Chemnitz University of Technology

Diploma Thesis

Management Systems for Geographical Phenomena

An Analysis with Special Attention on Cadastral Surveying using Cincom® VisualWorks® Smalltalk

Christian Wolfgang Burkert
<chris@chrisburkert.de>

30th August 2005

CHEMNITZ UNIVERSITY
OF TECHNOLOGY

Computer Science Department
Management of Data Chair
Prof. Dr. Wolfgang Benn

Advisers:
Prof. Dr. Wolfgang Benn
Dipl.-Inform. Georg Heeg

Thesis Declaration

I hereby declare that this thesis is my own, unaided work and that recognition has been given to the references used. It has not been submitted for any degree or examination at any other university.

Köthen, 30th August 2005

Abstract

Cadastral Surveying - and mobile acquisition of geographic information in general - is the most direct manner of collecting details about phenomena in the real world. It is also the first step where modelling becomes a centre of interest. In the majority of cases this is based on the gathering of primarily geometric *data*. But is it enough to define a structure and some operations for spatial data, trying to model the real world? Is it enough to define schemata for the handling of spatio-temporal phenomena to gain valuable information from that?

Today most applications in Geographic Information Science are developed in an object-oriented manner and the advantage of encapsulating structure and operation into objects is obvious and has proved to model the terms of the domain best. But still most of these applications are based on data repositories, separating between structure and operation and contribute to the problems of today.

As a consequence this thesis reinvestigates the relational and semi-structured model and studies the benefits of an object-oriented Geographic Information System based on an Object Management System. The collected insights will be verified in a practical manner by collecting geographic information, using a Personal Digital Assistant, equipped with a GPS receiver.

Acknowledgements

To my fiancée Sina, for their love and support through all of the days I was writing this thesis.

Many thanks to the Management of Data Chair at Chemnitz University of Technology, especially to Prof. Dr. Wolfgang Benn who made this thesis possible and spent his time when I fired my questions at him.

I would also like to thank Georg Heeg and the people at Georg Heeg eK. His criticisms incited me to work harder every day and helped me to rethink several aspects. My colleagues Alexander Augustin and Jörg Belger always contributed worthwhile ideas, especially when searching for the right translations.

Perceptive criticisms also came from my friends Gerrit Beine, Torsten Höfler, Enrico Minack, Martin Telzer and Marco Steinbrecher who untiringly read this thesis and helped improving it.

Finally I would like to thank my parents for their support.

Contents

1 Introduction

1.1 Motivation

Almost everything that happens, happens somewhere. Knowing where something happens is critically important.

[Longley et al., 2001]

Cadastral surveying in short is about the management of land parcels, boundary stones and the boundaries in-between. Technology has simplified the work of a surveyor enormously. Accurate positioning with PDA and GPS is such an improvement.

But these improvements have to be continued. As [Longley et al., 1999b] argues, the knowledge about the domain of Geographic Information Science (GISc) has not been integrated into software, especially management systems, to a satisfactory degree.

Indeed, research on non-standard Database Systems became a centre of interest and lead to major improvements in GISc in recent years. Also the efforts of the Open Geospatial Consortium contributed to this process. Today, the Simple Features Specification is a widely approved standard for the majority of spatial applications.

But the view of an expert in the domain of GISc is much more different than the view of an software engineer, having some kind of DBS in mind. In fact, the development in the area of non-standard Database Systems do not comprise improvements to the modelling of the domain, but only to the representation of so called feature geometries and in most cases this is also limited to relational technology. Instead of modelling land parcels and boundary stones, the development concentrates on points and lines without the necessary specific aspects.

However, the fundamental drawbacks of the relational approach have been uncovered in several publications (e.g. [Heuer, 1997, Goodchild and Gopal, 1989]) and questions for alternatives are necessary. Thus, the **first aim** of this thesis will be to analyse the most popular approaches, presented in list 1, if they are sufficient to model the domain of Geographic Information Science in the context of cadastral surveying.

- The Relational Approach and PostGIS
- The Semi-Structured Approach and LORE
- The Object-Oriented Approach and GemStone/S

List 1: Approaches and Systems of Interest

In fact, todays geographic information is available in many different sources, mostly based on the relational model and proprietary file formats, and technology has to deal with that. Due to this, the **second aim** of this thesis is to find the most powerful approach to become a basis for a Geographic Information Infrastructure (GII). The resulting model must be able to reflect concepts of the other approaches without major drawbacks.

These two aims will be based on a simple setup, visualised in figure 1.

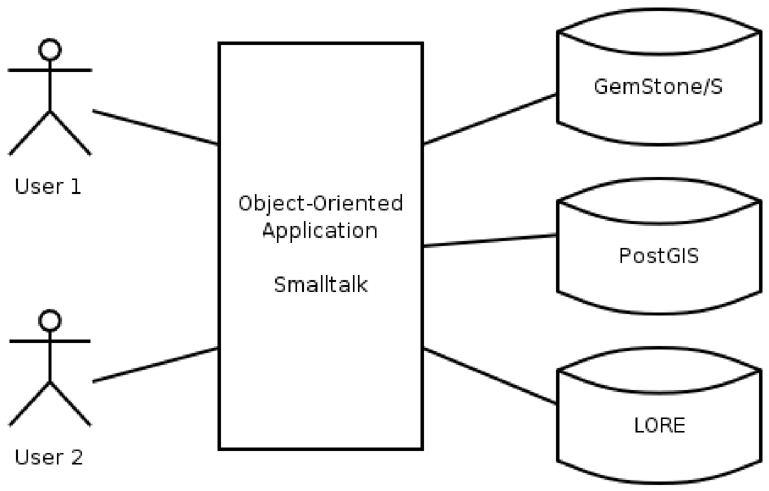


Figure 1: Basic Setup

As a methodology, this thesis will analyse the available approaches and systems if they are sufficient for the management of geographic information. As an example for further corroboration, the proposed simple conceptual model for phenomena in the real world tries to fit the language and knowledge of domain experts in cadastral surveying on one hand, and the requirements of a GII on the other. The limitation to the modelling of the domain of geographic phenomena is a volitional fact and covers only acquisition and maintenance as two tasks of a Geographic Information System (cp. list 2). Any further examination would go beyond the scope of this thesis and is assignment to the modelling of other domains (cp. [Evans, 2004]).

- Acquisition,
- Maintenance,
- Analysis and
- Presentation of geographic information.

List 2: Main tasks of a GIS

To understand the terminology of the domain, several well tried books have been used (e.g. [Imhof, 1950, 1965, Wittke, 1954, Werkmeister, 1943, Jordan and Reinhertz, 1910, Grob, 1941]) additionally to current publications (e.g. [Longley et al., 2001, Raper, 2000, DeMers, 2000, Uitermark, 2001, Torge, 2001]). The main reason for this is to avoid interferences of the emergence of computer sciences, which can be found in todays articles, leading to a prejudiced view of the issue.

To prevent expectations that do not fit with this thesis, it shall be highlighted that this work is about the way of doing things "conceptually right". There will be no detailed solution of some tricky, technical problem or, for example, performance benchmarks. It is about the general failures that have been done in the past and the alternatives to avoid those. It is about the limited approaches that make up the basis for todays technology and the alternatives to really model a domain. It is about the view of an expert and not about the view of a software engineer.

1.2 Background

This diploma thesis has been done under academical and economical circumstances and challenges the knowledge of both parties, introduced in the following.

As I have studied Applied Information Science at the Chemnitz University of Technology [http://www.tu-chemnitz.de/], doing my diploma thesis there stood to reason. Prof. Dr. Wolfgang Benn, head of the Management of Data Chair [http://dvs.informatik.tu-chemnitz.de/], always provided constructive advice and made this thesis possible.

CHEMNITZ UNIVERSITY OF TECHNOLOGY

Georg Heeg eK

After doing a placement at Georg Heeg eK [http://www.heeg.de/] in winter 04/05 at the location in Köthen, I decided to stay for the preparation of this thesis. Dipl.-Inform. Georg Heeg was an extremely enriching adviser and opened my mind for the real problems, I often passed over when digging into the details of technology a little bit too far.

A lot of this thesis is based on Smalltalk and the philosophy behind it. There are several reasons for that. For me, Smalltalk is the favoured tool for developing object-oriented applications and it has always been the first choice, compared to other object-oriented languages. Secondly, Smalltalk is the main tool at Georg Heeg eK and a lot of work, this thesis uses, has been done with Cincom® VisualWorks® Smalltalk (e.g. the GPS library). Last, but not least, Smalltalk supports the modelling of real-world problems in an easy way.

1.3 Structure

The contents at a glance are composed of the following sections.

The **Introduction**, including these sentences, provides an overview of this thesis where motivation and background are explained. Fundamental terms of the issue ensure a common language for further discussion.

Section **Context** argues the state of the art for todays technology, standards and research. Some of the major problems, this thesis addresses, become already apparent there.

In **Levels of Abstraction** the methodology for creating a model, to represent phenomena of the real world, is the main topic. The thread of the discussion is set by items like ontology, ubiquitous language, semantics, behaviour specific to the phenomenon, attributes and aggregates.

Gained insights have been verified on a prototype, debated in section **Practical Research**. PostGIS, LORE and GemStone/S have been used for further analysis of the theoretical approaches by means of these systems.

Section **Summary and Conclusions** finishes this thesis and provides a forecast for future research with basic recommendations.

Finally the **Appendix** itemises acronyms, figures, tables, listings, links and the bibliography.

1.4 Terms and Definitions

Ontology – The term ontology describes the collection of well-defined, interrelated concepts (cp. [Uitermark, 2001]) and the knowledge about their existence. While the domain ontology addresses the aspects of a specific discipline, the application and repository ontologies deal with implemented concepts. The importance of their definition shows in the multitude of available sources for geographic information.

Semantics – Semantics is the study of meaning. While ontology studies the nature of concepts, semantics is about the word that represents a concept and the mapping between different domains. It can be seen as a finite function that maps words from one domain to the other. Especially the development of a Geographic Information Infrastructure (GII) is based on semantics.

Real-World Phenomenon – A real-world phenomenon is an observable event or thing in reality. As Heisenberg has shown with his uncertainty principle, it is not possible to determine a real-world phenomenon with arbitrarily high precision. In its essence this is also true for Geographic Information Science (GISc) as it has to deal with an abstraction of the real world dominated by the perception of the observer.

Perception – The term perception comprises an abstraction of a real-world phenomenon in the external layer. While there might only be one certain phenomenon, perceptions of it likely differ according to an observer.

Entity – It is not meant to define the intentional fuzzy term entity in its entire denotation, but this thesis tries to hypothesise it as an abstraction of a distinct unit of a real-world phenomenon created by merging its different perceptions. At a minimum, this comprises semantics, behaviour, attributes and aggregates. Important to the domain of Geographic Information Sciences are spatial location and temporal existence. In contrast to more specific terms, entity will be defined as part of the conceptual layer and therefore, a concept of the domain ontology.

Feature – A feature is a more simple form than an entity. While its spatial location is explicitly given, semantics and meta information are implicit parts of the context in which the feature appears. For example the semantics can be obtained from the layer the feature is included in. Please note that the term layer, as stated in [Longley et al., 2001], is a collection of entities of the same geometric type and has nothing to do with the Three Level Architecture by ANSI/SPARC as used in section 3 on page 27.

Object – An object - in the domain of GISc - is, next to ordinary objects, the implementation of an entity in the object-oriented approach. Technically spoken it comprises the combination of a set of members in private memory and a set of operations, accessing the members in predefined ways (see also [Goldberg and Robson, 1983]). But more important are the services the object provides and the related behaviour evoked by sending messages. In contrast to the term entity, object will be defined as part of the logical layer (similar to tuple or node).

Tuple – A tuple - in the domain of GISc - is the implementation of an entity or feature in the relational approach. It represents an element of a relation and can be defined as a finite function that maps field names to certain attributes. In contrast to the term entity, tuple will be defined as part of the logical layer (similar to object or node).

Node – A node - in the domain of GISc - is the implementation of an entity or feature in the semi-structured approach. As there is XML, OEM and WebBus, the meaning of the term node will depend on the context it is used in. See section 2.6 on page 20 and section 3.5.2 on page 43 for further details. In contrast to the term entity, node will be defined as part of the logical layer (similar to tuple or object).

Data – A datum is a given statement. In computer science it is just a bulk of bits conforming to and interpreted by means of a syntactic structure. While an object encapsulates its internal aspects and provides services for transparent access, data is laid bare to act on it. There is no specific behaviour and no explicit semantics.

Schema – A schema is an abstracting model of the field of interest. With the help of a Data Definition Language (DDL) and a Data Manipulation Language (DML) it comprises structure and operations. As this is sufficient for the relational and semi-structured approaches, this thesis will show that it is not adequate for the object-oriented approach, and the domain of GISc in general, and should not be used in such a context.

Database System (DBS), Database (DB), Database Management System (DBMS) and Object Management System (OMS) – A Database System consists of a Database, which is a collection of interrelated information on a particular subject stored in a systematic way, and a Database Management System, organising and managing the DB with the help of a collection of programs. This definition is mostly incorrect for object-oriented technology, which encapsulates structure and operation in general. So this thesis uses the term Object Management System for object-oriented repositories.

Geometry – The geometry of a real-world phenomenon comprises its shape and the position inside a reference system. Typically the geometry can be represented either as a raster model or as a vector model. It is also possible to differ between explicit and implicit shape. The explicit variant keeps the shape for an entity while the implicit variant computes it with the help of nested subentities. Section 3.4.2 on page 36 discusses the aspects of geometry in detail.

Topology – As [Longley et al., 2001] points out, topology is the science and mathematics of relationships used to validate the geometry of entities. It is used for services such as neighbour finding and adjacency testing. A formal definition can be found in section 3.4.3 on page 37

2 Context

This diploma thesis takes available technology into account, which will be discussed in the following subsections.

At first, the language Smalltalk will be introduced, as it is the main tool for the development of object-oriented applications at Georg Heeg eK. Since its emergence in the early eighties of the last century, Smalltalk has influenced all major programming languages, including those, based on a totally different paradigm. It will be used for the development of the prototype in practise as well as being a representative for the analysis of the object-oriented approach.

Secondly, in section 2.2 on page 11, the terms *Data* and *Object* and their relevance will be discussed in the context of Database Systems and Object Management Systems.

Furthermore, the problem of multiple dimensions will be touched in section 2.3 on page 13.

The efforts of the Open Geospatial Consortium, especially the Simple Features Specification, have deeply influenced todays technology. Their fundamentals will be discussed in section 2.4 on page 14.

To corroborate the obtained insights in a practical manner, a prototype has been developed. Basic aspects and the available technology (e.g. GPS) will be presented in section 2.5 on page 16.

Finally, as it is a relatively new and emerging area, the basics of semi-structured data will be introduced in section 2.6 on page 20. It is assumed that the fundamentals of the relational and object-oriented approaches are known.

2.1 Cincom® VisualWorks® Smalltalk

Developed by Alan Kay and his team at the Xerox Palo Alto Research Center, Smalltalk is a pure object-oriented language (cp. [Goldberg and Robson, 1983]). It is based upon a few simple concepts and uses a straightforward syntax, making it easy to learn and even so powerful to develop complex applications. More information, than presented in this thesis, can be found at the Smalltalk Website [http://www.smalltalk.org].

As there are a lot of dialects and environments for Smalltalk, this thesis uses Cincom® VisualWorks® Smalltalk which can be found at the Cincom® Smalltalk Website [http://smalltalk.cincom.com/] and at the Cincom® Smalltalk Info Center [http://www.cincomsmalltalk.com/]. It is binary portable between many platforms, especially Windows™, Macintosh, x86 Linux and some commercial UNIX and is a direct descendant of Smalltalk-80 as described in [Goldberg and Robson, 1983]. Cincom® VisualWorks® Smalltalk is the main tool for developing applications at Georg Heeg eK.

The Object Management System (OMS) GemStone/S is based on the language "GemStone Smalltalk", which will be discussed in section 4.4 on page 67.

In general Smalltalk comprises a vocabulary of five terms (see listing 3) which will be argued in the following.

- Object
- Message
- Class
- Instance
- Method

List 3: Smalltalk Vocabulary

From a technical standpoint, an object is the combination of some private memory and a set of operations (cp. [Goldberg and Robson, 1983]). The private memory typically consists of a set of members or instance variables, which are references to other objects, and meta information. The methods are a description of how to perform a specific operation of an object. However, an object should be better defined as a service provider, responding to messages, than just as the sum of its parts.

