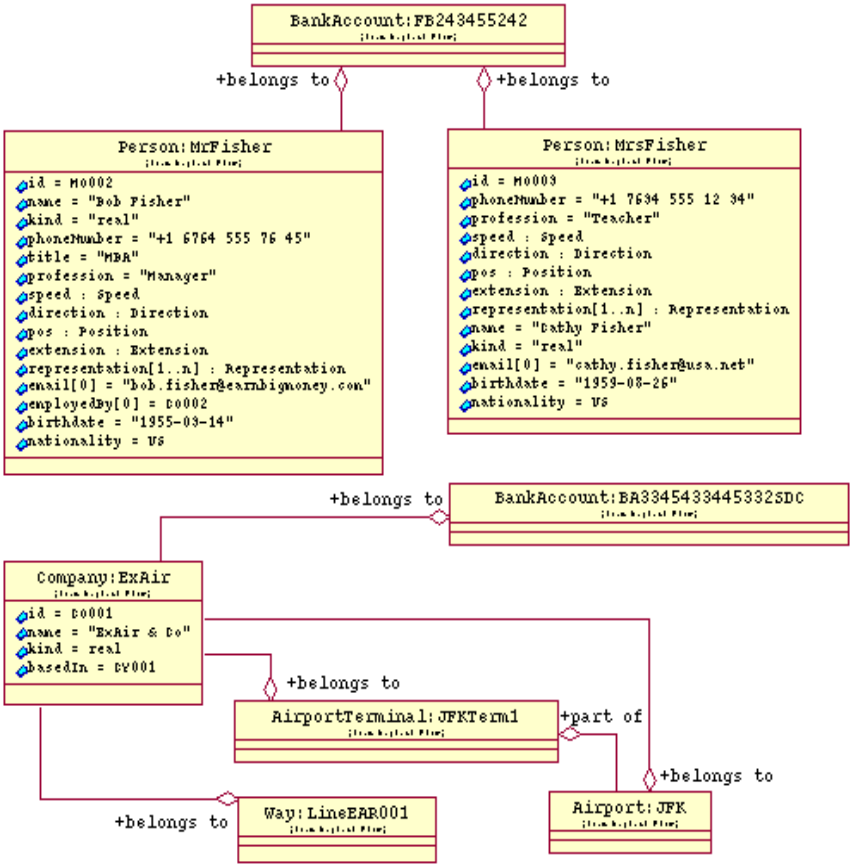


FIGURE 6-7: **Money Transfer for Flight Booking**

Navigation edges connect navigation nodes that are always less or equal restrictive than their connected edges. A navigation node is the union set of restrictions of the

adjacent edges. Thus we are able to model transfer points between different kinds of movement. A navigation node that is connected by a plane edge and a car edge, is restricted to planes and cars and offers a possibility for changing the kind of movement from cars to planes and vice versa.

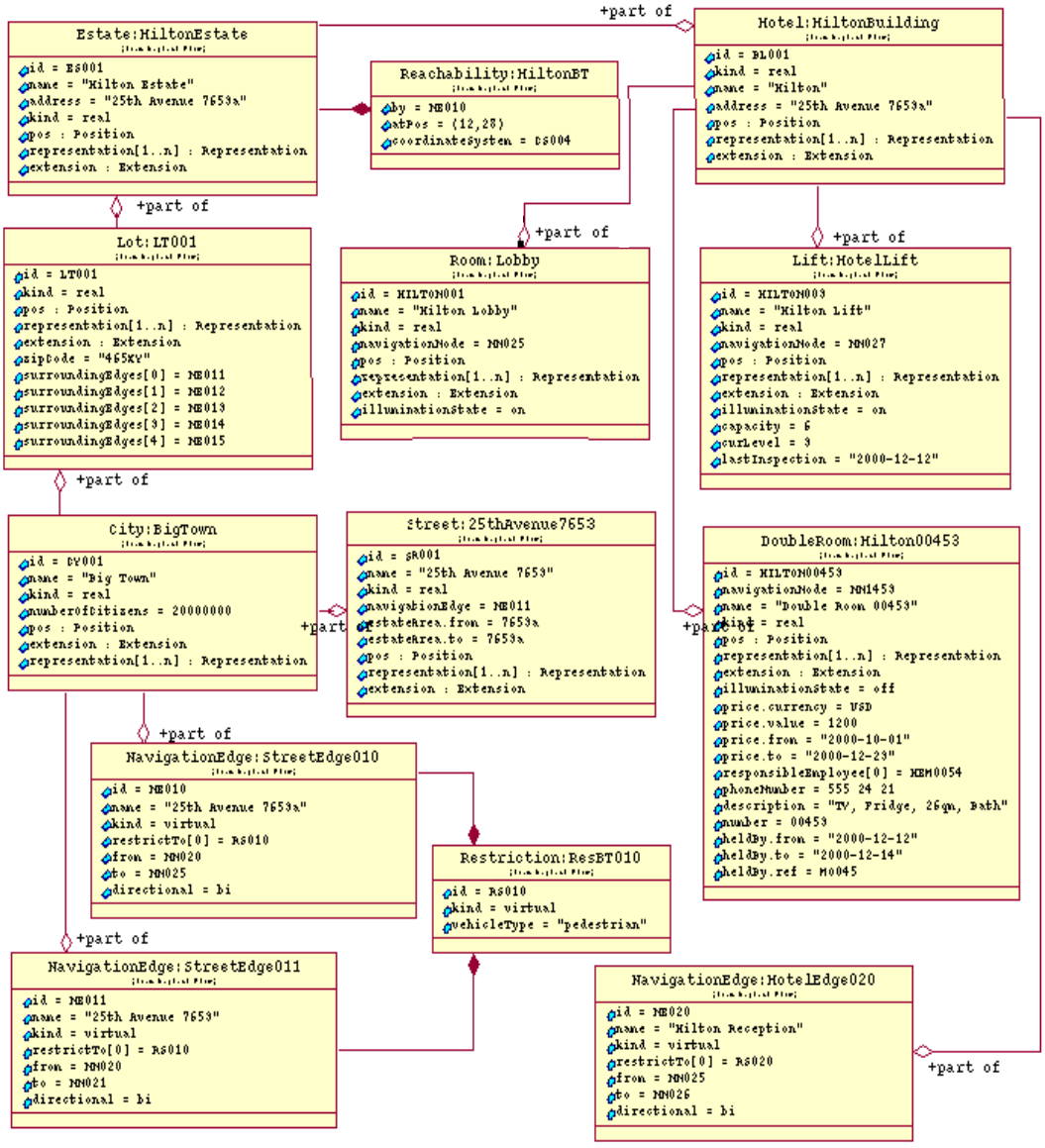


FIGURE 6-8: Reserving the Hotel Room

Figure 6-3 pictures the airport of Bakersfield and Figure 6-4 the airport of BigTown. Both airports offer navigation nodes that are connected by a navigation edge “PlaneEdge001” (see Figure 6-5). This navigation edge is restricted to planes and has a weight of 100 minutes. In other words: the flight from Delano to BigTown

takes 100 minutes. But we also want to know whether there are available seats on the flight. The plane might be already booked up.

In Figure 6-6 we take another look at the navigation edge for the plane. A way is connected to the restriction that specifies that there are only planes allowed on this edge (we left out this information in Figure 6-6 for we already showed the restriction data in Figure 6-5). A way declares a line for planes, ships, buses and other public transports. The way in Figure 6-6 has also schedule information that defines the timetable and the kind of freight that might be transported. In this example ServiceObject Luggage and Passenger are declared. By reading the freight.unit and freight.value attributes, we find out that there are seats available for 146 passengers altogether on this flight. If we want to know whether there are free seats disposable we have to check offeredObject. As we can see seat number SO003 is only booked up on the 1st December 2000 for the flight at 12 am. If there were any other entries, more heldBy attributes would have been declared. Now we assume that there were two available seats on the flight EAR001 that Fishers want to occupy. Of course they do also have to pay for the flight. So we take a glance at the money transaction.

In Figure 6-7 there is Fishers' bank account and the bank account of the airline. If we want to know whether the airline is the owner of the seats the belongsTo property should give us the appropriate identifier. So money can be transferred from one account to the other. Money transactions are dealt more detailed in section 6.2.5: "Transferring Money".

RESERVATION OF THE HOTEL ROOM

After booking the flight Fishers are in need of a hotel room for they have to stay overnight in BigTown. So they have to look for available hotel rooms that meet their demands. NEXUS offers them a variety of hotels that have some free rooms left. Therefore all hotels have to be queried for their heldBy that have an entry for the period of time Fishers want to stay in BigTown.

After going through a list of available hotel rooms the couple decides to spend the night in a Hilton. Figure 6-8 shows the hotel building and some exemplary rooms. One of these rooms is double room 00453 that is booked by Fishers. One can see that the room costs 1200 USD per night in the period from 1st october to 23rd december, which is drawn off Fishers' account.

Fishers' stay in the hotel is stored by using the heldBy attributes. A car can be rented similar to the booking of the flight and the reservation of the hotel. Cars are rental goods that are owned by a legal entity. The tenants are stored in the heldBy attribute of the car and the rent is drawn off their account credited the owner's account.

6.2.2 NAVIGATION TO DEFINED PLACES

Taking a look at the hotel in Figure 6-8 once again we also see navigation edges that can be used by pedestrians. Figure 6-9 illustrates the hotels surroundings on a map. Pedestrians that want to enter the building have to use navigation edge 20 that is part of the hotel building. It connects the entrance on the pavement in front of the hotel with the lobby within the hotel. This is an example for the standard design implementing an entrance to a building. Returning to the example where Fishers have to leave Delano for navigating to Bakersfield airport: first they have to quit their house. We can assume that they are able to find their way out on themselves so that they will not have to use NEXUS for this part. Even if they would not be able to - the procedure should be clear: rooms are modelled as nodes connected by edges whereby one edge is connected to the outside as seen in the hotel example.

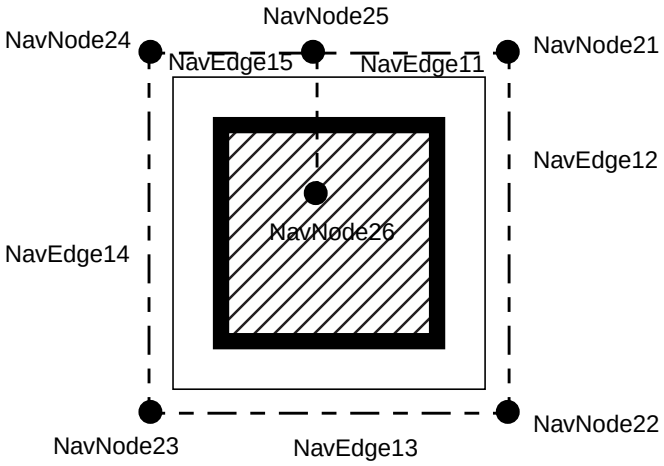


FIGURE 6-9: **Navigation Nodes and Edges around the Hilton**

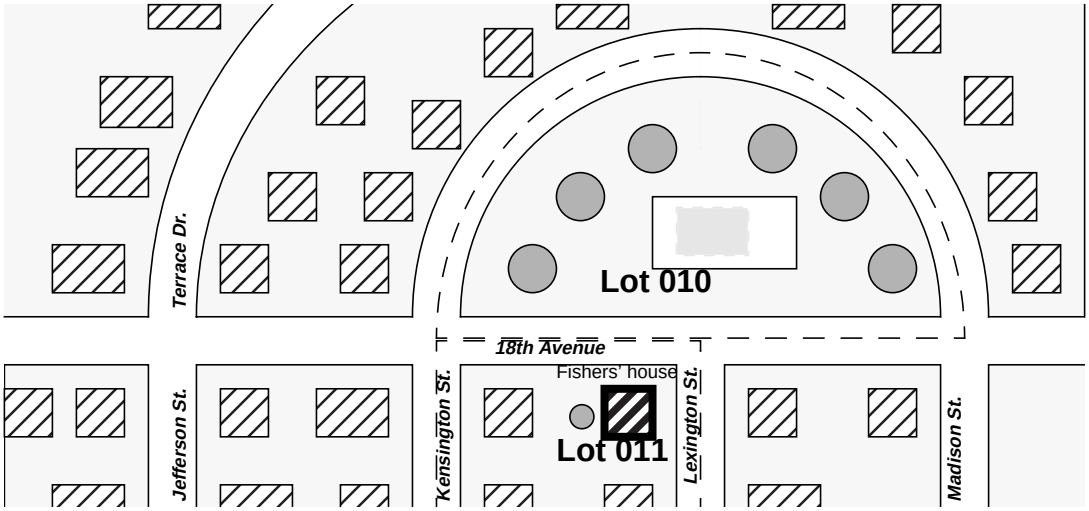


FIGURE 6-10: **Surroundings of Fishers' House**

Figure 6-10 shows the area around Fishers house. Corresponding to the Standard Class Schema we modelled the surroundings in Figure 6-12. There you can see Fishers' house, their estate, the lot where it is situated and some of the navigation nodes and edges. Navigation Edge 101 represents Lexington Street and Navigation Edge 102 the 18th Avenue (see Figure 6-11). In this example the 18th Avenue is restricted to trucks that weigh less than 20 tons and that are lower than 8 feet. It is also restricted to wheelchairs, pedestrians and cars. Every mobile user except pedestrians needs one minute to pass the edge. The restriction for pedestrians defines the mean time for passing the edge to 10 minutes. In Figure 6-12 Navigation Edge 101 and 102 share the pedestrian's restriction ResDL1012. Usually restrictions are not shared for the weight is different for varying edges.

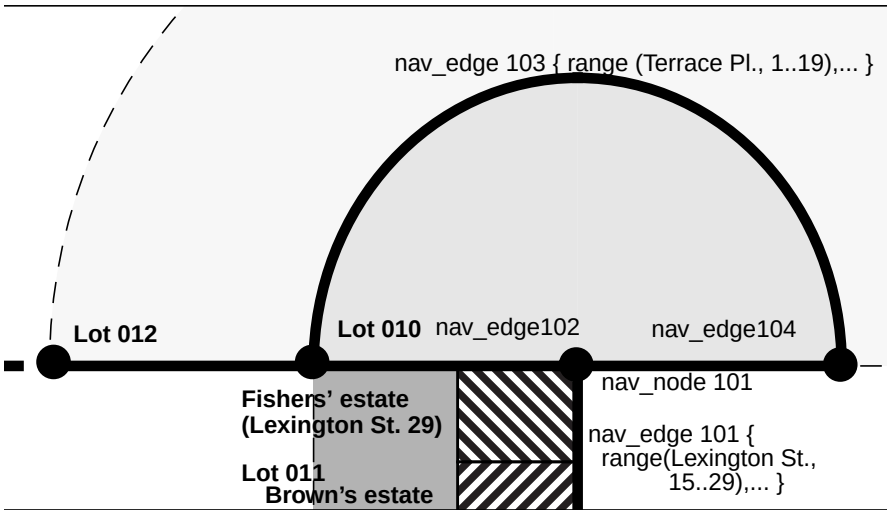


FIGURE 6-11: **Surroundings of Fishers' House (Navigation)**

Fishers' house does also include a Reachability. There, one can see that the house is reachable by navigation edge 101, i.e. Lexington Street at a defined position. The Reachability attribute can be used as an alternative to an additional navigation node in front of each house that connects it to the outside.

According to the Reachability attribute the front door should lead to Lexington Street. So we expect to leave them that way so that they have to pass Lexington Street as pedestrians. Assuming their car to be parked in the 18th Avenue they have to follow navigation edge 101 to the north up to navigation node 101 where they have to turn left for reaching their car.

After entering their car they are able to use all edges and nodes that are restricted to cars. Fishers are likely to pass several Augmented Areas on their way to Bakersfield. One server will probably not be able to manage all the data of the area Fishers have to pass. Figure 6-13 gives an overview of a probable distribution of Augmented Areas. Thus we have to examine the case where one navigation edge reaches from one Augmented Area to an adjacent one. In this case we have to remember section 5.1.1: "Generating Unique Object Identifiers". The second part of the edge's identifier accounting for the object locator is identical for both Aug-

mented Areas, the first part accounting for the area locator is different. So the edge can be distinguished in a distributed view and one is able to recognize that the referred navigation edges are identical in global view.