A message is a request for an object to carry out one of its operations. The receiver of a message determines the corresponding method, which is a description of how the operation should be performed, and executes it.

A class is a description of a set of objects, behaving the same way. This includes information about the structure of the private memory (e.g. by defining instance variables) and how operations should be carried out (by implementing methods). An instance is an individual object which is described by a class.

In general, Smalltalk is a system to model a domain. Essential are the services an object provides and not the structure or operation. This view of the object-oriented approach enforces a differentiation to the term Schema, often used in database terminology. The encapsulation on one hand and the breakup on the other is one major aspect of this thesis. Thus, to describe a concept, the more generic term Model will be preferred to Schema.

2.2 Management of Information

The whole is more than the sum of its parts.

Aristoteles

In the domain of Geographic Information Science the definition of the term Database System (DBS), as a combination of Database (DB) and Database Management System (DBMS), shows one of the biggest problems that arise when dealing with phenomena of the real world. This is more obvious by opposing the terms *data* and *object*. While data is pure structure to act on, an object is much more a service provider. Since GISc represents entities primarily as objects, there is no reason to use a Database System storing data. The encapsulation on one hand and the separation on the other shows a major conflict. So in terms of object-orientation it would be better to speak of an Object Management System (OMS) - or an object-oriented repository - instead of the term Object-Oriented Database System.

Nevertheless the relational and semi-structured models do differentiate between structure and operation. So the following is valid for those two approaches but it is only partially valid for the object-oriented approach.

> A typical DBS primarily aims to provide the service of access to and the manipulation of data. This is achieved by the encapsu-

lation of the DB by a Database Management System and the use of a query language. The tasks of such a system are itemised in list 4.

- Efficient, Homogeneous and Redundancy-Free Storage of Data
- Service of Access and Manipulation
- Authentication and Authorisation of Access
- Separation of Application and DB
- Service of Backup and Restore
- Transaction Processing (ACID)
- Error Handling and Securing of Integrity
- ...

List 4: Main Tasks of a DBS

A DBS shall also support different views on the same data. The access shall be easy, central, protected and concurrently.

Most of these requirements are fulfilled by a layered architecture and rules of transformation between those layers. The use of a Data Definition Language (DDL) and a Data Manipulation Language (DML) specify the structure of data and the valid operations to be performed.

As these aspects are true for the relational and semi-structured approaches, it is not sufficient for the real world and the domain of GISc. So, to speak in terms of an Object Management System (OMS), the main task is the storage of objects and the access to the services they provide (cp. list 5).

- Efficient Storage of Objects
- Service of Sending Messages
- Authentication and Authorisation of Access
- Separation of Application and Persistent Objects
- Service of Backup and Restore
- Transaction Processing (ACID)
- Error Handling and Securing of Integrity
- ...

List 5: Main Tasks of an OMS

The differences between a DBS and an OMS should be part of the terminology and will be emphasised in this thesis.

2.3 Multidimensional GIS

Acceptance of space and time integration implies that the world can be regarded as consisting of four-dimensional 'geo-phenomena' and their inter-relations.

[Raper, 2000]

Today, most Geographic Information Systems use a limited 2D perspective for representing phenomena of the real world (cp. SFS in section 2.4.1). Historical seen this derived from the domination of maps which, printed to paper, are flat in two dimensions. [Raper, 2000] argues that every 0D, 1D, 2D and 3D representation is just a projection of a 4D phenomenon perceived in the real world, leaving further dimensions alone.

The richness of a 4D representation shows in the possibilities of virtual reality and animation. Particularly social sciences would benefit from a better visualisation of events in time.

As [Raper, 2000] summarises, there are three separating aspects of representing multidimensional entities: spatio-temporal connection, discretisation and modelling.

The connection of space and time can be either hybrid or integrated. A hybrid model separates space from time, while the integrated model deals with time as just another dimension.

The discretisation of space and time can be either continuous or discrete. While the first approach is typically based on differential equations, a discrete measurement of space and time focuses on the selection of metrics, geometries and orderings.

Finally spatio-temporal modelling can be either absolute or relative. While absolute space and time only matter in a universal reference system, relative modelling contextualises neighbours which are close in spatial and temporal distance.

For further reading [Raper, 2000] provides an excellent compendium on handling time and space. [Wachowicz, 1999] discusses special aspects of the object-oriented view on representing the real world. More information has been taken from [Worboys, 1992] and [Langran, 1992].

2.4 Open Geospatial Consortium

The Open Geospatial Consortium (OGC), founded in August 1994, is a non-commercial organisation with members from governments, industry and universities. As [Cuthbert, 1999] points out, the aim of the OGC is not to model the structure of geographic data in the computer, but to specify standards for interoperability of Geographic Information Systems at a minimum level by defining procedures of accessing and processing geographic information. For this purpose, the OGC has released several documents comprising fundamental standards[1] and concrete specifications[2].

Released documents and information about the Open Geospatial Consortium in general can be found at the OGC Website [http://www.opengeospatial.org/] and at the OGC Specifications Site [http://www.opengeospatial.org/specs/].

2.4.1 Simple Features

The Simple Features Specification (SFS), based on the publications of the Open Geospatial Consortium in [OpenGIS® Feature Geometry, OpenGIS® SFS for Corba, OpenGIS® SFS for SQL] and several other documents, introduces 2D Features, represented by points, lines, polygons and collections of geometries.

As the philosophy of the OGC indicates, the SFS does not define the structure of features but the way of how geographic information could be accessed and processed via interfaces. There are specifications for SQL, COM and CORBA.

[1] e.g. *Topic 1 - Feature Geometry* which is the same as ISO 19107

[2] e.g. the *Simple Features Specifications* for CORBA, COM and SQL

The SFS concentrates on the geometry of an entity and does not address semantics or further attributes (see also the definition of the term feature in section 1.4 on page 5). Each geometry is associated with a spatial Reference System (RS) and constitutes some basic type represented in list 6.

- `Point` and `Multipoint`
- `LineString` and `MultiLineString`
- `Polygon` and `MultiPolygon`
- `GeometryCollection`

List 6: Basic Types of Geometries

A feature implementation depends on the used technology and can be a tuple, BLOB or ADT of a geometry relation in the case of SQL, a COM-Object in case of the Component Object Model or a CORBA-Object in case of the Common Object Request Broker Architecture.

For the exchange of features, the Simple Features Specification defines two standard ways of expressing spatial objects: the Well Known Text (WKT) and Well Known Binary (WKB) representations which will be presented in the following sections.

2.4.1.1 The WKT representation

The Well Known Text (WKT) representation of features is a textual format primarily developed for exchange (cp. list 7). In the case of SQL, WKT has been integrated in many systems[3] to express the geometry of features. Thus one can insert, update or select features based on their values.

2.4.1.2 The WKB representation

The Well Known Binary (WKB) representation of features is specified as a self-describing byte stream. This comprises information about the byte order (`BO`), the type of the feature (`WKB Type`) and the geometry itself (`WKB`

[3] e.g. PostGIS as extension for PostgreSQL, Oracle® Spatial

- `POINT(0 0)`
- `LINESTRING(0 0,1 1,1 2)`
- `POLYGON((0 0,4 0,4 4,0 4,0 0),(1 1,2 1,2 2,1 2,1 1))`
- `MULTIPOINT(0 0,1 2)`
- `MULTILINESTRING((0 0,1 1,1 2),(2 3,3 2,5 4))`
- `MULTIPOLYGON(((0 0,4 0,4 4,0 0),(1 1,2 1,1 2)))`
- `GEOMETRYCOLLECTION(POINT(2 3),LINESTRING((2 3,3 4)))`

List 7: Examples for the WKT Representation in PostGIS

`Value`). While figure 2 visualises the basic structure, listing 8 shows two examples of how the types are specified.

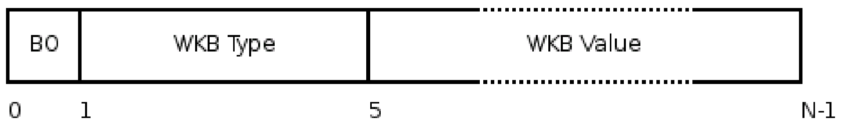


Figure 2: Basic WKB Structure

The definition of basic types, like `uint32` or `Point`, is specified by the SFS, too. In the case of types with a variable length (as in the example of `WKBLineString`), the Well Known Binary standard provides a number (e.g. `numPoints`), representing the quantity of following feature geometries, with their own type information. Therefore, it is always possible to inspect a WKB byte stream because of its self-describing basis.

2.5 Mobile Data Acquisition

While surveying and geodesy in general concentrate on the measurement of the earths surface or parts of it, cadastral surveying also comprises the acquisition of meaning and meta-information of land parcels, next to the pure geometry of entities. The terminology (cp. table 2 on page 34 for some basic examples) does not only address shape and position, but also owner, usage and housing and all the specific characteristics of phenomena. Figure

```
• WKBPoint {
   byte byteOrder;
   uint32 wkbType;
   Point point;
  }
• WKBLineString {
   byte byteOrder;
   uint32 wkbType;
   uint32 numPoints;
   Point points[numPoints];
  }
```

List 8: Example Type Definitions of the WKB representation

3 shows an example of a cadastral map with boundary stones, boundaries and land parcels.

Unfortunately todays most popular technology for geographic modelling, especially the SFS presented in section 2.4.1 on page 14, does not provide a sufficiently rich set of semantic constructs. The major need of a cadastral surveyor is a tool for mobile acquisition that reflects the terms of his expert knowledge.

In this thesis, the development of such a basic tool will, for example, integrate GPS, the modelling of the terms in table 2.4.1 and the handling of entities. The prototype will be based on the object-oriented system Cincom® VisualWorks® Smalltalk and uses the Personal Digital Assistant (PDA) MD 41600 by MEDION®. The PDA ships with a Sapphire G-Mouse (RGM-2000) GPS-receiver by RoyalTek. You can find more information on the websites of MEDION® [http://www.medion.de/] and RoyalTek [http://www.royaltek.com/].

Internal details of the prototype can be found in section 4.1 on page 55.

2.5.1 The Global Positioning System

GPS is the first system that allows accurate, direct, and inexpensive measurement of absolute position of the Earth's surface.

[Longley et al., 2001]

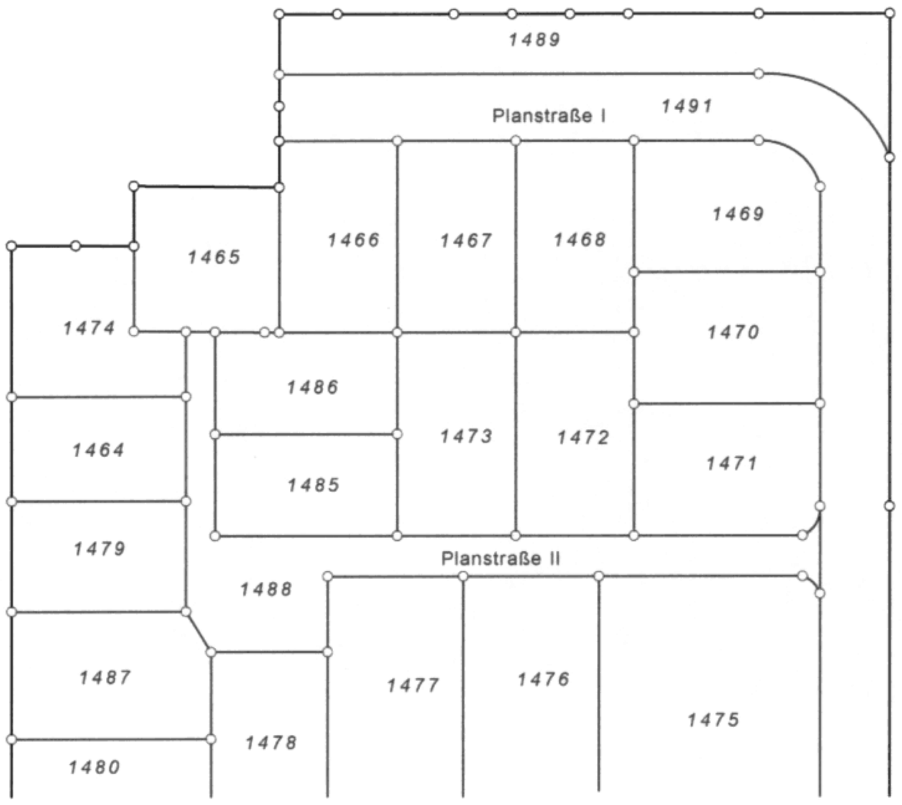


Figure 3: An Example of Land Parcels and their Boundaries

The Global Positioning System (GPS), also called NAVSTAR GPS by the US military, is a satellite navigation system for positioning services on Earth or in Earth orbit. The GPS system was designed by and is controlled by the United States Department of Defense. The Federal Radionavigation Plan [FRP 2001] defines two basic services: The encrypted Precise Positioning Service (PPS), mainly used by the US military, specifies a guaranteed accuracy of 22 meter horizontal, 27.7 meter vertical and 200 nanosecond time. In contrast, the Standard Positioning Service (SPS), used in civilian receivers, specifies a guaranteed accuracy of 100 meter horizontal, 156 meter vertical and 340 nanoseconds time. Nevertheless it is possible to achieve a much better quality with additional hardware at the receiver based on the difference of phase and DGPS. SPS is free of charge and can be used by anyone with the restriction that Selective Deniability, when enabled by the US military, will jam any civilian receiver leaving the encrypted PPS untouched.

GPS comprises three segments: space, control and user (cp. figure 4). The space segment consists of the satellites and their constellation. The control segment consists of stations on the Earth monitoring the flight paths of the satellites. Finally the user segment consists of GPS receivers decoding time signal transmissions from at least four satellites and calculating their positions by triangulation. Accuracy of this calculation mainly depends on the number of time signal transmissions from different satellites and the short-term stability of the receivers clock.

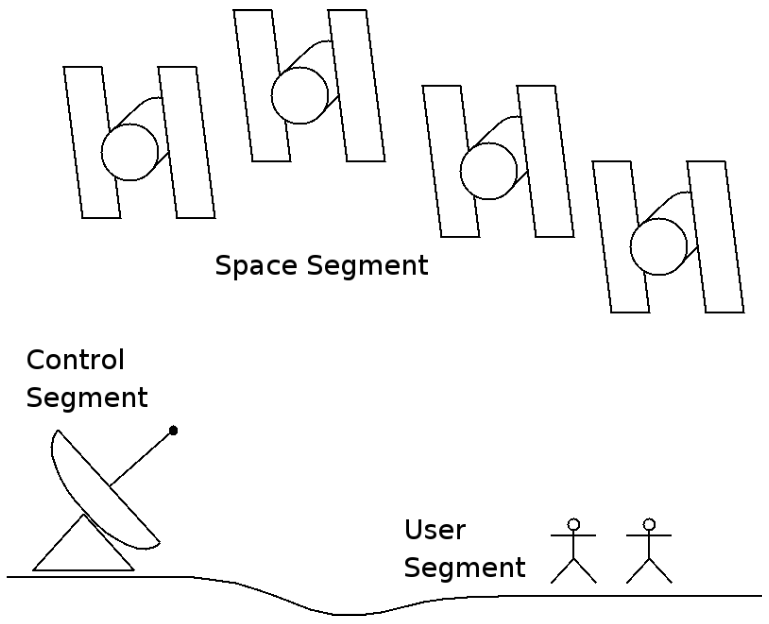


Figure 4: The segments of GPS

Although there are other systems evolving[4], GPS first clarified the importance of absolute positioning and data acquisition in Geographic Information Science (GISc). Also future receivers will likely combine time signal transmissions from GPS and Galileo to maximise accuracy.

[4] e.g. Galileo, developed by the European Union and European Space Agency

2.6 Semi-Structured Data

Research on semi-structured data is a relative new and emerging area. The basic aim is to get rid of a fixed model only perpetuating a fundamental structure of organisation. Also the data in a semi-structured repository shall be self-describing by keeping syntactical and semantical meta-information about the data.

In the following, this thesis will present three popular ways of modelling semi-structured data, which are the Extensible Markup Language (XML), the Object Exchange Model (OEM) and WebBus, a system developed by Qilin Software GmbH and the Chemnitz University of Technology

2.6.1 XML

XML combines all the inefficiency of text-based formats with most of the unreadability of binary formats.

Oren Tirosh

The Extensible Markup Language (XML) derived from and is a subset of SGML[5] and is available as version 1.1 from the World Wide Web Consortium (W3C) as to the time, this thesis was written (cp. [XML 1.1]). As it is based on Unicode, XML is portable between many platforms.