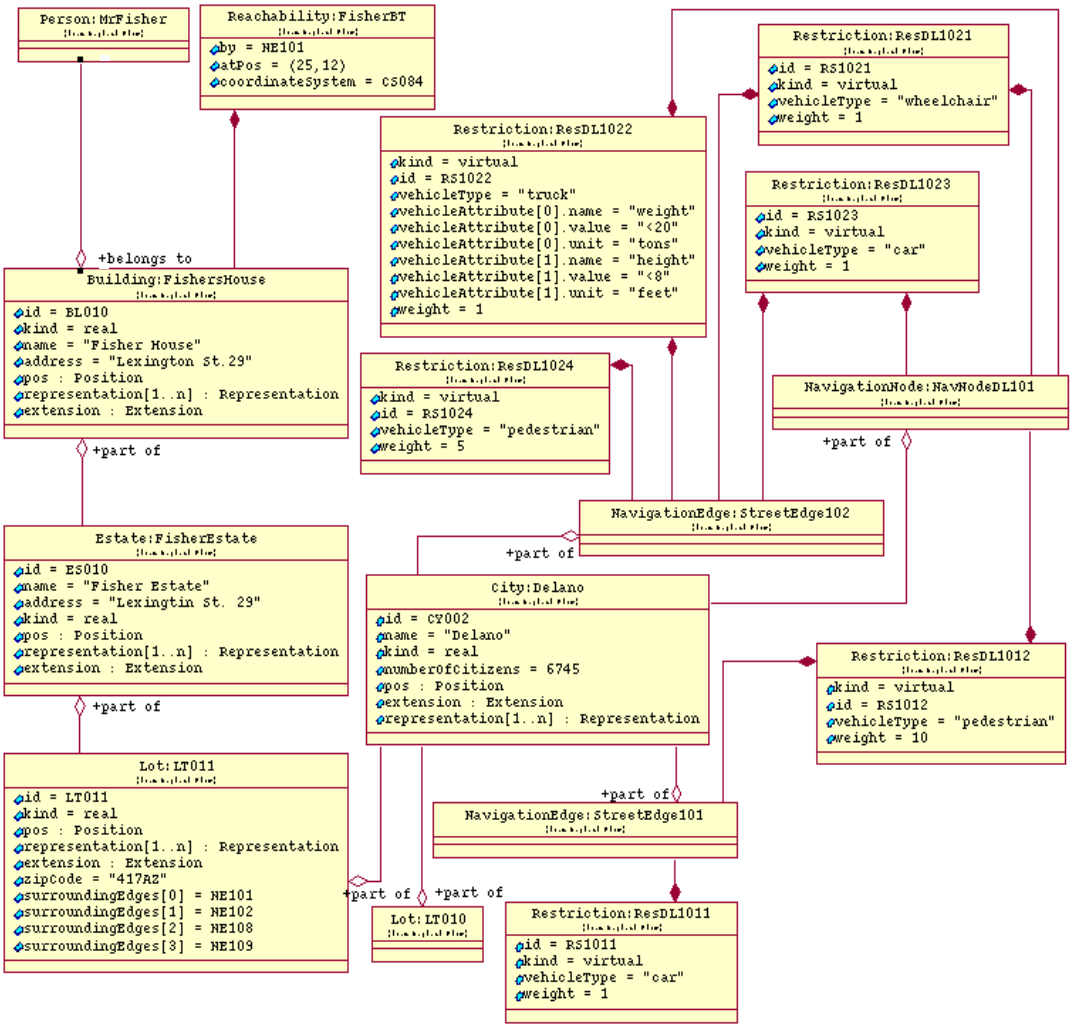


FIGURE 6-12: Surroundings of Fishers' House (Model View)

After arriving at the airport Fishers have to park their car on the provided car-park. The airport's NEXUS server manages the navigation to free parking spots. We designed an example in Figure 6-14 where the structure of a parking deck is shown. Similar to navigation in towns there are lots, navigation edges and nodes. Lots contain a set of adjacent parking spots which can be either free or occupied. If we

assume that Fishers arrive at the car-park at navigation node 051 they are able to follow edge 051 or 057 that are unidirectional.

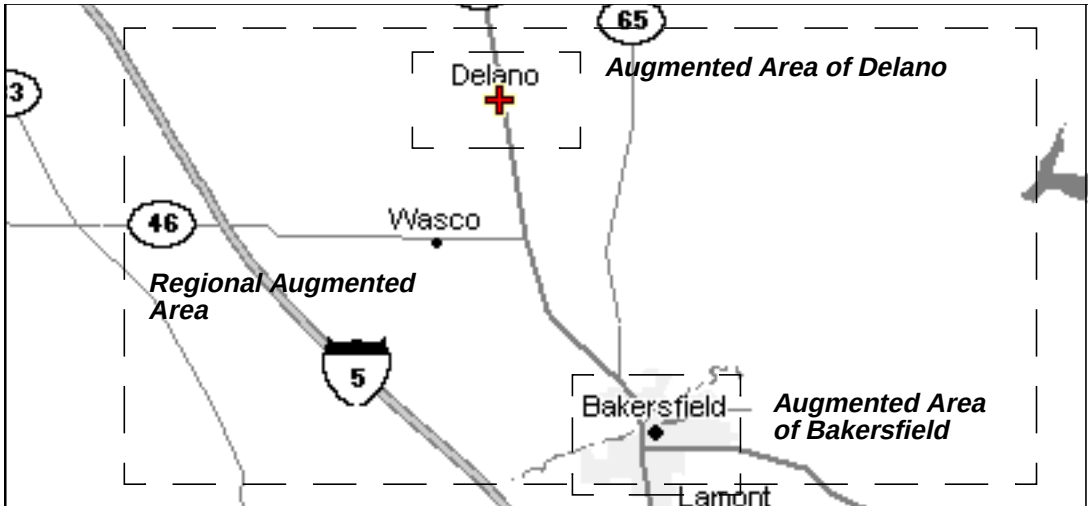


FIGURE 6-13: **Map of Augmented Areas**

The airport's navigation system directs the couple along edge 51 since all parking spots of lot 1 are occupied. In Figure 6-15 we can see an example for an occupied parking spot PS003 that is part of the SetOfParkingSpots LT052. SetOfParkingSpots inherits from Lot and is extended by one attribute, namely Occupied which declares the current state. If all parking spots were occupied this attribute is set to true, or false otherwise.

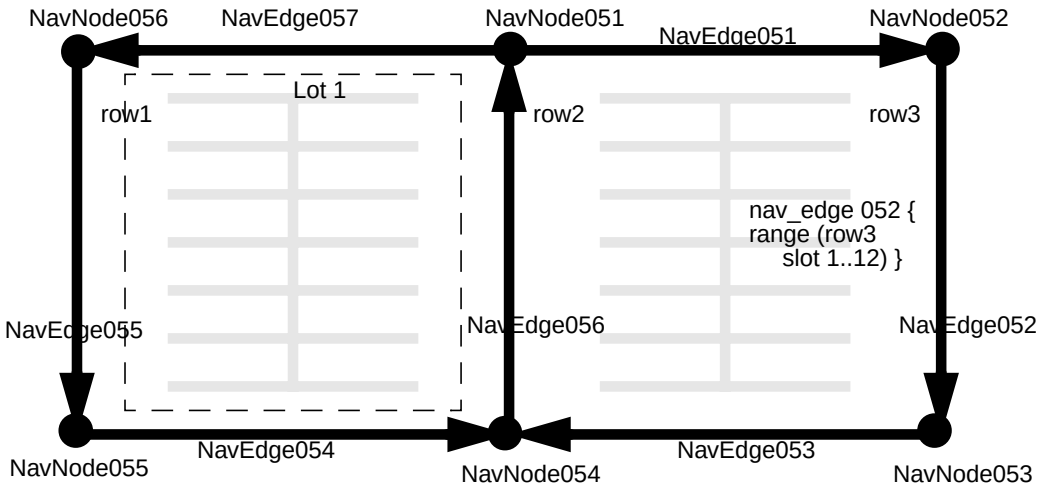


FIGURE 6-14: **Navigation on Airport's Car-Park**

In Figure 6-15 there must be free parking spots available because occupied is set to false. So the airport's navigation system directs them to navigation edge 56 where free parking spots are still disposable. Since each parking spot has a Reachability

attribute it is determinable which edge has to be used for reaching the parking spot. After parking the car on a free parking spot Fishers leave the car-park on the navigation edges and nodes for pedestrians that are not shown in Figure 6-14 for simplifying the picture. This example should be sufficient to show the operability of the Standard Class Schema's navigation system.

FIGURE 6-15: **Navigation on Parking Site**

6.2.3 EVENT-BASED ACCIDENT SUPPORT

When the couple is going to their hotel in BigTown there occurs an accident only a few cars in front of Fishers' car which blockades the shortest path to the hotel. NEXUS has to inform the local authorities that are setting up a Geocast for the affected

region whereby all vehicles within this area are informed about the accident. Also an event is created that informs all vehicles that are entering the affected area. This event puts the virtual warning triangle into practice. Figure 6-16 shows an example of the occurring accident. The Geocast object and the collided cars have been omitted for they are rather simple to imagine.

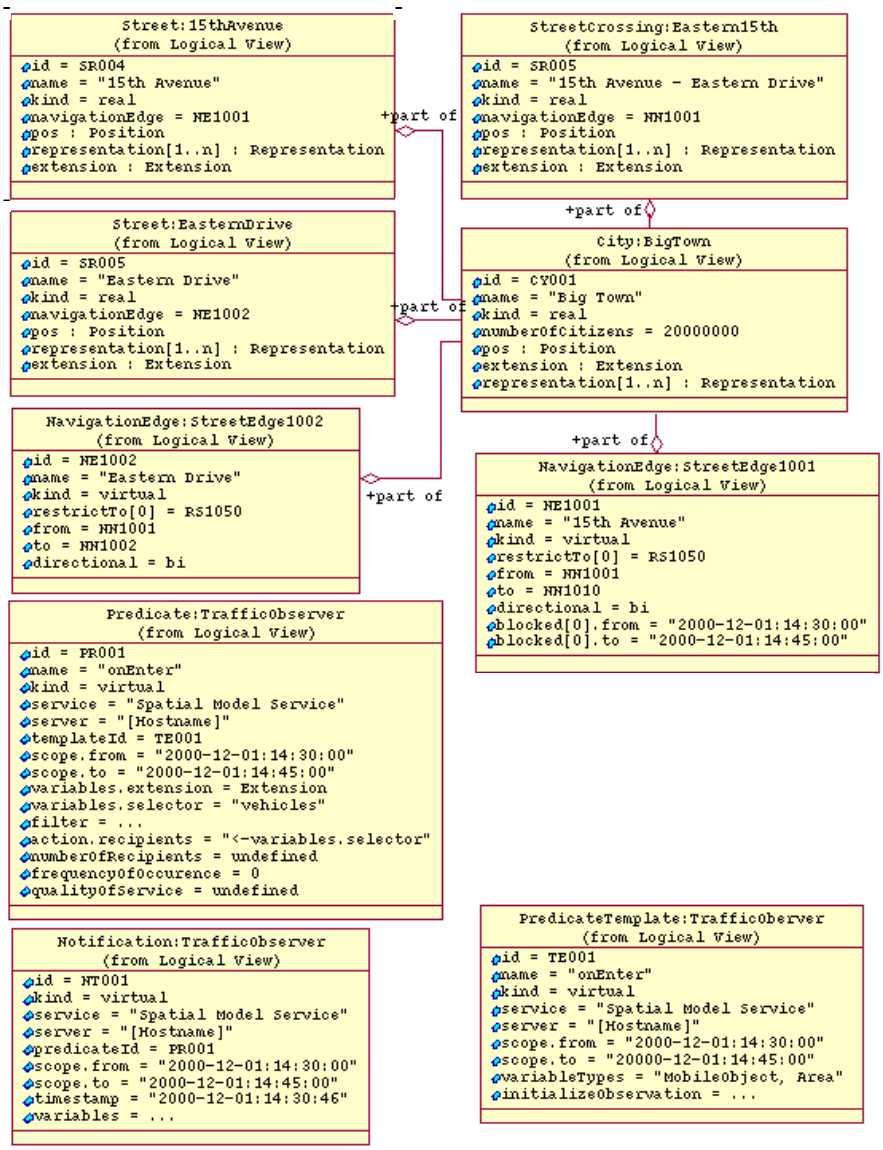


FIGURE 6-16: Consequences of Car Crash

Geocast contains information about the area that is the same as for the onEnter event. The recipients will be all people within the area. The collided cars are mobile users having a sensor that notices the car crash. The navigation edge where the accident occurred is immediately marked as blockaded as soon as the local authorities have been informed. Thus, all navigation systems are trying to avoid the edge, searching for alternative routes. Vehicles that cannot react in time are receiving the

notification of an onEnter predicate that warns them to drive carefully due to the accident in front of them.

The attributes of the onEnter predicate are also described in Figure 6-16. Variables.extension specifies the area where the onEnter is performed. Variables.selector claims that only vehicles (includes all subclasses) are going to receive a notification. Action.recipients returns the result of variables.selector, i.e. the car that entered the area. Usually recipients are predefined since any actuator of an event is registering for the event before it can be triggered. But in this special case action.recipients cannot be predefined because the receivers of the event are not the people that set it up. Events like that have to be set up by authorized institutions only. Otherwise a flood of advertising notifications would contaminate the NEXUS system. The ordinary way of receiving an event is that users are choosing from a list of offered events that are registered for the area they are currently residing in. Thus not only servers but users as well have to register for receiving events (see [BAUER 2000]).

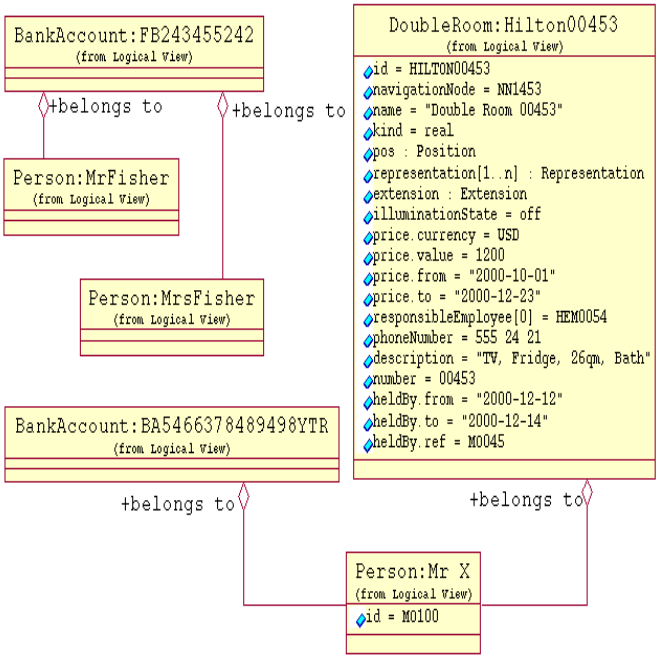


FIGURE 6-17: **Charging the Money for the Hotel**

6.2.4 ITERATIVE NAVIGATION

Whenever routes have to be calculated dynamically iterative navigation has to be used. Iterative navigation is similar to standard navigation in NEXUS. The only difference is that the route has to be calculated whenever the position of the destination or the state of the navigation network changes.

6.2.5 TRANSFERRING MONEY

As already described in section 6.2.1: “Reservation and Booking of Services” the money for goods can be charged automatically in NEXUS. Every object that can be

rented or purchased has to be assigned to an owner. The owner has to be in possession of a bank account as well as the purchaser or tenant. Then the money can be transferred from one account to another. Of course there are security aspects that have to be dealt with. Transaction have to be authorized by all participating trading partners and information about account information must only be sent encrypted.

6.2.6 VIRTUAL INFORMATION TOWERS

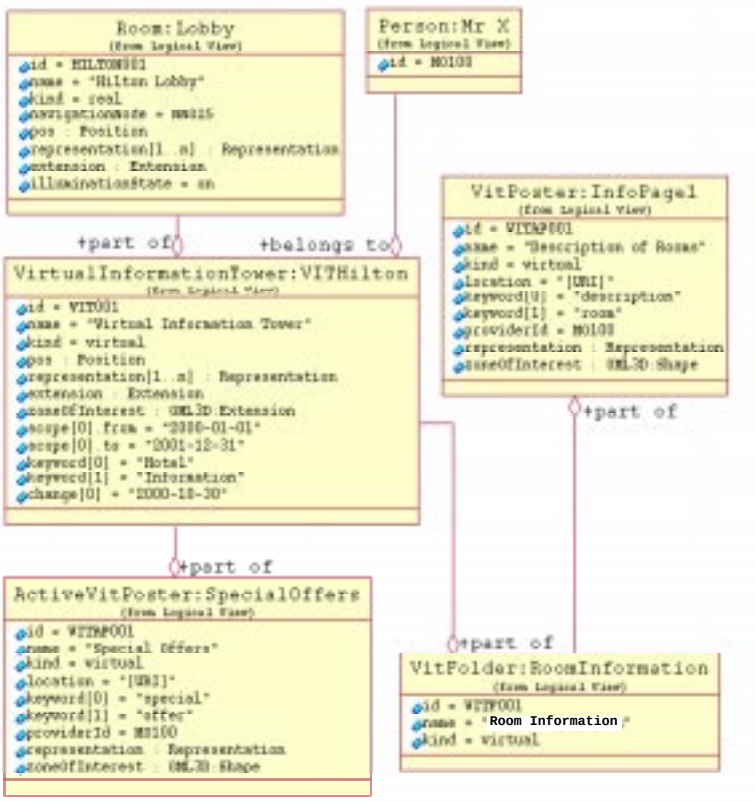


FIGURE 6-18: VIT in Hotel Lobby

This example shows the typical use and structure of a virtual information tower. The information tower is part of a room or place and contains one ActiveVitPoster and one or more VitFolders. The content of ActiveVitPosters is displayed on a navigator's status bar whereas VitPosters have to be selected by users for retrieving their content. VitPosters are part of VitFolders whereby every VitFolder is able to contain one or more VitPosters. With the exception of startpages VitPosters can only be part of VitFolders. A virtual information tower's startpage is implemented by attaching a VitPoster directly to the virtual information tower.