XML is a textual format for the definition of documents, structured as a tree. This comprises markup in the form of entities - in the following called nodes - and character data. The latter is unstructured text. A node consists of a tag, a sequence of attributes and a sequence of subnodes or character data. Each tag has to be closed in a way that it is either a single "empty-element-tag" or a "start-tag" closed by an "end-tag".

The ordered sequence of attributes consists of attribute/value pairs and may be empty. The content between "start-tag" and "end-tag" is made up by an ordered mixture of subnodes or character data and may also be empty.

[5] SGML has been standardised by the International Organization for Standardization (ISO) (cp. [ISO 8879:1986])

Both sequences enable XML to differentiate between attributes and aggregation or nesting of subnodes.

XML is not completely self-describing. While other approaches (like OEM and WebBus) specify a node with a semantical and a syntactical part (name and type), XML only specifies the tag name. It is not possible to make the type of attributes and nodes explicit without a schema and, thus, abrogate the semi-structured nature.

Because of the tree-structure of documents and the absence of node identifier, XML is not able to directly model many-to-many relationships. This is only possible by defining attributes like ID and IDREF to identify nodes and refer to them in other places manually.

XML provides possibilities to define a schema and, thereby, abrogate the semi-structured basis. One of them is the Document Type Definition (DTD). The following example shows a XML document with only the root node of the tree (greeting) and meta information in form of the XML declaration and a reference to the appropriate DTD.

```
<?xml version="1.1"?>
<!DOCTYPE greeting SYSTEM "hello.dtd">
<greeting>Hello, world!</greeting>
```

While XML best suites a document-centric view, the use of it in GISc is more data-centric. One problem of this is that the user has to select the documents, the query should be executed on.

Another disadvantage is that there are only a few native XML databases, while the majority are XML-enabled databases, meaning that those are able to import and export XML, but not to use XML inside, based on it as a model ([Bourret, 2004]). Also XML-enabled databases usually require a schema and thus abrogate the semi-structured nature.

One big problem of XML is the handling of so called "null data". Basically there are two approaches to this. One is to interpret a missing, but expected tag as null data. Another is to define a schema with a special tag representing null data.

Finally the performance of XML-based systems is in principle low. This is due to the fact that the sequential parsing of text and the conversion into another model is always necessary, before any information can be gained. Also a non-textual model, like the Document Object Model (DOM), does not guarantee fast access. The tree, a DOM represents, may not reflect the current view so that additional navigation and search, by means of ID and IDREF attributes, becomes necessary.

When selecting a native XML-Database, the multitude of available standards does not help much. When using schemata there is DTD, XML Schema, RELAX NG and others. The query language can be XQuery, XPath, XQL, XML-QL, QUILT etc. For programming there is SAX and DOM.

A management system for XML and semi-structured data is the Lightweight Object REpository (LORE) (cp. [Quass et al., 1996, McHugh et al., 1997, Goldman et al., 1996, 1999, 2000]). Based on OEM, it has been developed at Stanford University in the mid nineties and was adapted to XML later.

To summarise the above, XML is a standard for textual documents and has not been created as a data model for DB-design. Nevertheless, such systems have been developed and are available as XML-native or -enabled repositories. An analysis of these can be found in section 3.5.2 on page 43 and section 4.3 on page 64.

2.6.2 OEM

The Object Exchange Model (OEM), a pure data model for use in database design, is typically adumbrated by a labelled, directed graph with a root node (cp. [Stefanakis, 2002, Goldman et al., 1996]). A node has an Identifier (ID), a label, a type and a value. The ID shall uniquely identify the node in the domain of interest. A label contains semantic information about the node while the type keeps track of syntactic information. Both, label and type, enable OEM to become self-describing.

An atomic node has a value of some basic type (like integer or string) while a complex node has an unordered set of subnodes as the value. Thus, atomic nodes are leaf nodes and complex nodes are not.

Because of its graph structure, OEM can model one-to-one, one-to-many and many-to-many relations. But it is not able to differ between attributes and nested subnodes. Also it does not support high level features like classes, methods and inheritance as those, which can be found in the object-oriented approach.

One management system for OEM and semi-structured data is the Lightweight Object REpository (LORE) (cp. [Quass et al., 1996, McHugh et al., 1997, Goldman et al., 1996, 1999, 2000]). Based on OEM, it has been developed at Stanford University in the mid nineties and was adapted to XML later.

2.6.3 WebBus

WebBus, also a pure data model, is represented by a labelled graph with a root node (cp. [Fiedler, 2002]). It can be seen as an advancement of OEM and addresses some of its shortcomings.

A node in WebBus has an Identifier (ID), a name, a type and domains. Unlike in OEM, there is no distinction between atomic and complex nodes so that all nodes are equal. The ID is only visible to the system and can not be accessed by the user. For reference the user can identify the node only by name and type. The name contains semantic information about the node while the type keeps track of syntactic information. Unfortunately this information may not be unique. Both, name and type, enable WebBus to become self-describing.

Dynamic attributes, comparable with atomic nodes in OEM, are grouped into disjunctive domains and consist of a name and a type. Additionally each domain keeps track of an ordered list of edges to a nested subnode. An edge has a name, a type, a start node and an end node, but the direction of the edge is not important for the traversal of the graph. The concept of domains allows more flexibility for grouping attributes and subnodes, but also complicates things.

Like OEM, WebBus can model one-to-one, one-to-many and many-to-many relationships. In contrast, WebBus is able to model attributes, differentiated from nested subnodes.

2.6.4 Summary

To summarise the last three sections this thesis will oppose XML, OEM and WebBus (cp. [Fiedler, 2002]). The used symbols in table 1 on page 25 have the following meanings:

✓	OK, the item is supported sufficiently.
–	No, the item is unsupported.

While XML is a textual representation of semi-structured data, OEM and WebBus are pure data models allowing better database design. This shows in the inability of XML to direct model many-to-many relations by using a tree instead of a graph.

Modelling node-specific behaviour is impossible for all three semi-structured approaches and is one of the major drawbacks in further discussions.

XML does not support identity as it is realised in OEM. Since the identifier, WebBus uses, is a system ID not visible to the user, the ability to access WebBus nodes by name and type is also not satisfactory.

The distinction between attributes and aggregation is not made in OEM. Aggregation or nesting of nodes is implemented in all three approaches while attributes are not. XML has attributes by providing attribute/value pairs. Also WebBus can define named attributes. In OEM, one has to nest atomic nodes in the node of interest to pretend attributes, but it is not possible to differentiate between those and the nested ones.

Edges in XML have no label and nested nodes can only be accessed by providing the number of occurrence in the list of subnodes. Thus order becomes important for XML. WebBus also keeps track of the order of nested nodes, but does not depend on.

Finally the dissemination of XML, OEM and WebBus is pretty different. While WebBus is quite fameless, OEM has established in the academic society. Despite of its shortcomings, XML has become an important technology and is supported all over the world. There are a lot of tools and libraries

	XML	OEM	WebBus
Data Model	–	✓	✓
Structure	tree	graph	graph
many-to-many	–	✓	✓
Behaviour	–	–	–
Identity	–	✓	✓
Attributes	✓	–	✓
Aggregates	✓	✓	✓
Labelled Nodes	✓	✓	✓
Labelled Edges	–	✓	✓
Order	✓	–	✓
Dissemination	high	middle	low

Table 1: Opposing XML, OEM and WebBus

available and the Open Geospatial Consortium (OGC) has made progress in this area by developing the Geography Markup Language (GML).

2.7 Summary

The last sections have covered todays state of the art and discussed the available technology and other aspects in the context of Geographic Information Science, on which this thesis is based on. Part of this adumbration were basics of management systems, standards like the SFS and GPS, as well as multidimensional aspects. There was also a short introduction to the semi-structured approach.

As a forecast it can be said that there are a lot of problems. For example SFS can only handle 2D-information while GPS provides 4D-data. Also the definition of a Database System is not sufficient for objects in Smalltalk (and the object-oriented approach in general) because of the encapsulation on one hand and the breakup on the other. Furthermore modelling a domain requires the modelling of behaviour which is not supported satisfactorily by the relational and semi-structured approach (cp. [Longley et al., 1999b]).

Due to these problems, which will be discussed later, it is important to address the central area of interest: the domain of Geographic Information Science. The following sections will cover a methodology of modelling phenomena by abstraction. This will be based on ontologies to determine the semantics of the domain and to ensure a common language. The problems of different abstractions, like in the relational, semi-structured and object-oriented approaches, will be discussed in the appropriate sections.

3 Levels of Abstraction

The world is infinitely complex, but computer systems are finite. Representation is all about the choices that are made in capturing knowledge about the world.

[Longley et al., 2001]

To gain information from a Geographic Information System one needs a formalisation of the domain of interest. As [Longley et al., 2001] points out it is not possible to store all information of a real-world phenomenon, because of its infinite complexity. Often the information is also fuzzy[6] or not accurate which leads to further problems.

This thesis is based on an extended variant of the Three Level Architecture by ANSI/SPARC (see figure 5 and [Laurini and Thompson, 1992, Burrough and McDonnel, 1998]). The attempt is to represent a real-world phenomenon in a simplified manner by modelling entities in a management system. This requires a procedure of abstraction to merge different perceptions of the real world into a conceptual model. The creation of the logical model has to be based on the conceptual model and according to the selected system. The lowest layer is responsible for storing the bits based on an internal model.

While the creation of a conceptual model is a cognitive exercise (cp. [Burrough and McDonnel, 1998]), the emergence of the lower models should be processed on a well defined basis. The relational approach has such a basis which is implemented by rules of transformation in a Relational Database System, but the semi-structured and object-oriented approaches still lack a standardised foundation (cp. [Heuer, 1997]). Due to this fact this thesis will take some restrictions where necessary and apply best practise patterns.

It should also be said that the process of abstraction is not limited to one direction. As the knowledge about the domain deepens, the gained insight enriches all layers. A model is only relevant if it maps to the others. For example the implemented logical model must reflect the ideas of the conceptual layer and vice versa.

[6] e.g. where is the border of a desert or a high pressure area?

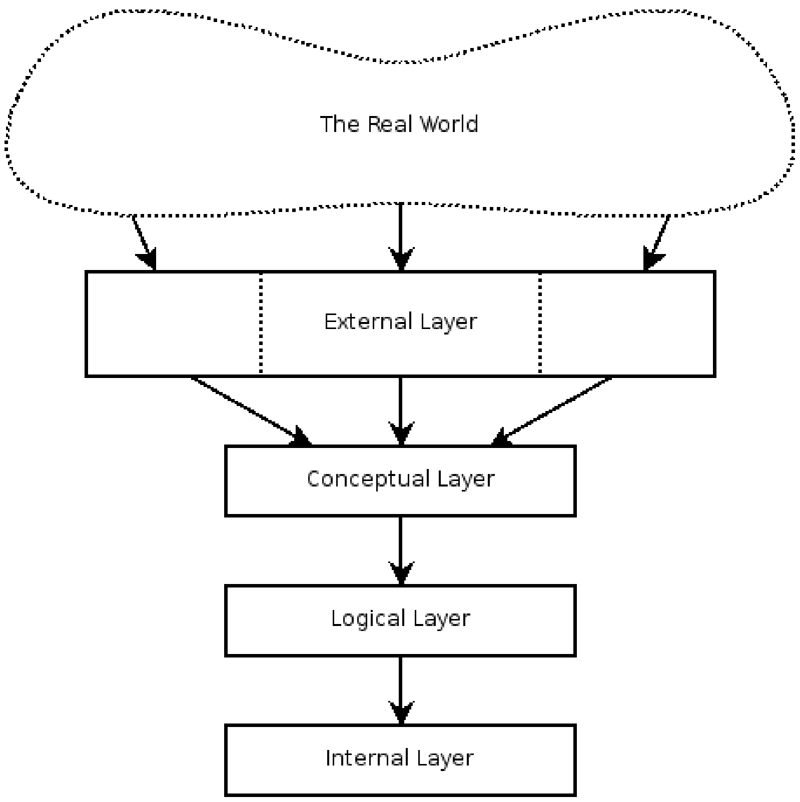


Figure 5: Level Architecture

3.1 Ontology

Making semantics explicit is a communication problem. Any successful communication requires a language that builds on a core of shared concepts. An ontology is such a collection of shared concepts.

[Uitermark, 2001]

When it comes to communicate a specific issue among others, a common language is needed. [Evans, 2004] argues that an ubiquitous language based on a domain ontology is essential for developing any system.

The term ontology, specified in section 1.4 on page 5, describes a collection of concepts and the best case would be to use these concepts throughout the system, leading to a homogeneous model on all levels of it.

However, there are usually three kinds of ontologies involved: domain ontology, application ontology and repository ontology. While the first reflects concepts of the domain experts, the latter describe technical terms of the application, a Database System or any other source of information. Unfortunately these ontologies are different in the majority of cases which complicates things. Having a Geographic Information Infrastructure (GII) in mind, technology has to deal with different repositories and, next to an explicit semantical mapping, transformation rules between ontologies are necessary.

In general, ontologies of domain, application and repository shall be the same. If this is not the case, there is a potential loss of information due to lacking transformation rules. Figure 6 shows one possible example where a system is built upon an ontology which does not differ between domain, application and repository and also integrates repositories based on their own ontologies. The concepts of domain and application have been kept the same and can also be found in Source A (cp. 6). Due to the same semantical mapping, there is no need for a transformation between them. In contrast to this, Source B to Z have a different ontology than application and domain and the semantical mapping may not be complete, a transformation of concepts even impossible, because of fundamental differences.

This thesis discusses the mismatch of these concepts by developing a basic ontology in the context of GISc for domain and application - and the corresponding conceptual and logical model - and tries to implement it in the models of different repositories.

As a conclusion it can be said that the emergence of a model shall be based on an ontology, reflecting the terms of the domain experts. The language to communicate aspects of the domain shall be used throughout the whole system to gain the best result. If this is not possible, rules of transformation have to be introduced to ensure the correct mapping of concepts between one ontology and the other.

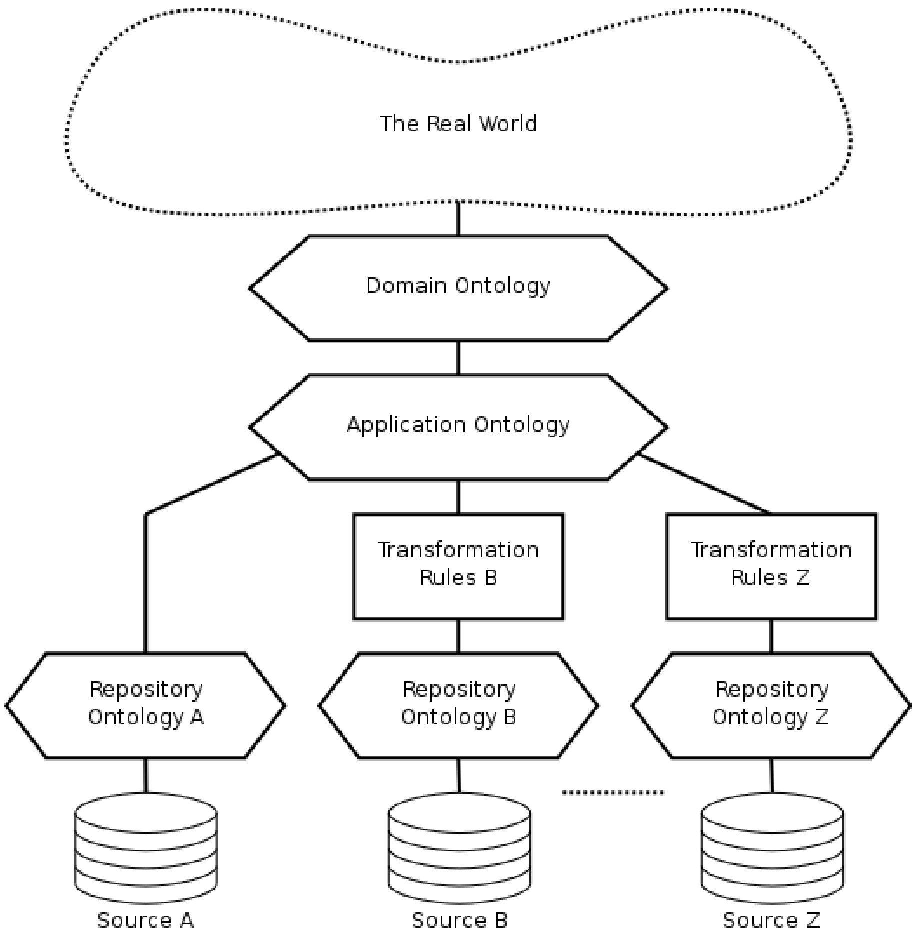


Figure 6: Ontology in a GIS

3.2 The Real World

As Heisenberg has shown with his uncertainty principle, it is impossible to determine a real-world phenomenon with arbitrarily high precision. Thus, it is also impossible to describe the real world and one could argue that we even do not know if there is a reality at all.

Todays standard of knowledge assumes the real world as an unique universe of phenomena which are perceived as observable events or things. Based on this definition an abstraction of some domain in reality is fundamentally dominated by the perception of the observer and leads to an external model, likely different from views of other people.

In the case of GISc, typical fields of application in this domain are cadastral surveying, agriculture, forestry, archaeology, navigation or social studies. The important thing is that all of them have different requirements to a Geographic Information System (GIS) and will lead to different perceptions of a real-world phenomenon among scientists.

For example, a tree can be seen as a resource including information about the age and the type of timber. Furthermore the perception may be some kind of natural habitat for animals. Or it is just a balk for vehicles routed by a navigation tool. The basic denominator of these views is only the term *tree*.

So the central aim is modelling the domain of geographic phenomena, excluding other domains. Sciences on ontology can help to understand the being of phenomena (cp. [Raper, 2000] and section 3.1 on page 28).

To give a basic insight into the terminology of geographic sciences some of the older, well tried books, e.g. [Imhof, 1950, Wittke, 1954, Grob, 1941, Werkmeister, 1943, Jordan and Reinhertz, 1910], provide an excellent view of the real world as it was, when there were no computers. This perception of reality is not impacted on the emergence of a schema or any other form of computer based models. Some examples have been itemised in list 9.

- terrain
- mountain
- ridge
- valley
- slope
- tree
- cover of soil
- building
- artefact
- ground wave

List 9: Examples of Terminology in GISc

The basic aim of a GIS should be to reflect such terms inside the software. The definition of a language containing such well-defined terms is a starting point for the modelling of phenomena of the real world and should be based on a domain ontology (cp. [Evans, 2004]). In fact there is some effort for the development of an ontology. In the Netherlands, a domain ontology for the discipline of topographic mapping, the Geo-Information Terrain Model, is under construction. Also Goodchild, Mark and Egenhofer proposed a project to determine an ontology for geographic phenomena (Ontology Project [http://www.geog.ucsb.edu/ good/275/v2kontology2.htm].

3.3 The External Layer

There is no objective view of terrain and landscape.

[Imhof, 1950]

As stated above, each observer of phenomena creates his own external model from his perception, which mostly is an incomplete and abstract subset of the real world (cp. [Laurini and Thompson, 1992]).

The emergence of a specific external model happens in mind and is influenced by the experience of the observer, his cultural background and the audience he addresses (see also [Burrough and McDonnel, 1998]). This perception influences all subsequent conclusions on the collected information.

To define a common denominator, todays academic discourse distinguishes between entities and continuous variations. Entities are for example houses, roads, cables etc. Typically they have a unique identity and a demarcated spatial comprehensiveness and position at a point in time. Continuous variations, like temperature measurements, high pressure areas or deserts, do not have an exact border or position and it is difficult to identify them at all, as they only exist because of their values.

These fundamentally different ways of abstracting the real world basically have one thing in common: both can not describe the real world completely. But as [Evans, 2004] argues that entities and value objects can be modelled similar, this thesis concentrates on distinct entities. Another reason for this decision is that in cadastral surveying, continuous variations play a minor role.

3.4 The Conceptual Layer

The synthesis of selected perceptions from the external layer is a main task and part of modelling the domain. The emergence and usage of an ubiquitous language between experts and developers plays a major role. So terms like *tree* or *terrain* should be part of the communication and the model itself (cp. [Evans, 2004]).

But that is not the case in todays technology and standards. The term *Feature*, as defined in section 1.4 on page 5, reduces the discussion on the geometry of a real-world phenomenon. Points and conglomerations of points make up the basis for features and show the inability to model reality. Of course the Simple Features Specification (SFS) highlights the word *simple*, but most of the other ambitions of the OGC are based on SFS and may be too simple at the end.

So the main pieces of the domain of geographic science are entities without any attention to a technical solution. These entities can, for example, be *trees*, *crossings*, *mountains* and *coasts*. Entities have a shape, usually called geometry, a spatial and temporal location and meta-information. But more important is that each specific entity has its own special behaviour, additional attributes and aggregated subentities.

Therefore modelling the domain of GISc is not primarily a task of implementing geometry algorithms. It is a process of understanding experts and modelling *trees*, *crossings* and *mountains*, to name a few. Of course, geometry plays a major role, but is part of its own domain.

As this thesis concentrates on cadastral surveying (cp. figure 3 on page 18), table 2 lists fundamental terms of this domain. In particular section 4 is based on these elementary concepts.

3.4.1 Entity

The primary aim of conceptual modelling is to understand the domain and to represent the things that make up the domain ontology. A boundary stone, for example, being one concept in our model, is an entity with its

Term	Explanation
Boundary Stone	A boundary stone is some kind of cairn or landmark to fixate boundaries between land parcels. It has a unique identity and a position.
Boundary	A boundary is a separating edge between land parcels, specified by at least two ordered boundary stones. In most cases a boundary will not be stored statically, but can be obtained from an ordered collection of boundary stones.
Land Parcel	A land parcel is an expanse, localised by an ordered set of boundary stones representing its closed boundary.

Table 2: Terms in Cadastral Surveying

own characteristics. It is part of the boundaries of the neighboured land parcels and keeps track of its position. A boundary stone can be identified uniquely and will be managed by a certain administrative authority.

So the conceptual model shall include some entity, representing a boundary stone, which models its characteristics. However, todays technology is more about some attributes of an entity, than about the entity itself. Mostly only the position of a boundary stone will be stored and its real meaning can only be recognised implicitly by the layer, or the database table it is included in.

Thus, the definition of what is the thing and what are the properties, is essential. The obvious view, to specify the boundary stone as the thing and everything else as its attributes, is not common in todays technology. For example the Simple Features Specification emphasises the geometry of a feature, while its meaning will be left as an attribute among others. It does not surprise that most experts of the domain and experts in Software Engineering do not agree with this definition (e.g. [Evans, 2004, Longley et al., 1999b]). It is simply not sufficient to model an entity.

Therefore, this thesis is based on the fact that an entity comprises explicit semantics, rich behaviour, complex attributes and aggregated subentities at a minimum. Unfortunately, todays Geographic Information Systems only concentrate on attributes and aggregates and do not model semantics and behaviour in a sufficient way, not to mention certain characteristics. The following discusses the importance of all four aspects.

Semantics, introduced in section 1.4 on page 5, help to understand the meaning of a thing. It can be seen as a finite function that maps concepts between different domains. It is also needed that semantics are explicit (e.g. it must be visible if a relationship is kind of a partonomy - `HasA/PartOf`- or taxonomy - `IsA`). In our case, terms of the domain ontology must semantically map to terms from the application and repository ontology.

Attributes comprise meta-information about the entity in a reflexive way, e.g. position and shape. A land parcel will have an owner, a boundary stone an administrative authority.

Subentities enable a GIS to refine its entities and to provide a deeper level of detail. For example, a boundary is made up of boundary stones.

The **Behaviour** of an entity is about the services it provides. For example, a boundary stone can provide information about the adjacent land parcels without knowing its location from the user perspective. Behaviour also enables a GIS to hide implementation details (cp. section 3.4.2 on page 36) and to become more flexible. These are technical and conceptual reasons for behaviour.

One might argue that behaviour is not part of a phenomenon, discussed in the following. From its definition, behaviour is the transition from one state to another. It is not important if there is an active or passive aspect, or if there is some form of life. Thus, behaviour is part of entities.

A simple example shall deepen this. There are two entities. The first one is semantically a river, filled with water. One attribute is its stream velocity. The second is some flotsam, e.g. a log. If there would be no behaviour, the log would remain at the same position without being made leeway. As this kind of progress is a concept of a river, there has to be the possibility to technically model it inside the repository.

3.4.2 Geometry

Geometry is one of the most important aspects of modelling geographical information. It comprises the position and the shape of an entity. In general, every geometry is defined relatively to a reference system, which defines its own mathematical space including multiple dimensions.

Depending on how multidimensional information shall be handled (cp. section 2.3), a position may represent a spatial location (3D) or even a spatio-temporal location (4D). Due to the complexity of multidimensional information handling and the need for additional research (cp. [Raper, 2000]), the following discussion will only address a 3D-representation.

Another reason for this decision is the fact that an integrated approach of space and time would conceptually limit the temporal aspects to the spatial attributes of an entity. This would debar other attributes of the entity from their own temporal progression. For example, a land parcel is to be sold to another owner without a change in shape or position. An integrated approach of space and time would imply a change of the geometry, while the hybrid approach would imply a change of the entity. The question is if the temporal aspect is only an attribute of the geometry or an attribute of the entity. However, a detailed debate about it would go beyond the scope of this thesis.

There are two different approaches to the modelling of geometry: vector and raster representation. While pixel (or voxel) make up the raster model, an element of a vector space is defined by a magnitude and a direction. Both approaches can be used to represent a geometry. The following paragraphs discuss these approaches for the shape and position of an entity.

Shape is usually modelled by some basic types: point, line, surface, space and their combinations in a geometry collection. Like it is the case for the position of an entity, it is totally unimportant if these will be represented as raster or vector models. The use of polymorphism for different types of shapes, which provide the same services, enables a GIS to become a Geographic Information Infrastructure and to integrate different representations. Nevertheless, some possible approaches for modelling shape are itemised in list 10.

- vectors (points, lines)
- Triangular Irregular Network (TIN) (surface, space)
- Array of Pixel/Voxel (points, lines, surface, space)
- Quadtree (points, lines, surface)
- Octree (points, lines, surface, space)
- . . .

List 10: Different Approaches for Modelling Shape

The **Position** of an entity is usually defined by a vector. But it is also possible to use a raster model, like the Octree, with only one voxel. A GIS should support both and it should be possible to compare two positions, even if they use a differing internal model.

Not knowing about the internal representation of a position or shape is one of the most important aims of a GII. As mentioned earlier, this requires the possibility to model behaviour for the a geometry and plays a key role in this thesis.

By the use of behaviour, the geometry of an entity must not be stored statically. It is also possible to obtain it as the intersection of geometries of aggregated subentities.

Another advantage of behaviour is the ability to transform a complex geometry in a less detailed one, depending on the used scale. Thus zooming could be easily solved. For example, a town shall be drawn on a map. Depending on the scale and the size of the town, the geometry could render itself as point or as surface.

3.4.3 Topology

Topology, also called Analysis Situs, is the study of space. It comprises the science and mathematics of relationships to validate the adjacency of entities. These relationships form some kind of environment.

Mathematically defined a topology is a collection T of open subsets of a basic set X where

- the empty set and X is part of T,
- the union of any collection of sets in T is also in T,
- the intersection of any pair of sets in T is also in T.

In Geographic Information Science, topology is often confused with aggregated entities. This mistake shall be clarified here. An aggregate does not necessarily imply any geometrical nearness or adjacency at all. It is possible that nested entities are adjacent (e.g. the ordered collection of boundary stones, which make up a closed boundary of a land parcel), but it does not have to be.

Therefore, topology can be used to test entities if they are adjacent to each other, in its own special meaning. For example two boundary stones can be adjacent if they share the same boundary and no other stone is in-between. In this example, the topological space is made up of partially ordered boundary stones. In another example two entities can be adjacent if their shape is tangent to each other (the topological space is $\mathbb{R}^3$).

3.4.4 Summary

As stated in the sections above, there are five main requirements to a logical model, to sufficiently represent the concept of an entity (cp. list 11).

- Ability to model Semantics
- Ability to model Behaviour
- Ability to model Attributes
- Ability to model Aggregates
- Ability to model any other important Trait of an Entity

List 11: Basic Requirements on a Logical Model

Explicit semantics provide an obvious mapping between the terms of different ontologies. Behaviour makes the handling of different representations possible. Attributes are fundamental to store any useful information about an entity and aggregates enable a GIS to improve the level of detail in terms of partonomy.

More important is the support for domain modelling in general. Instead of analysing all requirements in detail - which is not possible - a supple and flexible model allows the melt of ontologies of domain, application and repository (cp. [Evans, 2004]). Its a difference in the way of addressing the problem.

Nevertheless, there are a lot more, mostly technical requirements (e.g. many-to-many, performance), which will be discussed in the appropriate paragraphs of the following sections.

3.5 The Logical Layer

With deeper knowledge of the domain and the ontology of its terms, the conceptual model becomes a basis for the logical model (cp. [Burrough and McDonnel, 1998, DeMers, 2000]). The selection of a technical solution for the physical management of information is the first step where the computer comes to interest.

There are two seemingly contradictory requirements formulated by experts of Software Engineering and experts of Database Systems. On one hand, the conceptual and logical models shall reflect and map to each other. [Evans, 2004] argues that this is essential to keep the models relevant. On the other hand logical independence of information, meaning that changes to the logical model do not affect the conceptual model, shall be given, but is hard to achieve. The truth might be in-between. Code from the logical model only gets meaning if it reflects and maps to the conceptual model. But this does not necessarily eliminate flexibility even if changes to one model might affect the other.

To gain logical independence of information, the relational approach uses rules of transformation between adjacent layers. The object-oriented approach offers a much more flexible solution to this. As each object manages itself, local changes to the class do not affect the whole as long as the interface is kept the same.

A better requirement than independence between layers, would be to speak of flexibility between layers. Thus relevance of each model does not get lost and local changes are possible, too.

There are a lot of approaches at the logical layer and it would go beyond the scope of this thesis to debate them all. List 12 itemises five of the most popular ones discussed in the following.

- Hierarchical Approach
- Network Approach
- Relational Approach (section 3.5.1 page 41)
- Semi-Structured Approach (section 3.5.2 page 43)
- Object-Oriented Approach (section 3.5.3 page 45)

List 12: Approaches to Logical Modelling

The **Hierarchical Approach** (cp. [DeMers, 2000]), organised as a tree, is one of the oldest models and stores pure data, which can be searched by navigating the tree. The decision for traversing a specific subtree is based on one criterion and is thus, limited to the hierarchy itself. Searching via other criteria is not possible without traversing all nodes. Due to this, refactoring the structure is hard. Also the linkage of meta-information is a problem because of the fixed hierarchy. Nevertheless, relationships in the form of many-to-many are not possible which leads to redundancy.

The **Network Approach** (cp. [DeMers, 2000, Laurini and Thompson, 1992] addresses some of the disadvantages of the hierarchical approach and introduces many-to-many relationships. In short, the approach is less rigid, but there are still major drawbacks, making it hard to model entities (e.g. the lack of a rich set of semantic constructs and missing behaviour).

Both archaic approaches do not fit the requirements on modelling the cadastral domain and will not be included in further debates. The following will discuss the relational, semi-structured and object-oriented approaches in detail.

3.5.1 The Relational Approach

Relational databases dominate GIS today, as they do in many other business areas.

[Longley et al., 2001]

Relational technology is based on the separation of a structural and an operational part (cp. [Heuer, 1997]) and this is also one of the major disadvantages of the approach. The concept of sets and the relational algebra is, from a mathematical standpoint, similar to functional languages and adequate for solving mathematical tasks. But a schema is mostly unusable for modelling the real world and the domain of GISc, where semantic and behaviour is a main requirement.

One of the few positive arguments for relational technology is the fact that it is based on a secure model, meaning that there is always a result in finite time (cp. [Heuer, 1997]). This is mainly due to missing recursion and the orthogonal, closed and adequate algebra. Unfortunately SQL breaks the closed character of the relational algebra so that the secure model is irrelevant in most aspects.

Normalisation leads to further problems of the relational model and there are three requirements to reduce their effects: constancy of dependencies, constancy of the compound and minimality of the result. These requirements are to reduce redundancy, to eliminate wrong interpretations of relationships and to prevent a loss of information when decomposing and resynthesising entities from their tuple representation. As [Heuer, 1997] argues, the decomposition is not always able to keep the correct dependency among the parts and to ensure minimal relations. A violation against two of the above requirements. For example the coast of a small lake and the coast of the Atlantic Sea could be put in the same relation even as they have major differences, e.g. the tides will differ extremely. Modelling the real world with a RDBS will lead to a mix of disrupted parts with no perceivable or wrong interpreted interrelations among them. Resynthesising entities is done by joins, which is, when no index is used, a combination of two nested loops iterating over all tuples in both relations. Most of the performance is lost here.