When Fishers are arriving at the hotel they become informed about the existence of the virtual information tower as soon as they enter the specified zoneOfInterest.

Their portable NEXUS client displays the start page, the folder hierarchy and the ActiveVitPosters. The zoneOfInterest specified for VitPosters and ActiveVitPosters is a sub-area of the virtual information tower's zoneOfInterest. In the original version [VILIS2000] the virtual information tower's zoneOfInterest was defined as a cylinder and the VitPoster's zoneOfInterest as a sector.

FIGURE 6-19: **Network Topology**

6.2.7 INFORMATION HOARDING

If we want to know whether it makes sense to employ information hoarding we have to check the network topology first. In areas with low or no bandwidth hoarded data can be used for substituting the lack of external data input. Assuming that Mr Fisher lingers at the hotel lobby that is situated in the network cell NC0D10453 (see Figure 6-19) and that he has to pass cell NC01D10454 that also has a bandwidth of 9600 bits per second. It would not be possible to transfer picture maps with such a low bandwidth for the navigation application. Thus data for generating maps on the client and a navigation application should be downloaded near the info-station in the hotel lobby (Figure 6-20). The download will be started by the client automatically as soon as it knows that Mr Fisher wants to meet Mr Andrews. A route to Mr Andrews can be roughly estimated and therefore the cells that have to be passed can be determined. After realizing that the bandwidth would not be

sufficient the application starts downloading the navigation application and the necessary data.

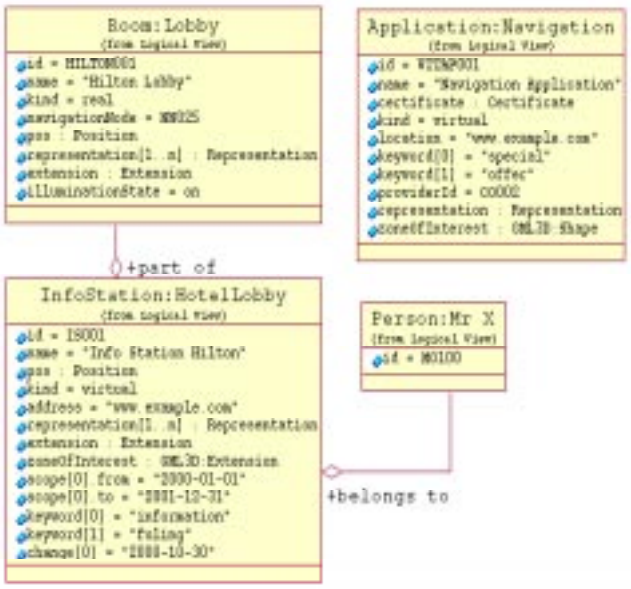


FIGURE 6-20: **Providing Downloadable Applications for Areas with small Bandwidth**

6.2.8 QUERYING THE DATABASE DEPENDING ON CURRENT POSITION

For finding a restaurant next to the current position we have to query the database for positions of restaurants that are nearest to the position of the mobile users Mr Fisher and Mr Andrews. Additionally, the business men prefer a Italian restaurant so that the search has to be performed for a set of n closest Italian restaurants. The resulting list of restaurants is displayed on the portable NEXUS client. Mr Fisher and Mr Andrews navigate through the list beholding the menu and the prices. Frank's&Benny's Italian restaurant (see Figure 6-21) attracts their attention. Before setting off for the Italian restaurant they already order the "Lasagne al Forno" for accelerating the preparation.

The example of Frank&Benny does also demonstrate the possibility of offering web pages. So people can learn more about the establishments they are interested in. The Italian Restaurant does also include the price information for the particular meals. A complete dish information includes the name and the price whereby the price information should include the amount of money and the currency. In this example MiJang's menu is only an exemplary excerpt and incomplete.

6.2.9 VIRTUAL SHOPPING

When Mrs Fisher is on her shopping expedition she enters a mall such as shown in Figure 6-22. The mall described here is not complete and shall only provide an overview of the possibilities and functions of a potential mall. But the fur coat that Mrs Fisher wants to buy works the same way as the dog that is for sale in the pet shop. Therefore this example is sufficient to show the operability of the scenario.

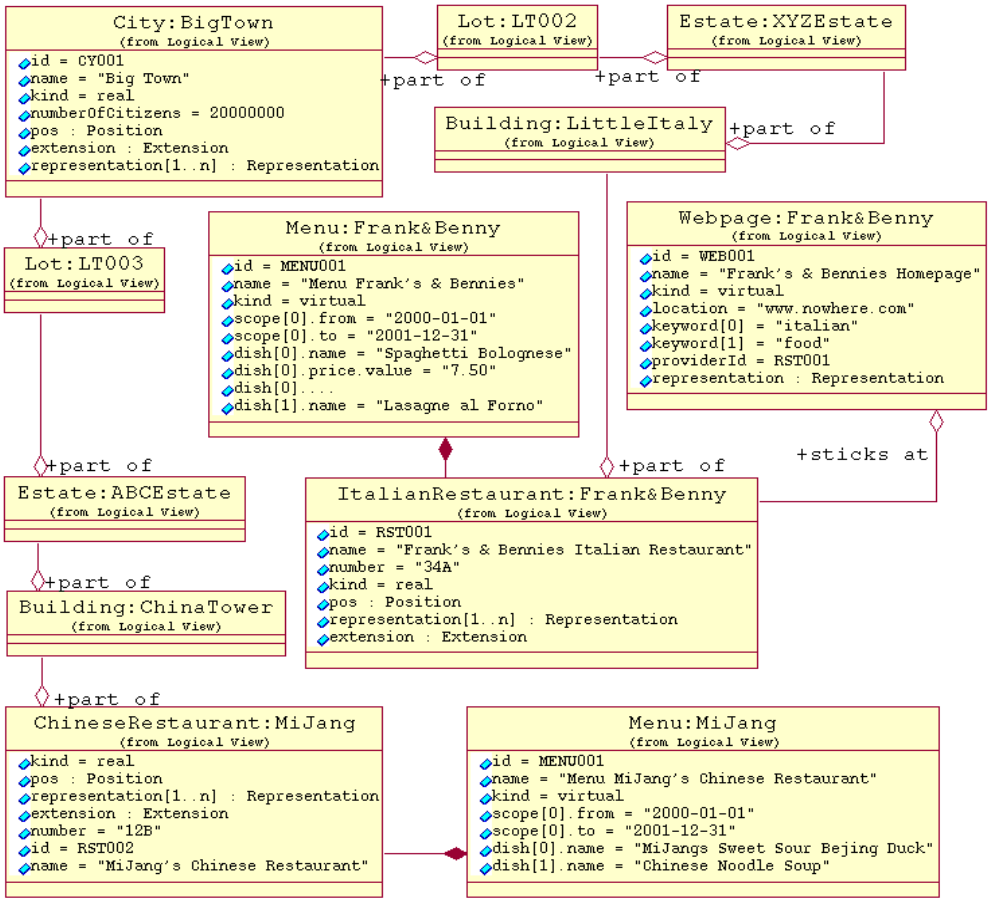


FIGURE 6-21: Looking for an Italian Restaurant in Big Town

Now we want to take a closer look at Figure 6-22. Usually the class Goods includes a product description that can be used for a short summary of the product's characteristics. The description is left to the reader's imagination and therefore not specified. The possible representations can contain photos in different perspectives or a

three-dimensional model of the object. In the event of a closed bargain the trading partners can be determined by defining the *belongsTo* attribute.