Another major disadvantage, Heuer points out, is that it is not possible to directly model sets of attributes or multi-component attributes in a tuple. There is either the possibility to insert the tuple several times and adding redundancy, or to create a new relation with the appropriate normalisation.

Furthermore, the relational approach is not able to reflect different kinds of relationships, represented with foreign keys, and the semantics of it gets lost. Thus, it is not possible to distinguish between `IsA` (taxonomy) and `HasA/PartOf` (partonomy). This causes two major problems. First, it is not possible to model class hierarchies directly. Second, relationships like Functional Dependency (FD) and Multivalued Dependency (MVD) can lead to wrong conclusions about the associations between entities ([Heuer, 1997]).

When it comes to model geometry in a RDBS, there are three basic approaches. The first naive attempt is to use BLOBs handled by the application, which is inadequate. The second is to apply normalisation to each type of geometry, e.g. a 3D-point could be represented as a tuple in a relation POINTS with foreign keys to three tuples from a relation XYZ. Due to the resulting joins, the result would be horribly slow. The last approach, the SFS is based on, uses ADTs for user-defined data types and appropriate procedures. Unfortunately, todays systems do not implement most of the known spatial indexes, so that the performance is usually slow, too (cp. section 3.6.2 on page 52).

As a conclusion it can be said that the relational approach is not sufficient for the representation of phenomena in the real world. There is only few support for semantics, meta-information and the specific behaviour. In general, the Impedance Mismatch, referring to the set of conceptual difficulties between two different models, describes the problems that arise when working with a RDBS. Especially the object-relational Impedance Mismatch comprises all the problems that show up, when object-oriented application concepts can not be mapped to relational repository concepts. Finally, slivering the reality into small pieces, as normalisation does, is not the aim of an expert in Geographic Information Science and should not be for an expert in software engineering. Missing conceptual locality is also the main reason for performance bottlenecks.

List 13 summarises the main drawbacks of the relational approach and the reasons, why it is not sufficient for cadastral modelling.

- Behaviour is not part of a tuple, as it is an integral part of an entity.
- The few semantic constructs are not sufficient.
- The integration of meta-information is not sufficient.
- The possibilities of modelling aggregates are not sufficient.
- The possibilities of modelling semi-structured aspects are not sufficient.
- The necessary level of performance is not provided.

List 13: Shortcomings of the Relational Approach

For further reading the following books shall be recommended: [Heuer, 1997, DeMers, 2000, Laurini and Thompson, 1992]. Also [Schneider, 1997] tries to handle some of the problems above by introducing Realms. But obviously this is just a fight against symptoms, and does not solve the problem on its basis.

3.5.2 The Semi-Structured Approach

Three popular approaches of semi-structured modelling have been introduced in section 2.6 on page 20. The following paragraphs will discuss them in the context of GISc.

The **Extensible Markup Language (XML)** provides one popular dialect: the Geography Markup Language (GML). Developed by the Open Geospatial Consortium, GML is a grammar (written in XMLSchema) designed for the modelling, transport and storage of geographic information. At a minimum, these aims require sufficient semantic constructs, a small size by less overhead and redundancy and an appropriate performance level. But even those basic requirements can not be fulfilled by GML and [Stefanakis, 2002] argues that GML, and XML in general, is not sufficient for the modelling and storage of entities in the area of GISc. Some of the reasons for this will be discussed in the following. As GML is based on features - and not entities - it is not sufficient for the requirements on modelling behaviour and semantics. This is due to the definition of the term feature (cp. section 1.4 on page 5) and its restrictions. Regarding the second aim transport, the textual representation of XML and the resulting tree-structure do not provide the necessary preconditions to gain a small size and to avoid redundancy. Finally the storage of text does not allow fast access and the needed performance because of parsing and text conversion. [Bourret, 2004] argues that also the tree-structure of XML can lead to poor performance when the view of the application does not reflect this structure. A graph-like view will require additional navigation and search actions to use a repository, based on a tree.

The **Object Exchange Model (OEM)** (cp. [Goldman et al., 1996, Stefanakis, 2002]) is basis for the work of Emanuel Stefanakis in Athens and adapting OEM for GISc in theory has been quite successfully. But there are still shortcomings. The differentiation between nested subnodes (partonomy) and basic attributes of the node is not explicitly given. Due to this, it is necessary to handle it in the application. One could argue that the object-oriented approach is also not able to differentiate between them, but the modelling of behaviour and the encapsulation inside a repository provide the necessary constructs to achieve this anyway. Another disadvantage of OEM is the missing support for order of subnodes and - to the best knowledge of the author - the only solutions are based on an additional number, which can be interpreted in the application.

As **WebBus** is an enhancement of OEM, it supports order and it can differ between nesting and basic attributes. However WebBus is not able to model behaviour of nodes inside the repository.

As a conclusion it can be said that all three variants of the semi-structured approach are not sufficient for modelling the domain of GISc due to fundamental problems, itemised in list 14.

- Behaviour is not part of a node, as it is an integral part of an entity.
- The set of supported semantic constructs is not enough.
- The necessary level of performance is not provided.

List 14: Shortcomings of the Semi-Structured Approach

3.5.3 The Object-Oriented Approach

As [DeMers, 2000] argues, the object-oriented approach has an enormous potential for the representation of real-world phenomena in application and repository. It supports the necessary semantic constructs and is able to model complex entities, including behaviour, attributes and aggregates. A sufficient level of performance and efficient navigation by sending messages is provided. An unique ontology can be shared between domain, application and repository. Due to this, there is also no Impedance Mismatch between application and repository, as these are based on the same model.

Nevertheless, there are shortcomings of the object-oriented approach. As semi-structured facets are not supported directly, a meta-layer has to be created. But due to encapsulation and the modelling of behaviour, this layer is transparent and only visible to the object itself and thus, integrated into the repository.

It is also often argued that the close linkage between an object-oriented combination of application and repository is a restriction and inhibits necessary independence. But the development of Object Management Systems that can store objects from different applications, based on different programming languages, show the opposite. The example GOODS [http://www.garret.ru/ knizhnik/goods.html] does not depend on the specific language, a client has been written in. A meta-object protocol is used to achieve this and enables the system to provide the necessary independence. Also GemStone/S (cp. section 4.4 on page 67) provides access from clients, based on different programming languages - e.g. Smalltalk and Java.

Finally it is often stated that the performance of an Object Management System can be slow, when a search is done over a set of objects. But the modelling of objects makes navigation an efficient alternative, and searching all entities becomes seldom.

Even though, searching a set of objects is essential and several indexes (cp. section 3.6.2 on page 52) have been proposed to optimise this task. The object-oriented approach makes it possible to integrate indexes in a repository easily, while other approaches would require major changes of the repository itself. In fact, todays DBSs do support only a few of the known indexes.

- Semi-structured aspects have to be modelled using a meta-layer.

List 15: Shortcomings of the Object-Oriented Approach

As a conclusion it can be said that the object-oriented approach is sufficient for modelling the domain of GISc. The only disadvantage (cp. list 15) can be easily compensated by a meta-layer.

3.5.4 Summary

Regarding geospatial applications, relational databases have fallen short of effectively achieving that purpose for two main reasons: 1. the relational model has not provided a sufficiently rich set of semantic constructs to allow users to model naturally geospatial application domains; 2. relational technology has not delivered the necessary performance levels for geospatial data management.

[Longley et al., 1999b]

The statement above summarises the fundamental shortcomings of the relational approach and it is also true for semi-structured technology. Table 3 on page 47 itemises each requirement and compares the three approaches, discussed in the following. The used symbols have the following meanings:

✓	OK, the item is supported sufficiently.
✳	Well, the item is supported only to a degree.
–	No, the item is unsupported.

A logical model for real-world phenomena shall be able to represent taxonomy (`IsA`) and partonomy (`HasA/PartOf`) and to differ between them. While the object-oriented approach supports both including its differentiation, the others only provide constructs for `HasA/PartOf` relationships.

	Relational	XML	OEM	WebBus	OO
Taxonomy	−	−	−	−	✓
Partonomy	✓	✓	✓	✓	✓
Its differentiation	−	−	−	−	✓
Attributes	✓	✓	−	✓	✓
Aggregates	✓	✓	✓	✓	✓
Its differentiation	✓	✓	−	✓	✓
Behaviour	✳	−	−	−	✓
Many-To-Many	✓	−	✓	✓	✓
Semi-Structured Aspects	−	✓	✓	✓	✳
Conceptual Locality	−	✳	✓	✓	✓
Conceptual Performance	−	−	✓	✓	✓

Table 3: Opposing the Logical Models

A logical model shall also be able to represent nested subentities (aggregates) and attributes of an entity and to differ between them. All approaches, except OEM, do so.

Modelling behaviour inside the repository is crucial and only the object-oriented approach supports it in a sufficient way. Relational technology came up with so called Active Database Rules, but these have not kept the original promises. Also object-relational DBSs were not able to reach the flexibility of pure object-oriented systems. XML, OEM and WebBus do not have any support for modelling behaviour.

Today, the representation of many-to-many relationships is supported by relational, object-oriented and the most semi-structured approaches. Only XML does not support it directly, because of its flat, textual character.

Research on getting rid of a fixed schema was not solved for relational systems and resulted in semi-structured technology. Nevertheless, the object-oriented approach, especially Smalltalk, is able to integrate semi-structured aspects with dynamic typing and meta-layer concepts. It is also self-describing because of its reflexive manner.

Due to normalisation, conceptual locality can not be achieved with relational systems and joins are necessary to resynthesise the parts of an entity from tuples in relations. Also the tree-structure of XML requires additional efforts when many-to-many relationships are traversed or nodes refer to others by manual identifiers. Only the object-oriented approach, OEM and WebBus provide conceptual locality. Especially in object-oriented systems, traversing the graph of objects becomes transparent and more flexible by sending messages.

Finally the query performance of relational systems has always been slow due to normalisation and the necessary joins. XML also reduces the possible performance because of its tree-structure and the resulting search for manual identifier. The message concept of objects is much more efficient than joins or tree-based navigation, and also allows the integration of behaviour. For example, Double Dispatching enables objects to use polymorphisms instead of an "if..then..else" to make a decision for further navigation. Nevertheless, the overall performance will be influenced by additional parameters, discussed in section 3.6.1 on page 51.

Table 3 summarises the arguments stated before. As a conclusion it can be said that the object-oriented approach suites best for the modelling of real-world phenomena and should be the primary choice for application and repository.

3.5.4.1 Mapping

Accessing relational or semi-structured repositories from an object-oriented application is a main task of a Geographic Information Infrastructure (GII). As [Uitermark, 2001] argues, this requires transformation rules between different ontologies and their models. While the mapping from a simple model to a richer model is quite simple, the other way round usually involves a loss of information.

Accessing **relational repositories** is mainly based on views, resynthesising all needed tuples to create objects. A lot of systems build upon the object-relational mapping and it has been proved that the transformation from tuples to objects is doable. This is mainly due to the fact that the object-oriented approach is able to model the concepts of relational technology.

Accessing **semi-structured repositories** is more difficult. The object-oriented approach does not directly support the concept of a variable structure. A solution to this is an additional meta-layer. An object, which represents a node, could manage its subnodes in a collection. This collection is a meta-layer, allowing the integration of the semi-structured aspect in an object-oriented application.

In our case, the object-oriented approach provides the richest model. Therefore, the access to other systems based on other models, is doable.

Problems arise when it is needed to map a rich model to a less rich one. Storing objects in a relational or semi-structured DBS involves a loss of information and the last sections have shown this[7]. Thus, with the example of figure 1 in mind, this thesis argues to handle each repository, with a different model than the application, mainly as a retrieval source. Every new or updated information shall be primarily handled by a repository, based on the same model as the application. In our case, an object-oriented application requires an Object Management System. Nevertheless, new or changed objects shall be synchronised to those retrieval repositories as it is possible.

3.6 The Internal Layer

The internal model of a DBS or OMS specifies the structure of entities in memory and their organisation and physical access paths. It derives from the logical model and completes it with technical and platform-dependent stipulations.

One aim of the internal model is to gain physical independence of data or objects. This means that changes to the internal model itself, e.g. changes to the location of objects or structure of data, do not affect parts of the logical model. This way, a collection of garbage is possible and optimisation and reorganisation becomes easy to accomplish. For this purpose, rules of transformation between the logical and internal model will be used. As entities only exist in form of bits in the internal or physical representation, the

[7] e.g. the Impedance Mismatch, many-to-many, taxonomy and partonomy

management system (DBMS or OMS) is build according to the model of the internal layer to extract information at the time of the transaction.

It is not the aim of this thesis to comment on the internal model in detail, but to give a review of its central concepts. As mentioned above, there are two basic aspects of the internal model:

- structure of information
- organisation and access paths

List 16: Basic Aspects of the Internal Model

When storing entities of the relational, semi-structured or object-oriented model, the structure of tuples, nodes and objects is pretty different.

A tuple is mostly implemented as a record bearing the aspects of a Hard Disk Drive (HDD) in mind. Issues of performance led to the practise to design the structure of tuples according to the position of the heads of the HDD. As a result the Indexed Sequential Access Method (ISAM) and Virtual Storage Access Method (VSAM) emerged. The organisation of tuples is usually solved sequential (in form of lists), by using trees (like the B-Tree) or via hashing (cp. [Heuer, 1997]).

As XML is a textual format it will typically be stored as text. Anyway there is also the possibility to use DOM and store it as a binary graph with nodes and edges as OEM and WebBus do. Trees and graphs in general are the main organisation method (cp. [XML 1.1, Stefanakis, 2002, Goldman et al., 1996, Fiedler, 2002]).

The structure of objects is a private property and shall not be altered except by the object itself. Besides this, the organisation is based on a graph where the nodes are objects and the edges are references, not visible to the user. The access is done by altering the top of the stack with adequate methods (cp. [Goldberg and Robson, 1983]).

Regarding geometry, most relational systems provide the possibility to create an Abstract Data Type (ADT) with the appropriate operations. For example, the implementation of the SFS for SQL in PostGIS is done that way to avoid normalisation. Since Smalltalk is based on objects, there is no special handling at the internal layer for those, representing a geometry. Also semi-structured nodes are not stored differently according to their meaning.

3.6.1 Performance

The performance of a management system depends on multiple aspects. In this context the biggest bottlenecks and the reason for them will only be mentioned.

Generally it can be said that the non-persistent and expensive primary storage, like RAM, is a lot faster than the persistent and cheap secondary storage, like HDD. Thus, a system that keeps potential information of interest in the primary storage is faster than one, reading everything from the bottleneck HDD. To achieve this, technology came up with clustering related information near at each other, so that reading a page also pushes near information into the primary storage and the head is moved much less. The relational model usually keeps the bits in a tuple (as a record) and organises these sequential. GemStone/S can organise objects with the message #cluster. Also generation scavenging, as the preferred way of garbage collection, contributes to this by grouping objects into generations. XML, as the most popular representative of the semi-structured model, is slow because of its textual representation. In contrast, OEM, WebBus and even DOM for XML, can cluster related nodes and thus, become faster.

The next big factor to performance is the underlying model. In history, when hardware was considerably slower than today, relational systems were second quality. Normalising information into relations lead to multiple joins when resynthesising entities. If there is no index available, a join between two relations is a nested for-loop and takes a lot of time. Today, the hardware is less important and people tend to use relational systems for all their tasks. But the cause for the loss of performance is not gone and will appear when the size of a RDB grows. The alternative is to avoid joins and set based searches, discussed in the following paragraphs.

Conceptual locality, as mentioned before, ensures fast access to related aspects of an entity. This is usually solved by direct references. In the case of Cincom® VisualWorks® Smalltalk, only a message send is needed, altering the top of the stack. Costly joins of relational systems, which lack conceptual locality, are necessary to resynthesise the information from different tuples.

The second way to improve performance is to avoid the inspection of mostly all members in a set, as the relational approach does. Instead, navigation via direct references, by accessing the structure of nodes or sending messages to objects, is more efficient. However, this requires additional efforts in the management of nodes and objects.

Regarding navigation, XML makes an exception here. As it is based on a tree, following an edge is only possible when the view equals the organisation of the nodes. Only OEM, WebBus and the object-oriented approach ensure fast navigation.

The third aspect of performance, presented here, is the access path to information by using indexes. As we deal with spatial entities, the next section will primarily discuss the typical ways of organising multidimensional information.

3.6.2 Indexing

Searching in space is usually a costly task (cp. [Samet, 1990a,b]). Thus, indexes for spatial organisation have been developed. Without them, a search would require a sequential iteration over all entities in the repository in the worst case. The core of indexing is a special search tree or hash, which can be traversed quickly to find a particular entity. Unfortunately only a few indexes have been implemented in todays systems. Some of the most popular approaches will be adumbrated in the following.

The **R-Tree** (cp. [Longley et al., 1999b, PostGIS Manual]) is a BSP-Tree and one of the few implemented indexes (e.g. PostGIS supports it). The basic structure is a tree which splits space with hierarchically nested, and possibly overlapping, bounding boxes. Entities are referenced in the leaf nodes. Different subtypes, like the quadratic or linear R-Tree, support different algorithms of splitting or merging leaf nodes (also called balancing). In 2004, a new variant, the priority R-Tree, has been published by [Arge et al., 2004] and claims to be as efficient as the currently most efficient methods, being worst-case optimal at the same time.

The **Quadtree** in 2D, and the similar **Octree** in 3D, have been primarily used to store geometries as a raster model, by interpreting the leaf nodes as pixels or voxels. The usage as index is known, but has not been implemented in the most popular Database Systems. The tree has been discussed in detail in books like [Samet, 1990a,b, Longley et al., 1999b] and is based on recursively splitting a surface or space in four or eight non-overlapping nested boxes. As entities can range over multiple of such boxes, the reference will be managed by all nodes in the tree, in contrast to the R-Tree, which only references entities in the leaf nodes.

The **kd-Tree** (cp. [Longley et al., 1999b]) is an index over a k-dimensional space. Each level in the tree represents a dimension in such a way that

$$dimensionOf(level_N) = dimensionOf(level_{N+M*k})$$

with $N \in \mathbb{N}\backslash 0$ and $M \in \mathbb{N}\backslash 0$. The first level is the root node and is labelled as level 1. While the normal kd-Tree references entities in each node, the Adaptive kd-Tree uses only the leaf nodes for this.

Hashing is an indexing method, which is based on a function that maps values from a bigger co-domain to a smaller co-domain. Usually the hash of a spatial position will be used as key in a fixed size array. One example of its usage are **Grid Files** (cp. [Longley et al., 1999b]), which split space into a grid (multidimensional array) where each spatial dimension is organised by a linear hash. The cells of the grid contain references to entities.

3.7 Summary

The last sections have shown that representing real-world phenomena is about modelling the domain of GISc. To implement a Geographic Information System, the terminology of experts has to be understood clearly. An ontology about the domain concepts and the semantical mapping to implemented concepts has to be explicitly given. To communicate this model of concepts, an ubiquitous language unambiguously reflects all the terms of the domain that describe a concept of its ontology.

Based on this, a phenomenon of reality can be represented by abstracting its perceptions. A conceptual model of all the concepts of the domain specifies the requirements to an implementation. The logical model has to support these, to successfully reflect the terms of experts and to finally build a powerful GIS.

As we do not attempt to develop a new management system as repository for geographic information, the internal model of it is important for performance and optimisation and not the central issue of this thesis.

The results of the analysis of relational, semi-structured and object-oriented approach show that the concept of objects suites the requirements of the domain best. Nevertheless, there are implemented Geographic Information Systems based on relational and semi-structured technology, where the missing concepts of the repository have been balanced in the application.

The following sections verify the above insights by means of the development of a prototype and further analysis of selected representatives for the theoretical approaches, namely PostGIS, LORE and GemStone/S.

4 Practical Research

To corroborate the gained insights from previous sections, a prototype application has been developed (cp. section 4.1). It implements a small and simple model of cadastral surveying, using the object-oriented system Cincom® VisualWorks® Smalltalk.

In the second step, this model has been mapped to management systems, namely PostGIS, LORE and GemStone/S. Sections 4.2 to 4.4 discuss the problems of this process and its implementation.

The experiences in building a GIS and storing its entities in different repositories have shown that technically nearly everything is possible. The question is if the repositories can model the domain without additional major efforts. This includes an easy semantical mapping between different ontologies.

4.1 The Prototype

The implemented prototype application is meant to run on a PDA and integrates an external GPS receiver for mobile acquisition of geographic information in the area of cadastral surveying. Details of the setup have been itemised in list 17.

- PDA: MEDION® MD 41600 with Intel® PXA255 CPU
- GPS Receiver: RoyalTek Sapphire G-Mouse (RGM-2000)
- OS: Microsoft® Pocket PC 4.20
- Environment: Cincom® VisualWorks® Smalltalk

List 17: Prototype Setup

The first aim of the prototype is the implementation of the conceptual model, from section 3.4 on page 33, in the domain of cadastral surveying. The second aim is to map this logical model to the repositories PostGIS, LORE and GemStone/S.

Using the GPS receiver in Cincom® VisualWorks® Smalltalk, required libraries for serial ports and the NMEA-0183 standard, developed by the author at Georg Heeg eK. An event-driven approach has been used to make the current position available.

The prototype is based on a layered architecture, differing between four main levels (cp. figure 7 and [Evans, 2004]).

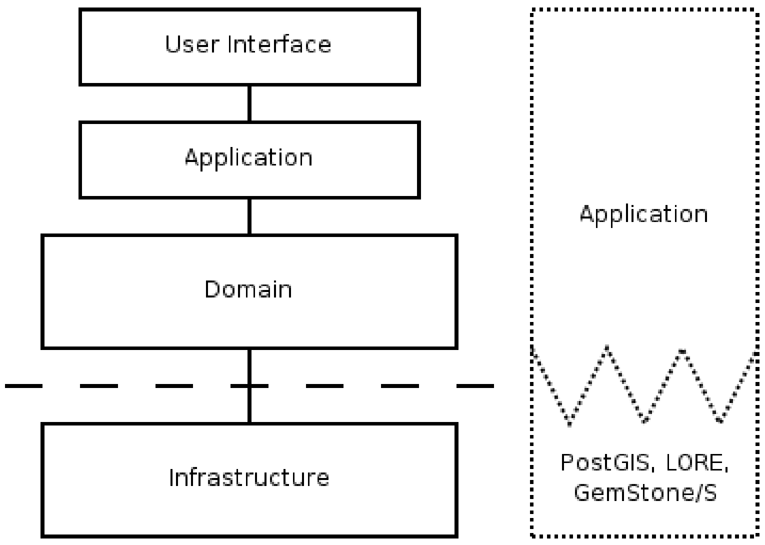


Figure 7: Basic Layers of the Prototype

The User Interface (UI) is responsible for displaying the current position - obtained from the GPS receiver - and implements a few basic use cases. It is based on Wrapper, a framework specific to Cincom® VisualWorks® Smalltalk.

The application layer only implements the handling of entities in terms of adding, manipulating and removing objects. Especially the creation of complex objects of the domain layer is realised with so called Object Factories (cp. [Evans, 2004]).

The domain layer implements specific entities, like boundary stones or land parcels. It is the most interesting layer, as all the modelling is done there. Depending on the used management system, it melts with the infrastructure, which is part of the repository.

However, as any project progresses with time, knowledge about the domain grows and is most substantial at the end of it. Thus, it is not meant to provide an application being a complete GIS. So the following is the result of developing a simple prototype just to show the possibilities of modelling a domain with Smalltalk. It does not claim to be a perfect solution and indeed, it has been refactored a lot of times in the process of development.

To give a basic understanding of the software, the UI will be presented. Among others, it comprises five main windows, called screens, visualised as transition diagram in figure 8.

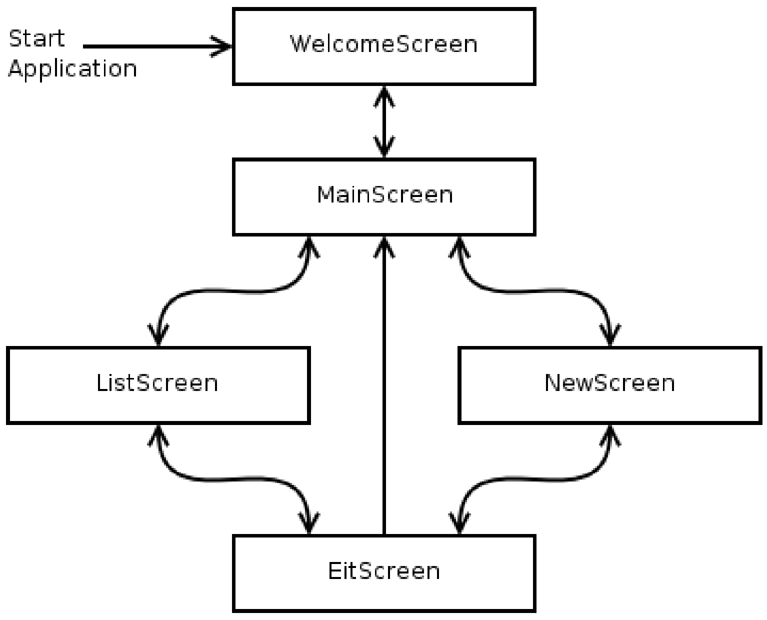
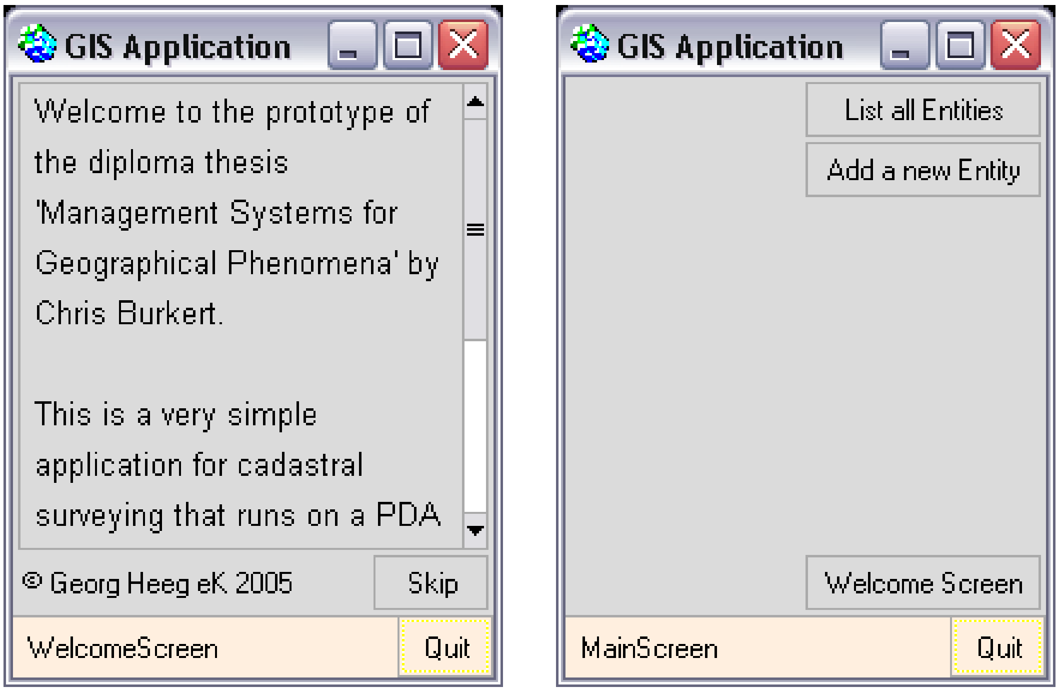


Figure 8: UI Transition Diagram

The **WelcomeScreen** (cp. figure 9 on page 58) is a simple introduction to the prototype. It displays a text about the application and the copyright. All screens have a `Quit`-button to close the application and a status bar to display the kind of the active screen. The button `Skip` directs the user to the next screen.

The **MainScreen** (cp. figure 9 on page 58) is a dispatcher for the tasks a user wants to accomplish. Currently there are two main tasks supported. To list the available entities for further editing or removing, the button `List all`

`Entities` directs the user to the ListScreen. This is a sufficient simplification, as the prototype only has to deal with a few entities. The button `Add a new Entity` shows the NewScreen to select the class of the new entity and finally guides the user to the EditScreen, specific to the kind of the new entity. The button `Welcome Screen` leads the user back to the first screen after starting the application.


Welcome to the prototype of the diploma thesis 'Management Systems for Geographical Phenomena' by Chris Burkert.

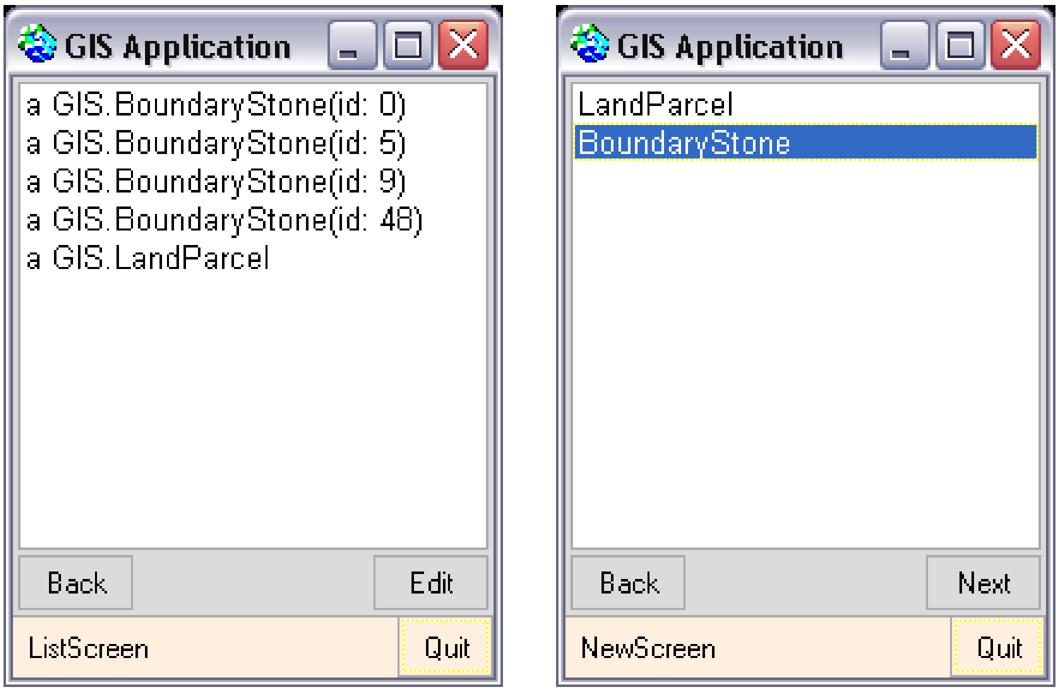
This is a very simple application for cadastral surveying that runs on a PDA

© Georg Heeg eK 2005

Figure 9: Screenshots of Welcome and Main Screen

The **ListScreen** (cp. figure 10 on page 59) gives an overview of the available entities with a short description of each and a menu for further actions. The `Back`-button provides a way to leave the screen without doing anything. To edit a selected entity, the first item of the menu and the `Edit`-button lead the user to the EditScreen. The menu also provides the possibility to remove a selected entity and to inspect one with a standard Smalltalk Debugger.