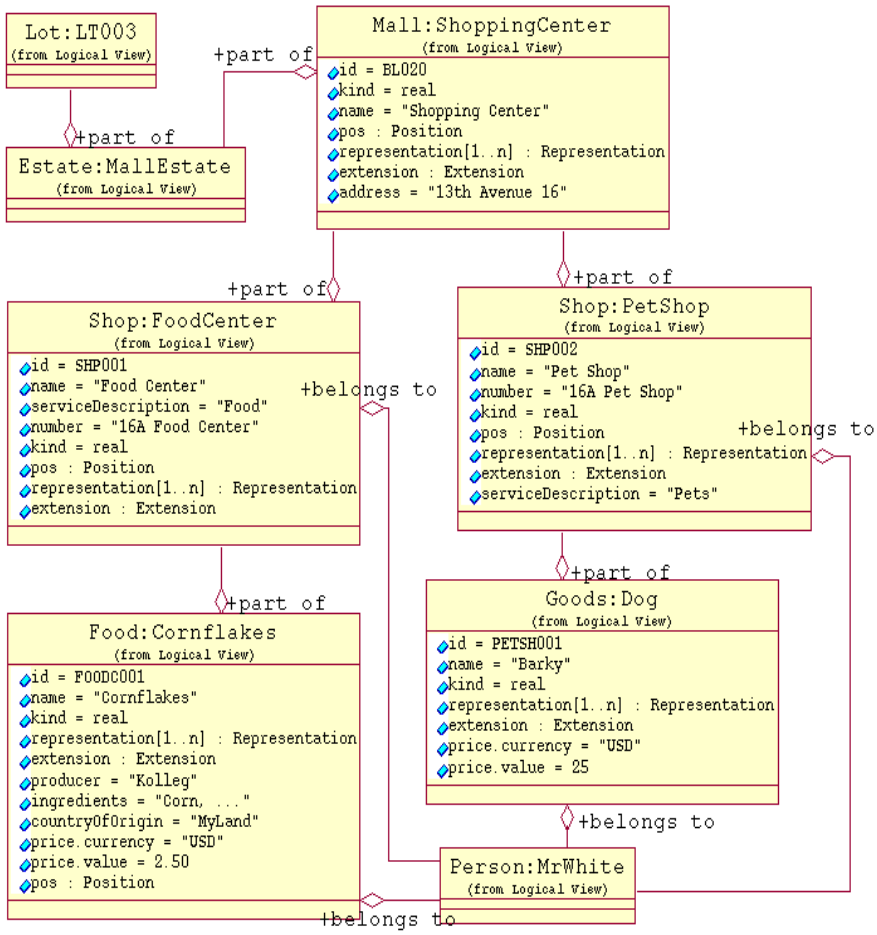


FIGURE 6-22: **Shopping in a Mall**

6.2.10 Navigation Depending on Timetables

In the last section of the scenario Fishers are on a sight seeing tour in BigTown. The information retrieval for the sights works the same way as it does for the shopping example.

More interesting is navigation and the coupling of different public transports. In contrast to the flight booking example the timetables of several public transports have to be taken into account. Since we cannot fully rely on timetables (there might be delays) we use the positions of the transports for estimating expected approach times as far as it makes sense to do so.

Arrival times for vehicles that we want to change for next, or vehicles that are less than 30 minutes away, should be estimated by using their current position. Arrival times for transports that do not fit these conditions are better computed by using their schedule data.

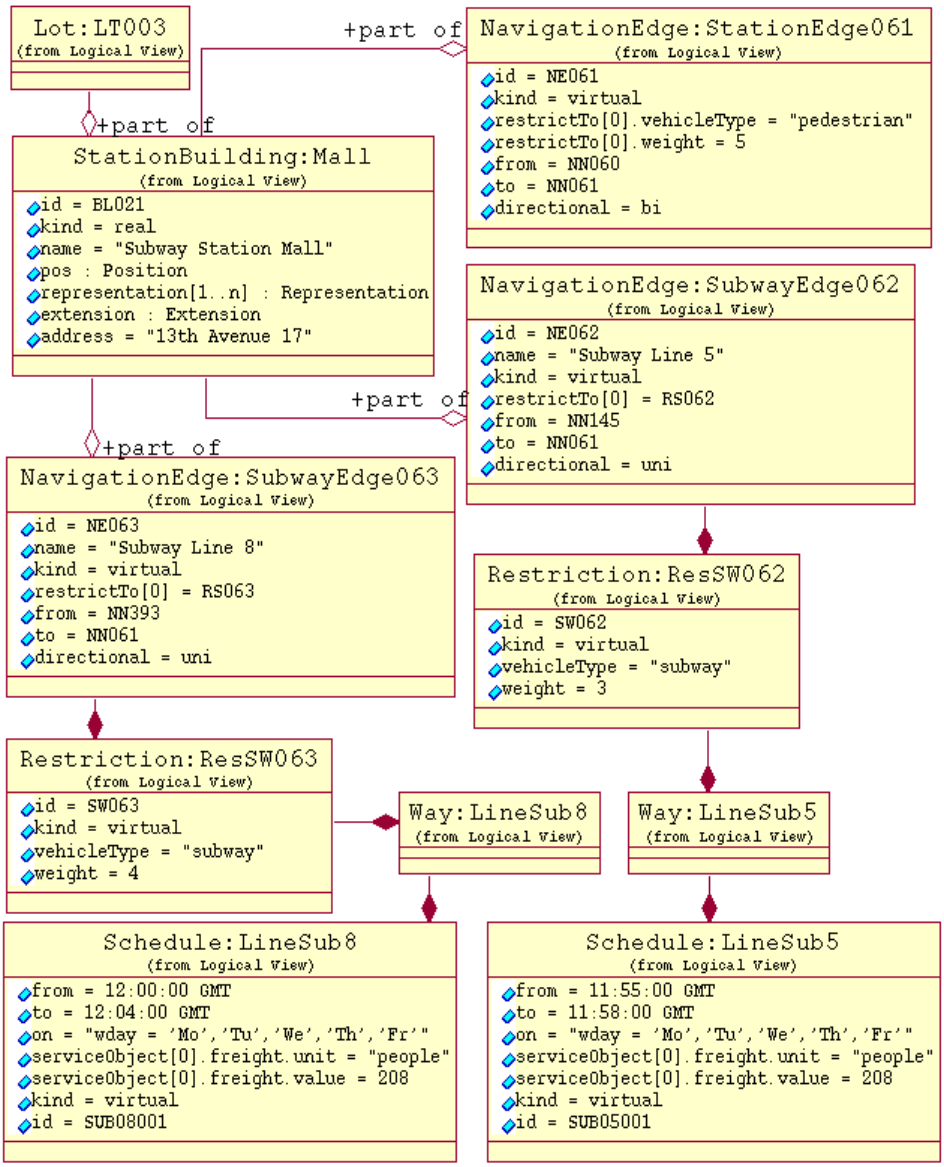


FIGURE 6-23: Navigation with Public Transportation

In Figure 6-23 there are two tubes (line number 5 and number 8) that do both pass the Station called Mall. If a passenger of line 5 wants to change for line 8 he has to wait 2 minutes provided that the timing of the subways is the same the whole day through. Figure 6-24 shows the example of Figure 6-23 on a map.

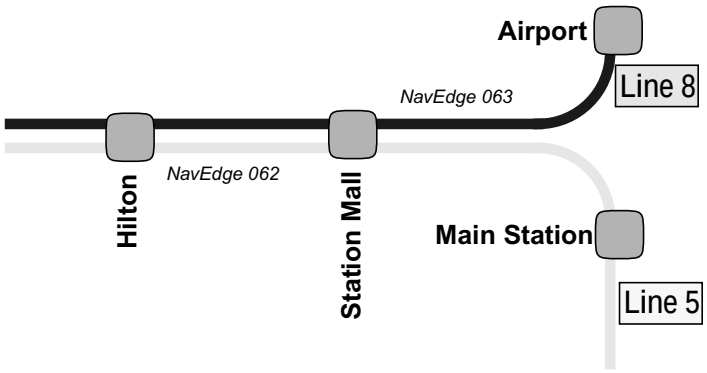


FIGURE 6-24: Navigation with Public Transportation (Map)

6.3 SUMMARY

In this chapter we gave the reader an impression of the Augmented World Standard Class Schema's functionality and proved its workability by implementing an example scenario. As a matter of course, we are only able to show the correctness for the classes appearing in this example. So there might still be some undiscovered discrepancies in the design of the standard schema. Only a prototype system will be able to prove the workability of the model or to reveal difficulties, respectively.

CHAPTER 7

CONCLUSION

Il semble que la perfection soit atteinte non quand il n'y a plus rien à ajouter, mais quand il n'y a plus rien à retrancher.

(Antoine de Saint-Exupéry, Terres des Hommes)

In this chapter we summarize the whole thesis and conclude it. We evaluate certain issues concerning distribution, navigation and implementation.

7.1 SUMMARY

In this thesis we discussed the creation of an Augmented World Model. We considered the background, the reason for establishing an Augmented World Model. Therefore we had to study the NEXUS system and the project´s intention. For enabling a uniform communication process, we defined the most important terms used in the further work, such as *Augmented World Model*, *Augmented Area Model*, *Augmented World Standard Class Schema* and *NEXUS Object*.

Next, we examined the needs on the part of the NEXUS project, including its system components and the project´s intention. We used two ways to establish requirements. Firstly, we made out conceptional requirements that are based on application-specific and consequential necessities, such as object orientation. Secondly, we examined system-driven requirements with regard to their importance to the system.

Then, we created the Augmented World Standard Class Schema corresponding to the resulting requirements. This provides the basic elements for generating Augmented Area Models.