The **NewScreen** (cp. figure 10 on page 59) is a window to select the class of a new entity. As before, the `Back`-button provides a way to leave the screen without doing anything. The `Next`-button creates a new instance of the selected class and guides the user to the appropriate EditScreen for the completion with further information.



```
a GIS.BoundaryStone(id: 0)
a GIS.BoundaryStone(id: 5)
a GIS.BoundaryStone(id: 9)
a GIS.BoundaryStone(id: 48)
a GIS.LandParcel
```

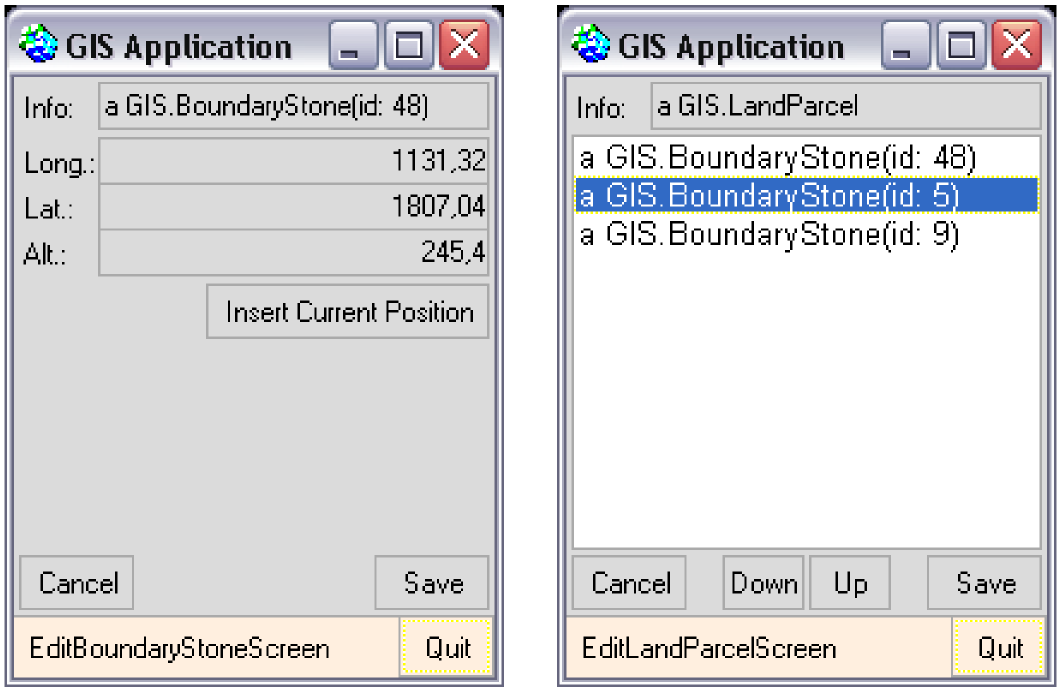
```
LandParcel
BoundaryStone
```

Figure 10: Screenshots of List and New Screen

The look of the **EditScreen** (cp. figure 11 on page 60) depends on the object being altered. There are currently three screens for abstract entities, boundary stones and land parcels. All these provide a `Cancel`- and a `Save`-button and a short description of the entity.

When editing a boundary stone, the position of it will be displayed. The button `Insert Current Position` retrieves longitude, latitude and altitude from the GPS-receiver and fills in the appropriate values. The screen for editing a land parcel provides an ordered list of the boundary stones that make up its boundary. Additionally a menu comprises items to add, remove, edit and inspect them. To alter the order of boundary stones, two buttons `Up` and `Down` have been integrated.

The class hierarchy used for the domain layer of the prototype is being kept very simple and does only reflect the terms in list 2 on page 34. The sub-


GIS Application

Info: a GIS.BoundaryStone(id: 48)

Long.: 1131,32

Lat.: 1807,04

Alt.: 245,4

Insert Current Position

Cancel Save

EditBoundaryStoneScreen Quit

GIS Application

Info: a GIS.LandParcel

a GIS.BoundaryStone(id: 48)
a GIS.BoundaryStone(id: 5)
a GIS.BoundaryStone(id: 9)

Cancel Down Up Save

EditLandParcelScreen Quit

Figure 11: Screenshots of different Edit Screens

classes of the abstract class `Entity` implement and reflect the appropriate characteristics of land parcels, boundary stones and their boundaries. Whereas land parcels and boundary stones are modelled as identifiable things and stored in the repository, a boundary will be created on the fly and is not persistent.

```
Object
  Entity
    Boundary
    BoundaryStone
    LandParcel
```

The class hierarchy of the geometry of entities only implements the classes `Position` and `Shape` as subclasses of the abstract class `Geometry`. As stated above, their instances encapsulate the specific representation of shape and position. For the issue of the prototype it was only necessary to implement the concept of a point with longitude, latitude and altitude. Nevertheless, further representations are necessary.

```
Object
  Geometry
    Position
    Shape
```

The experiences with the prototype turned out that the simple example of cadastral surveying is more complex than expected, but it was possible to model all aspects with Smalltalk. For example, a boundary uses aggregates for its boundary stones. Behaviour is used to provide the position of a land parcel, obtained from its boundary stones.

Thus, the implementation of cadastral concepts shows that it is not possible to create a rigid schema that fits the domain without loosing information and flexibility. Instead, a supple and flexible system is necessary to model that domain. Smalltalk has proved to be such a system. Storing entities requires explicit semantics, specific behaviour and the modelling of attributes and aggregates at a minimum. Cincom® VisualWorks® Smalltalk is able to model much more: the certain characteristics of a domain. The result is a set of different classes, representing concrete domain concepts.

While developing the prototype, it was also necessary to be able to refactor the model as time went by (cp. [Beck, 1999, Evans, 2004]). Thus, evolution of ideas, knowledge and the model itself has to be supported by application and repository. This is a major requirement for the following sections.

4.2 Relational Systems - PostGIS

PostGIS is an extension of the RDBS PostgreSQL for the handling of spatial data. It is not meant for the modelling of a domain. Spatial data is the central concept and it has been realised with an implementation of the Simple Features Specification (SFS) by the Open Geospatial Consortium.

There are several resources for information on PostGIS including the PostGIS Website [http://postgis.refractions.net/] and the PostGIS Manual [http://postgis.refractions.net/docs/]. The PostgreSQL Website [http://www.postgresql.org/] hosts information about the underlying RDBS.

As said before, PostGIS does not provide support for the modelling of domains. It is based on relational technology, which lacks a sufficient set of semantic constructs. It is based on the SFS that reduces an entity to a feature (cp. section 2.4 on page 14). Therefore it can be said that PostGIS is unsuitable for representing phenomena of the real world.

Nevertheless, a practical analysis has been done. The installation of PostgreSQL 8.0.2 and PostGIS 1.0.0-rc6 on Microsoft Windows XP® was no problem. To use the extensions of PostGIS, an ordinary table has to be created. After this, a column of type `geometry` with an appropriate subtype can be added. In general, each SQL-query on a geometry column follows the WKT standard. Due to this, the allowed subtypes in such a column can be one of the itemised terms in list 18.

- POINT
- LINESTRING
- POLYGON
- MULTIPOINT
- MULTILINESTRING
- MULTIPOLYGON
- GEOMETRYCOLLECTION

List 18: Allowed Geometry Types in PostGIS

PostGIS also provides the procedures for SQL, specified by the SFS, e.g. Distance(), Intersects() or Contains(). This kind of behaviour has the disadvantage that there is no way to easily add new behaviour so that database rules or standard SQL has to be used for further management in the application.

As PostGIS is based on the relational system PostgreSQL, there is no conceptual locality. Normalisation slivers the entity into tuples of several relations and joins are necessary to resynthesise it. A lot of the performance is lost due to this fact.

A refactoring of the schema, also called schema evolution, is not supported for geometries. In fact, it is possible to remove or add columns, but a change of the geometry type (e.g. streets shall be stored as polygons instead of linestrings) is unsupported.

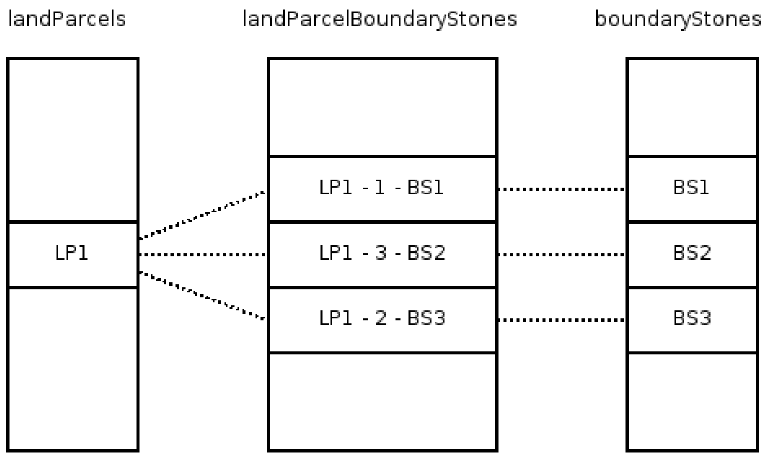


Figure 12: Relations for Cadastral Surveying

Representing the concepts of cadastral surveying required relations for land parcels, boundary stones and their relationships (cp. list 19 and figure 12).

- $R_{landParcels} = (ID, owner)$
- $R_{boundaryStones} = (ID, position)$
- $R_{landParcelBoundaryStones} = (ID, landParcelID, nr, boundaryStoneID)$

List 19: Relations for Cadastral Surveying

$R_{landParcels}$ identifies each tuple with an ID and stores its owner. $R_{boundaryStones}$ also uses an ID and has a column `position` of type geometry and SFS-subtype POINT. To keep compatibility to WKT, those positions have only a 2D expansion - a major loss of information.

As a land parcel consists of ordered boundary stones, the order has to be stored in some way. $R_{landParcelBoundaryStones}$ provides a column `nr`, which keeps track of that. For example, figure 12 shows a land parcel (LP1), which consists of three boundary stones (BS1, BS3 and BS2 in that order). Unfortunately, this requires complex actions when this order will be changed, by inserting or deleting tuples in the relation. Also other solutions to this require costly update procedures.

As a conclusion it can be said that PostGIS and PostgreSQL do not fit the requirements of modelling the domain of GISc. Next to the problems of the relational approach, itemised in list 13 on page 43, the system does not provide sufficient support for the evolution of the model and refactoring in general.

4.3 Semi-Structured Systems - LORE

The Lightweight Object REpository (LORE) is a DBS for semi-structured data, including XML and OEM. Even though it was a success (as declared by the Stanford University Database Group) the development was stopped in the year 2000, after publishing a lot of information about it [Goldman et al., 1996, 1999, 2000, Quass et al., 1996, McHugh et al., 1997].

Nevertheless, binaries of it are available at the LORE Website [http://www-db.stanford.edu/lore/] and version 5.0 has been successfully tested on a Suse Linux 9.3 system. The available tools for the command line and a library for C++, enable the user to get access to the system. Unfortunately it was not possible to connect Cincom® VisualWorks® Smalltalk with LORE in the short time. Thus, the following is based on experiences with the command line query tool.

The access to a LORE repository is based on the LOREL Query Language. It is, similar to SQL, a descriptive kind of language for the navigation through nodes. To connect to the system, one has to create a database and run the query tool on it. As LORE does not require a schema, there is and can not be a special support for spatial, temporal or any other form of nodes.

The requirements on modelling real-world phenomena are only supported partly. Taxonomy and behaviour have not been integrated, as the semi-structured approach does also lack to specify them. LORE supports attributes and aggregates for XML, but not for OEM. The differentiation between them is solved in the query language. The next example shows this.

```
<institute>
  <person name="Arthur Dent" age="42" />
  <person name="Tricia McMillan">
```

```
    <age>42</age>
  </person>
</institute>
```

Based on this simple document, we want to search persons that are 42 years old with the following statements. The first select would answer only those persons, where the age is stored as subnode (>). The second answers those persons, with an attribute age (@). The last one would not mind about the difference between subnode or attribute and search both.

```
# only search for subnodes named 'age' - Tricia
select institute.person
 where institute.person.>age = "42";

# only search for attributes named 'age' - Arthur
select institute.person
 where institute.person.@age = "42";

# search for both named 'age' - Tricia and Arthur
select institute.person
 where institute.person.age = "42";
```

Furthermore, evolution of the schema, as one effect of refactoring the model, is not supported by LORE. This is due to the fact that there is no schema at all. Thus, every change to the model has to be embedded by hand, including a reorganisation of data.

LORE will handle GML-documents like any other XML-document. Thus, the application has to model a lot of the domain, which should be part of the repository. The following example shows a typical document for cadastral surveying:

```
<LandParcel gml:id="lp1">
  <owner> This is mine </owner>
  <BoundaryStone xlink:href="#bs2" />
  <BoundaryStone xlink:href="#bs1" />
</LandParcel>
```

```
<BoundaryStone gml:id="bs1">
  <gml:location>
    <gml:Point>
      <gml:coordinates>
        31:45:00S 110:50:00E
      </gml:coordinates>
    </gml:Point>
  </gml:location>
</BoundaryStone>

<BoundaryStone gml:id="bs2">
  <gml:location>
    <gml:Point srsName="#myReferenceSystem">
      <gml:coordinates>
        21 42 84
      </gml:coordinates>
    </gml:Point>
  </gml:location>
</BoundaryStone>
```

For the representation of many-to-many relationships, XPointer has been used, but there is also the possibility to use XLink or XPath. However, these linkages have to be handled by the application and can not be managed by LORE.

As a conclusion it can be said that LORE and GML do not fit the requirements of modelling the domain of GISc. Next to the problems of the semi-structured approach, itemised in list 14 on page 44, the system does not provide sufficient support for the evolution of the model and refactoring in general. Nevertheless, GML is one of the best formats for exchanging geographical data between applications.

4.4 Object-Oriented Systems - GemStone/S

The Object Management System GemStone/S, developed by GemStone Systems [http://www.gemstone.com/], is available since 1987 and spread all over the world after that. It is one of the most popular systems for the management of objects, especially for clients, running Java and Smalltalk.

As GemStone/S and Cincom® VisualWorks® Smalltalk are based on the same paradigm, there is no mismatch between application and repository. Thus, modelling the domain of GISc in one of these is coequal and adequate, as the experiences with the prototype in section 4.1 on page 55 have shown.

The basic architecture of GemStone/S, visualised in figure 13, is made up of clients accessing a server, which usually runs one unique Stone process and multiple Gem processes. In general, all objects inside the system are persistent by default.

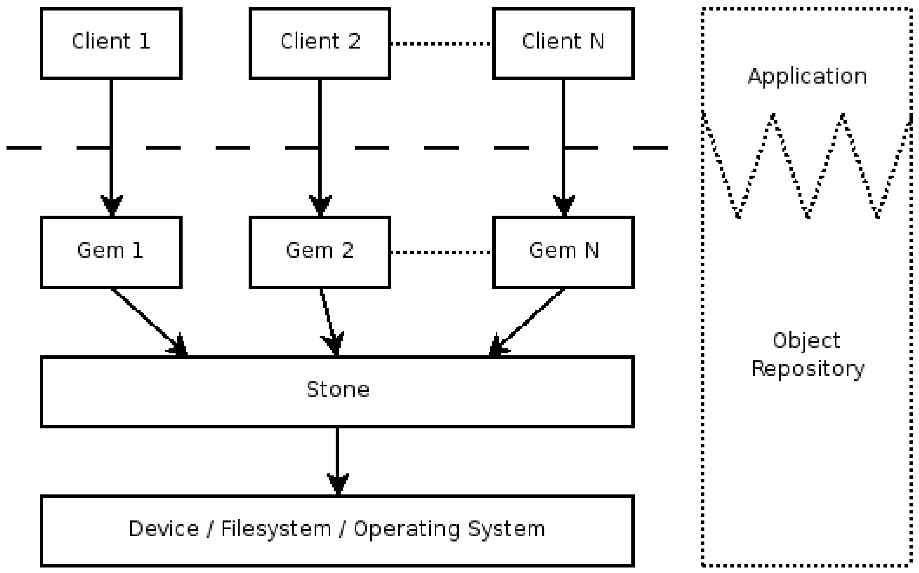


Figure 13: GemStone/S Architecture

While each Gem process handles one client, the Stone process is responsible for ensuring integrity on the objects. The following will discuss internal aspects of GemStone/S. For more information in general, visit the

GemStone/S Website [http://www.gemstone.com/products/smalltalk/] and compare [Heuer, 1997, McFarland et al., 1999].

GemStone/S provides its own language, called "GemStone Smalltalk", for the definition of classes. In contrast to traditional Smalltalk, a kind of static typing by the definition of constraints is possible, but not necessary. This is due to optimisation reasons.

For each group of similar objects usually two classes will be defined. The first, e.g. SomeClass, is the class that represents the description of objects. The second, e.g. SetOfSomeClass, provides a description of a container for such objects. If there is no need to search all objects of the same kind, the second class is not necessary and access is mostly done by navigating through an aggregate of objects. Nevertheless, GemStone/S provides typical methods for iterating over a set. The messages #select:, #reject:, #detect: and #remove: behave like their counterparts in Cincom® VisualWorks® Smalltalk, but accept an additional kind of block in the form `{:obj|}` instead of `[:obj||]`. These blocks enable GemStone/S to optimise iterations over sets and allow a special point-operator, for the direct access to instance variables without sending a message.

As an example the following two lines represent the same expression. While the first one is standard Smalltalk, the second uses direct access to the instance variable name inside a special block.

```
aSetOfSomeClass select: [:obj | obj name = 'My Name'].
aSetOfSomeClass select: {:obj | obj.name = 'My Name'}.
```

These concepts enable GemStone/S to optimise an iteration over a set, but it also breaks encapsulation. Consequently, this shall only be used when there is a need for better performance. One could say, it should only be done in the last stage of the saying "Make it Work, Make it Right, Make it Fast".