Afterwards we delved into some interesting aspects concerning the Augmented World Model, such as distributional aspects and arising problems, namely communication via data exchange formats and the complexity of the navigation system. It is especially distributional aspects that are responsible for complicating the system in the areas of unique object identifiers, cooperation and communication among subdivided parts of the world model, as well as consistency among those parts. Coming to the data exchange formats, we studied a set of potential languages whereby we selected one, namely XML Schema, for exchanging data between system components. Concerning the navigation system, we suggested some possibilities for reducing the complexity and for simplifying the navigation process as a consequence. As a matter of course this comes along with some disadvantageous characteristics concerning the system's abilities which are reduced to a certain extent.

Finally, we provided an insight in the Augmented World Standard Class Schema´s possibilities by showing its workability in an example scenario.

7.2 CONCLUSION

Altogether, we came to a solution which suffices the system´s demands, as far as the demands are known. We were not able to prove the feasibility in a runnable system, since there are no implemented system components available at the moment. Thus, we had to rely on the fact for the example scenario to cover all classes of the Standard Class Schema and on the correctness of its implementation.

Up to now we paid no attention to the examples of [XML SCHEMA 2000] and [OGC 00-029] where structures similar to those needed in NEXUS are used. This is because the decision to use XML Schema was made too late for including any insight gained. The Geography Markup Language also contains examples very similar to the object hierarchy we need in NEXUS. Since GML objects are already partly included in this work[1], it might be possible that existing Augmented World Standard Class Schema can be slightly improved by adapting it more conformable to the Geography Markup Language.

[MOERKOTTE 1992] remained also unattended. In this paper there are some ideas concerning hierarchical object-oriented structures of machines in an industrial complex. The topic is obviously related to our work so that some interesting ideas might be enclosed. A superficial look at the paper revealed that they suggest a greater distinction between different relations, namely “part-of”, “is-a”, “facet-of” and “location-of”, which seemed to be unnecessary for our work. Nonetheless, a closer look might not be amiss.

Furthermore, we remarked problems in certain areas, especially distributional ones, whereby we suggested some possible solutions. Some future work must be performed in these areas, for these solutions are not thoroughly examined.

7.3 FUTURE WORK

The future work can be divided in several parts: one dealing with the creation of a DTD (Document Type Definition) by using XML Schema, another one concentrating on security aspects, a third one concerning the distributional aspects, especially functional dependencies, a fourth addressing the unattended papers of [MOERKOTTE 1992] and [XML SCHEMA 2000], and last, testing the functionality of the Augmented World Model in practice.

1. *Creation of a Document Type Definition:* First, we address ourselves to the specification of a DTD defining the Augmented World Standard Classes. The result of this work provides us with the Augmented World Modeling Language which can be used for implementing Augmented Area Models then.

2. *Security Issues:* Up to now, security issues have been more or less unattended. But it is important to know which people are allowed to generate objects and to delete or update objects. User authorizations and permissions must be defined. We also have to develop possibilities for encrypting information transferred via the Augmented World Modeling Language.

3. *Distributional Aspects:* We have to deal with distributional aspects, too. The most important issue in the area of distribution concerns functional dependencies. We are able to provide possibilities to avoid inconsistencies by utilizing the Event Serv-

1. GML objects form the representational part of augmented world objects.

ice. But what happens to scalability when too many dependencies must be observed? Another problem is revealed when thinking of cyclic dependencies. Strategies for avoiding infinite loops when treating a cyclic dependency must be established.

Also related to the last topic is the difficulty concerning the partOf structures. Since most hierarchical dependencies are modelled with this construct, its complexity is difficult to estimate at the moment. Additionally, data objects can be distributed among several area models corresponding to their use which results in even more partOf relations. Ultimately, only the system´s implementation is able to show its feasibility.

The navigation system behaves similar to the partOf structure. Its complexity grows by dividing the information corresponding to the vehicles travelling on it. Therefore we have to study the scalability by implementing the system.

4. *Examination of Unattended Papers:* The papers of [XML SCHEMA 2000] and [OGC 00-029] should be studied for their conformance concerning the existing Augmented World Standard Schema as suggested in this thesis. Additionally, [MOERKOTTE 1992] can be examined for its relevance regarding relations in NEXUS.

5. *Implementation of Prototype System:* As a consequence, it is difficult to predict the system´s behaviour without knowing much about the ultimate complexity. Some issues can be solved without a prototype system, such as cyclic dependencies and security. Other problems can only be examined by trying. And more difficulties are likely to appear when implementing the system.

APPENDIX A: NEXUS USE CASES

The following sequence of use-cases is based on the internal NEXUS paper [USE CASE 2000] that was created for describing some characteristic and important functionalities of the NEXUS platform. That is why the consideration of use-cases is restricted to NEXUS relevant examples but they should provide an extensive insight in the abilities of location- and spatial-aware applications in general.

USE-CASE 1: PASSIVE ENVIRONMENTAL INFORMATION

Importance: Fundamental
Specification: A list of changing topics is shown to a user who is moving through a particular environment (e.g. city, mall, or exhibition). The topics correspond to his current geographical position and his potential interests. That means that a tourist walking through Stuttgart, passing the 'Altes Schloß' will receive some historical information on demand.

USE-CASE 2: ACTIVE ENVIRONMENTAL INFORMATION

Importance: Fundamental
Specification: A user wants to be informed explicitly when passing a sports shop.

USE-CASE 3: DETERMINING A USER'S CURRENT POSITION AND HIS MOTION VECTOR

Importance: Fundamental
Specification: A user queries the current position of another particular user and optionally the speed that user is moving with and the direction that user is moving in.

USE-CASE 4: NAVIGATION

Importance: Fundamental
Specification: A user wants to know how to navigate from his current position to an arbitrarily chosen destination point. Alternatively, he might want to know the route from an arbitrary starting position to a given target position. A path from a starting to the destination point as a query result is supposed to be sufficient, for navigation itself is a very complex interaction. The application receiving that tuple should be able to visualize the route and to give 'left' or 'right' directives. Starting and target points can be static or mobile likewise.

USE-CASE 5: GEOMAIL

Importance: Fundamental
Specification: Tickets, that were originally reserved for an already sold-out opera presentation, are now available because their holders did not pick them up. Thus, the theatre wants to give all tourists lingering in the city a corresponding message.
GeoMail is also needed in the situation of an accident on a motor way that causes a traffic jam. All affected car drivers who approach the traffic jam have to be informed.

USE-CASE 6: QUERY A SERVICE IN A CERTAIN AREA

Importance: Fundamental
Specification: A user would like to receive a list of all Italian restaurants within the city. Other possible queries could be a list of all urban post offices, museums etc.

USE-CASE 7: RETRIEVING INFORMATION BY POINTING AT OBJECTS

Importance: Strongly desired
Specification: By pointing with an appropriate device (telepointer) at particular objects (e.g. buildings) a user can access information on these objects (a web site for example).

USE-CASE 8: LOCATION-AWARE MEMO

Importance: Strongly desired
Specification: A user is planning his weekly shopping tour. Therefore he writes down things he wants to buy on a list. When passing an appropriate shop or product he intended to buy, the user receives a reminder.

USE-CASE 9: REDIRECTING REQUESTS FOR COMMUNICATION TO AN APPROPRIATE TERMINAL DEVICE CLOSE TO THE USER'S CURRENT LOCATION

Importance: Strongly desired
Specification: Whenever a user tries to communicate with another user (by phone, facsimile, E-mail or whatever) the communication path is redirected to an adequate device near the user. Of course, this will only be possible if the called user wishes to be reachable. Assuming a telephone call to the office of user A while A is sitting in the cafeteria, the call of user B will be redirected to the phone placed in the cafeteria. If there was no phone in the cafe, an E-mail (possibly speech-generated) message would be created, telling A that he had a call from B. Asynchronous communication is being stored and can be retrieved by the user on any terminal device. Different setups allow for the systems features to be used without interaction.

USE-CASE 10: VIRTUAL WARNING TRIANGLES

Importance: Desired
Specification: An occurring accident on a motor way causes a virtual warning triangle to be placed a few hundred meters in front of the scene of the accident. Approaching vehicles are warned by their NEXUS capable navigation system to reduce the speed and to drive more carefully.

USE-CASE 11: VIRTUAL TICKETS

Importance: Desired
Specification: A NEXUS user buys his ticket at a virtual kiosk in front of an exhibition hall / trade fair. Passing the entrance, the system checks whether the user holds a valid ticket and informs the exhibition control otherwise. This can be repeated for subsections of the exhibition. Food and beverage he is consuming is automatically charged during his visit.

USE-CASE 12: SHOWING CONDUCTIONS

Importance: Desired
Specification: By viewing a piece of a wall using a special augmented reality display, all the conductions are shown.
This use case has implications on the precision of the sensors and data's level of detail rather than having new demands on functionality.

USE-CASE 13: CONTROL OF REAL OBJECTS VIA THEIR MODEL REPRESENTATION

Importance: Strongly desired
Specification: Changing the state of an object in the model world must result in a change of the corresponding real world object. Imagine a model railway and a computer or a hand-held as control device, this use-case is more realistic than the first impression one might get.

USE-CASE 14: CHANGING OBJECTS

Importance: Fundamental
Specification: The opposite of Use-Case 13. Changing the state of a real world object results in the change of the virtual model.

USE-CASE 15: IDENTIFYING OBJECTS BY IMAGE PROCESSING

Importance: Desired
Specification: A user transmits a picture of a real world object to a NEXUS application that tries to identify the object by matching it to objects of the augmented world model.

USE-CASE 16: EVENTS ADDRESSING DIFFERENT PEOPLE WITH DIFFERENT PURPOSE

Importance: Desired
Specification: A shopping centre wants to inform all entering customers of their today's special offers. Therefore the event service sends them a short message with an overview of the offers. In contrast to use-case 5, the carriers of the supermarket are interested in the amount of entering customers, and certain attributes their customers have. Thus an additional notification to the carriers is sent. This can be implemented by either instantiating two events or by changing the current event specification [BAUER 2000], so that a notification can be sent to someone else than the person that registered for the event.

USE-CASE 17: NAVIGATION WITHIN MOBILE OBJECTS

Importance: Fundamental
Specification: Navigation within huge mobile objects like ships or trains should be possible just like navigation within static objects (use-case 4). Thus, users moving on a ship crossing the atlantic ocean must be able to find each other by using the NEXUS navigation system. They should also be able to find the cinema or the restaurant aboard.

APPENDIX B: REFERENCES

[BAHL & PADMANABHAN 1999]

Bahl, Paramvir and Padmanabhan, Venkata N.

RADAR: An In-Building RF-based User Location and Tracking System

Technical Report MSR-TR-99-12, Microsoft Research, 1999

http://www.research.microsoft.com/~padmanab/papers/msr-tr-99-12.pdf

last visited: 10th December 2000

[BAUER 1997]

Bauer, Manfred

Vermessung und Ortung mit Satelliten: NAVSTAR-GPS und andere satellitengestützte Navigationssysteme; eine Einführung für die Praxis

Heidelberg: Wichmann, 1997

ISBN 3-87907-309-0

[BAUER 2000]

Bauer, Martin

Event Management für mobile Benutzer

Diplomarbeit Nr. 1836, Computer Science Faculty, University of Stuttgart, Germany, 2000

[COSCHURBA ET AL. 2000]

Coschurba, Peter; Kubach, Uwe and Leonhardi, Alexander

Research Issues in Developing a Platform for Spatial-Aware Applications

Proceedings of SIGOPS European Workshop, Kolding, Denmark, 2000

[DANA 2000]

Dana, Peter H., 2000

The Global Positioning System

http://www.colorado.Edu/geography/gcraft/notes/gps/gps_f.html

last visited: 10th December 2000

[DECKER 1986]

Decker, B.L.

World Geodetic System 1984

Proceedings 1986, Austin

[GML 1.0]

OpenGIS Cons.

Geography Markup Language Specification 1.0

https://feature.opengis.org/rfc11/GMLRFCV1_0.html

last visited: 10th December 2000

[GROßMANN 2000]

Großmann, Matthias

Identifiers for the Nexus-System

http://www.informatik.uni-stuttgart.de/ipvr/as/personen/grossmann/misc/NexusId.pdf

[HÄRDER & RAHM 1999]

Härder, Theo and Rahm, Erhard

Datenbanksysteme, Konzepte und Techniken der Implementierung

Springer Verlag, 1999

ISBN 3540650407

[HOHL ET AL. 1999]

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

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

Proceedings of the Fifth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'99), Seattle, Washington, USA

ACM Press, 1999, pp. 249-255, August 15-20, 1999, T. Imielinski, M. Steenstrup, (Eds.)

http://www.informatik.uni-stuttgart.de/ipvr/vs/Publications/1999-hohlEA-01.ps.gz

[KAHMEN 1997]

Kahmen, Heribert

Vermessungskunde

19., überarb. Aufl. - Berlin; New York: de Gruyter, 1997

(De-Gruyter-Lehrbuch)

ISBN 3-11-015399-8, 3-11-015400-5

[KHOSHAFIAN & ABNOUS]

Khoshafian, Setrag; Abnous, Razmik

Object Orientation: Concepts, Analysis & Design, Languages, Databases, Graphical User Interfaces, Standards

John Wiley & Sons 1995, 2nd edition

ISBN 0471078344

[LEONHARDI & KUBACH 1999]

Leonhardi, Alexander and Kubach, Uwe

An Architecture for a Universal, Distributed Location Service

Proceedings of the European Wireless '99 Conference, Munich, Germany

ITG Fachbericht, VDE Verlag, 1999, pp. 351-355

http://www.informatik.uni-stuttgart.de/ipvr/vs/Publications/1999-leonhardi-01.ps.gz

[LEONHARDI 1999]

Leonhardi, Alexander and Rothermel, Kurt

A Comparison of Protocols for Updating Location Information

Technical Report TR-2000-05, University of Stuttgart, 1999

ftp://ftp.informatik.uni-stuttgart.de/pub/library/ncstrl.ustuttgart_fi/TR-2000-05/TR-2000-05.ps.gz

last visited: 10th December 2000

[MOERKOTTE 1992]

Moerkotte, Guido and Walter, Hans-Dirk

Structuring the Distributed Object World of CIM

INCOM'92, 7th IFAC/IFIP/IFORS/IMACS/ISPE

Symposium on Information Control Problems in Manufacturing Technology, May 25-28, 1992, Toronto, Canada

[OGC 99-049]

OpenGIS Project Document 99-049: Simple Features Specification For SQL;

Revision 1.1, OGC, May 1999

http://www.opengis.org/techno/specs/99-049.pdf

last visited: 10th December 2000

[OGC 00-029]

OpenGIS Project Document 00-029: Geography Markup Language;

Version 1.0, OGC, May 2000

http://www.opengis.org/techno/specs/00-029/GML.html

last visited: 10th December 2000

[OPENINVENTOR 1994]

Silicon Graphics Inc.

http://www.sgi.com/Technology/Inventor/

last visited: 10th December 2000

[PAGE-JONES 1999]

Page-Jones, Meilir

Fundamentals of Object-Oriented Design in UML

Dorset House Publishing, New York

ISBN 0-201-69946-X

[SEDGEWICK 1995]

Sedgewick, Robert

Algorithmen

Addison Wesley, 1995

ISBN 3-89319-402-9, 3-89319-301-4

[SCHÜTZNER 1999]

Schützner, Johannes

Entwicklung einer Serverkomponente für ein räumliches Modell in Nexus

Diplomarbeit Nr. 1768, Computer Science Faculty, University of Stuttgart, Germany, 1999

[USE CASE 2000]

Nexus Research Group

Use-Cases für Nexus und mögliche Realisierungen

Nexus Research Group - Internal Paper, June, 14th, 2000

[UUID 1998]

Leach, Paul; Salz, Rich

Internet Draft: Format and Creation of UUID

Draft, February 1998, Expiration Date: August 1998

http://www.informatik.uni-stuttgart.de/ipvr/as/personen/grossmann/misc/draft-leach-uuids-guids-01.txt

last visited: 10th December 2000

[VILIS2000]

Angstmann, Karsten; Bindel, Stefan; Juchart, Frederik; Strobel, Alexander; Csallner, Christoph; Drosdol, Tobias; Ruffner, Christoph

Vilis - Administrator Guide

University of Stuttgart, 2000

[VRML 1.0]

VRML 1.0 Specification

http://www.vrml.org/fs_specifications.htm

last visited: 10th December 2000

[WANT ET AL. 1992]

Want, Roy; Hopper, Andy; Falcao, Veronica and Gibbons, Jonathan

The Active Badge Location System

ACM Transactions on Information Systems, Vol. 10, No. 1, 1992, pp. 91-102

http://www.acm.org/pubs/citations/journals/tois/1992-10-1/p91-want/

[WILLIAMS 1998]

Williams, Tad

Otherland, Volumes I-IV

Daw Books, 1998-2001

ISBN 0886777631, 0886778441, 0886778492

[XML 1998]

World Wide Web Consortium

Extensible Markup Language (XML) 1.0

http://www.w3.org/TR/1998/REC-xml-19980210

last visited: 10th December 2000

[XML SCHEMA 2000]

World Wide Web Consortium

XML Schema Part 1: Structures

http://www.w3.org/TR/xmlschema-1

last visited: 23rd December 2000

APPENDIX C:

INDEX

A

C

S

U

V

W

X

Erklärung

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

_____

Jens Meßmer