Further optimisation comprise messages for clustering objects in memory (#cluster, #clusterDepthFirst) and for the creation of indexes (identity and equality index).

To avoid searches over a set, GemStone/S makes transparent navigation possible. References connect objects - like edges in a graph - and behaviour makes the handling of them even more flexible. For example if one searches for all boundary stones in a certain area around another stone, it is not needed to search the set of all possible ones. Instead, an appropriate method could find the neighbours of the first one, by traversing the adjacent boundaries.

GemStone/S also supports ACID transactions (#commitTransaction, #abortTransaction), multi user access with authentication and authorisation, error handling for the securing of integrity and backup/restore procedures.

Like Cincom® VisualWorks® Smalltalk, the evolution of the model (e.g. by changing instance variables or modifying methods) is possible and easy. This is important as a model usually changes with deeper knowledge of the domain and refactoring becomes necessary (cp. [McFarland et al., 1999]).

Recapitulating these facts reaches the conclusion that GemStone/S is a full object-oriented management system with support for objects, classes and meta-classes, simple inheritance, self defined methods and the possibility to redefine them. Belonging to the modelling of real-world phenomena, it is coequal to Cincom® VisualWorks® Smalltalk and provides all necessary concepts that have been used in the application. This comprises a sufficient rich set of semantic constructs and the possibility to model behaviour, attributes and aggregates. It enables the user to put a lot of code - which makes up the implemented model of the domain - inside the repository that would otherwise strain the application.

As a conclusion it can be said that GemStone/S and the object-oriented approach do fit the requirements of modelling the domain of GISc including support for refactoring and evolution of the model. The only disadvantage of missing semi-structured aspects (cp. list 15 on page 46) can be compensated easily by an additional meta-layer.

5 Summary and Conclusions

This thesis attempted to analyse approaches for modelling the domain of GISc by means of a cadastral surveying example. It has been argued that ontologies are necessary to make semantics explicit and to form a fundamental ubiquitous language.

Furthermore, the abstraction of a phenomena requires the creation of a conceptual, a logical and an internal model. The discussion lead to the conclusion that each level of abstraction has to be based on the ontology.

Finally the analysis of technology for the logical layer points out, that modelling a domain is more than creating data structures and schemata. It has been argued that objects are the best approach for this task.

The following paragraphs will summarise the important insights and conclusions of this thesis and provide a forecast for future research.

Managing geographical phenomena of the real-world is a task of understanding the domain of Geographic Information Science at first. This comprises the emergence of an ontology that is used throughout a project and makes up the basis for an ubiquitous language, spoken by domain experts and developers.

> Thus, future research should concentrate on the specification of an ontology for GISc. The work that has been done (e.g. the Geo-Information Terrain Model (GTM) of The Netherlands or the project proposal of Goodchild et. al.) is only a starting point and has to be deepened and verified much more.

Furthermore, the conceptual model of a Geographic Information System must use the same concepts that are part of the ontology to make it relevant. The model comprises semantics, behaviour, attributes and aggregates at a minimum and represents real-world phenomena with all their specific aspects, but without any attention to computer science.

> Thus, future conceptual models, like the standards of the Open Geospatial Consortium, must reflect the concepts of the ontology. They must include explicit semantics between model and ontology, entity specific behaviour, attributes and aggregates and all the aspects that make up a concrete entity.

Also a logical model shall be based on the same ontology and makes the conceptual model relevant, when it supports all the concepts of it. More than programming, it is about modelling the domain and the logical layer has to provide the necessary constructs to realise the conceptual model in application and repository without constraints.

> Thus, with the results of previous sections in mind, modelling real-world phenomena at the logical layer is best done with an object-oriented approach, because the relational and semi-structured approaches do not meet the demands of GISc. This is true for both, application and repository.

A Geographic Information Infrastructure (GII) requires the application to handle different repositories. It must be build on the most powerful and rich model, to be able to integrate other models, which do not reflect all concepts of an ontology.

> Thus, a GII is best implemented with an object-oriented approach, which is capable of accessing relational, semi-structured and, of course, object-oriented repositories without a major loss for the retrieval of information. Even proprietary sources can be integrated easily.

Even though the semi-structured approach does not meet the demands of GISc today, future research might come up with an adequate model that can be used for the development of a semi-structured management system for the representation of real-world phenomena.

> Thus, research on the integration of object-orientation and semi-

> structured concepts could lead to a better implementation for GISc. For today XML will remain a technology for the exchange of information, while OEM and WebBus, being basically some kind of experiment, can contribute to a future semi-structured object system.

The implementation of a GIS shall not be encumbered by the restrictions of a programming language or developing system. Instead the language has to support the developer in thinking about the domain and solving their problems.

> Thus, Smalltalk is one of the most benefiting tools for modelling a domain, especially phenomena of the real world. The experiences of the author affirm that the development of complex applications in a short time is commonplace and nothing in particular compared to other programming languages.

The terminology in the area of management systems is partly improper, especially the definition of *data* and *object* is conflictive. While disciplines like software engineering benefited from more powerful, but less restrictive concepts, the database area often missed the opportunities.

> Thus, in the context of complex domains, terms like Database and Schema must be reconsidered. The definition of *data* is insufficient for many real-world problems and causes a prejudiced view about the things, a domain expert deals with. Modelling these domains must be the central aim of management systems again.

6 Appendix

6.1 List of Figures

6.2 List of Tables

6.3 List of Listings

6.4 List of Links

Ontology Project - http://www.geog.ucsb.edu/ good/275/v2kontology2.htm	32
PostGIS Manual - http://postgis.refractions.net/docs/	61
PostGIS Website - http://postgis.refractions.net/	61
PostgreSQL Website - http://www.postgresql.org/	62
RoyalTek - http://www.royaltek.com/	17
Smalltalk Website - http://www.smalltalk.org	10

6.5 Acronyms

The following itemises the meaning of used short names and points out their first occurrence in this thesis.

0D - Zero Dimensional	13
1D - One Dimensional	13
2D - Two Dimensional	13
3D - Three Dimensional	13
4D - Four Dimensional	13
ACID - Atomicity / Consistency / Isolation / Durability	12
ADT - Abstract Data Type	15
ANSI - American National Standards Institute	6
BLOB - Binary Large Object	15
COM - Component Object Model	14
CORBA - Common Object Request Broker Architecture	14
CPU - Central Processing Unit	55
DBMS - Database Management System	7
DBS - Database System	1
DB - Database	7
DDL - Data Definition Language	7
DGPS - Differential Global Positioning System	18
DML - Data Manipulation Language	7
DOM - Document Object Model	22

DTD - Document Type Definition	21
DVS - Management of Data Chair	1
FD - Functional Dependency	42
GII - Geographic Information Infrastructure	2
GISc - Geographic Information Science	4
GIS - Geographic Information System	4
GML - Geography Markup Language	25
GPS - Global Positioning System	4
GTM - Geo-Information Terrain Model	32
HDD - Hard Disk Drive	50
ID - Identifier	22
ISAM - Indexed Sequential Access Method	50
ISO - International Organization for Standardization	14
LOREL - LOREL Query Language	64
LORE - Lightweight Object REpository	5
MVD - Multivalued Dependency	42
NMEA - National Marine Electronics Association	56
OEM - Object Exchange Model	7
OGC - Open Geospatial Consortium	1
OMS - Object Management System	4
OODBS - Object-Oriented Database System	11
OO - Object-Oriented	47
OS - Operating System	55
PDA - Personal Digital Assistant	4
PPS - Precise Positioning Service	18
RAM - Random Access Memory	51
RDBS - Relational Database System	27
RDB - Relational Database	51
REL - Relational	47

RS - Reference System 15

SFS - Simple Features Specification 1

SGML - Standard Generalized Markup Language 20

SPARC - Standard Planning and Requirement Committee 6

SPS - Standard Positioning Service 18

SQL - Structured Query Language 14

TIN - Triangular Irregular Network 37

TUC - Chemnitz University of Technology 1

UI - User Interface 56

VSAM - Virtual Storage Access Method 50

W3C - World Wide Web Consortium 20

WKB - Well Known Binary 15

WKT - Well Known Text 15

XML - Extensible Markup Language 7

6.6 References

Lars Arge, Mark de Berg, Herman J. Haverkort, and Ke Yi. The Priority R-Tree: A Practically Efficient and Worst-Case Optimal R-Tree. In Weikum et al. [2004].

Kent Beck. *Extreme Programming Explained*. Addison Wesley Longman, Inc., 1999.

Ronald Bourret. XML and Databases, 2004. URL `http://www.rpbourret.com/xml/XMLAndDatabases.htm`.

K. Brassel and H. Kishimoto, editors. *Proceedings of the 4th International Symposium on Spatial Data Handling*, 1990.

P. Bresnahan, E. Corwin, and D Cowen, editors. *Proceedings of the 5th International Symposium on Spatial Data Handling*, 1992.

Peter A. Burrough and Rachael A. McDonnel. *Principles of Geographical Information Systems*. Oxford University Press, 1998.

Robert Burtch, Chuck Drinnan, Craig Gooch, Robert Moore, Tim Nyerges, Billie Swenson, and Connie Wester. Multi-Purpose Geographic Database Guidelines for Local Governments, 1988.

Sophi Cluet and Tova Milo, editors. *Proceedings of the 2nd International Workshop on the Web and Databases*, 1999. ACM Press.

Edgar F. Codd. A relational model of data for large shared data banks. *Commun. ACM*, 13(6):377–387, 1970.

Adrian Cuthbert. OpenGIS: Tales from a Small Market Town. In Včkovski et al. [1999].

Michael N. DeMers. *Fundamentals of Geographic Information Systems*. John Wiley & Sons, Inc., 2000.

Eric Evans. *Domain-Driven Design*. Addison Wesley Longman, Inc., 2004.

Gunar Fiedler. Indexierung semistrukturierter Daten, 2002. URL `http://archiv.tu-chemnitz.de/pub/2002/0121/data/DA.pdf`.

FRP 2001. Federal Radionavigation Plan, 2001. URL `http://www.navcen.uscg.gov/pubs/frp2001/FRP2001.pdf`.

Adele Goldberg and David Robson. *Smalltalk-80: The Language and its Implementation*. Addison Wesley Longman, Inc., 1983.

Roy Goldman, S. Chawathe, A. Crespo, and Jason McHugh. A Standard Textual Interchange Format for the Object Exchange Model (OEM). Technical report, Stanford University Database Group, 1996. URL `http://www-db.stanford.edu/lore/pubs/oemsyntax.pdf`.

Roy Goldman, Jason McHugh, and Jennifer Widom. From Semistructured Data to XML: Migrating the LORE Data Model and Query Language. In Cluet and Milo [1999]. URL `http://www-db.stanford.edu/lore/pubs/xml.pdf`.

Roy Goldman, Jason McHugh, and Jennifer Widom. LORE: A Database Management System for XML. *Dr. Dobb's Journal*, 25(4):76, 78–80, 2000. URL `http://www.ddj.com/documents/s=886/ddj0004i/`.

Michael Goodchild and Sucharita Gopal. *The Accuracy of Spatial Databases*. Taylor & Francis, Inc., 1989.

Richard Grob. *Geschichte der Schweizerischen Kartographie*. Kümmerly und Frey, 1941.

Andreas Heuer. *Objektorientierte Datenbanken: Konzepte, Modelle, Standards und Systeme*, volume 2. Addison Wesley Longman, Inc., 1997.

Eduard Imhof. *Gelände und Karte*. Eugen Rentsch Verlag, 1950.

Eduard Imhof. *Kartographische Geländedarstellung*. Walter De Gruyter & Co., 1965.

ISO 8879:1986. *ISO 8879:1986 Information processing. Text and office systems. Standard Generalized Markup Language (SGML)*. International Organization for Standardization.

H. V. Jagadish and Inderpal Singh Mumick, editors. *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 1996. ACM Press.

W. Jordan and C. Reinhertz. *Handbuch der Vermessungskunde*. J.B. Metzlersche Buchhandlung, 1910.

Gail Langran. *Time in Geographic Information Systems*. Taylor & Francis, Inc., 1992.

Robert Laurini and Derek Thompson. *Fundamentals of Spatial Information Systems*. Academic Press Limited, 1992.

Paul A. Longley, Michael F. Goodchild, David J. Maguire, and David W. Rhind. *Geographical Information Systems: Management Issues and Applications*, volume 2. John Wiley & Sons, Inc., 1999a.

Paul A. Longley, Michael F. Goodchild, David J. Maguire, and David W. Rhind. *Geographical Information Systems: Principles and Technical Issues*, volume 1. John Wiley & Sons, Inc., 1999b.

Paul A. Longley, Michael F. Goodchild, David J. Maguire, and David W. Rhind. *Geographic Information Systems and Science*. John Wiley & Sons, Inc., 2001.

Duane F. Marble, editor. *Proceedings of the Third International Symposium on Spatial Data Handling*, 1988.

David Martin. *Geographic information systems and their socioeconomic applications*. Routledge, 1991.

Ian Masser and Michael Blakemore, editors. *Handling geographical information: methodology and potential applications*. Longman Scientific & Technical, 1991.

Gregory McFarland, Andres Rudmik, and David Lange. *Object-Oriented Database Management Systems Revisited: An Updated DACS State-of-the-Art Report*. Air Force Research Laboratory - Information Directorate (AFRL/IF), 1999. URL `http://www.dacs.dtic.mil/techs/oodbms2/`.

Jason McHugh, S Abiteboul, Roy Goldman, D. Quass, and Jennifer Widom. LORE: A Database Management System for Semistructured Data. *SIGMOD Record*, 26(3):54–66, 1997. URL `http://www-db.stanford.edu/lore/pubs/lore97.pdf`.

Jean-Claude Müller, Jean-Philippe Lagrange, and Robert Weibel, editors. *GIS and Generalization: Methodology and Practice*, volume 1 of *GISDATA Series*. Taylor & Francis, Inc., 1995.

Beng Chin Ooi. *Efficient Query Processing in Geographic Information Systems*, volume 471 of *Lecture Notes in Computer Science*. Springer Verlag, 1990.

OpenGIS® Feature Geometry. The OpenGIS® Abstract Specification: Topic 1 - Feature Geometry, 2001. URL `http://www.opengeospatial.org/docs/01-101.pdf`.

OpenGIS® GML. OpenGIS® Geography Markup Language (GML) Implementation Specification, 2003. URL `https://portal.opengeospatial.org/files/?artifact_id=7174`.

OpenGIS® SFS for Corba. OpenGIS® Simple Features Specification for Corba: Revision 1.0, 1998. URL `http://www.opengeospatial.org/docs/99-054.pdf`.

OpenGIS® SFS for SQL. OpenGIS® Simple Features Specification for SQL: Revision 1.1, 1999. URL `http://www.opengeospatial.org/docs/99-049.pdf`.

PostGIS Manual. *PostGIS Manual*. PostGIS. URL `http://postgis.org/docs/`.

D. Quass, Jennifer Widom, Roy Goldman, K. Haas, Q. Luo, Jason McHugh, S. Nestorov, A. Rajaraman, H. Rivero, S. Abiteboul, J. Ullman, and J. Wiener. LORE: A Lightweight Object REpository for Semistructured Data. In Jagadish and Mumick [1996]. URL `http://www-db.stanford.edu/lore/pubs/lore-demo.pdf`.

Jonathan Raper. *Multidimensional Geographic Information Science*. Taylor & Francis, Inc., 2000.

D. Richardson and P. van Oosterom, editors. *Proceedings of the 10th International Symposium on Spatial Data Handling*, 2002. Springer Verlag.

Hanan Samet. *Applications of Spatial Data Structures: Computer Graphics, Image Processing and GIS*. Addison Wesley Longman, Inc., 1990a.

Hanan Samet. *The Design and Analysis of Spatial Data Structures*. Addison Wesley Longman, Inc., 1990b.

Markus Schneider. *Spatial Data Types for Database Systems: Finite Resolution Geometry for Geographic Information Systems*. Lecture Notes in Computer Science: 1288. Springer Verlag, 1997.

Emmanuel Stefanakis. Representation of Map Objects with Semi-structured Data Models. In Richardson and van Oosterom [2002], pages 547–562.

Wolfgang Torge. *Geodesy: 3rd Edition*. Walter de Gruyter & Co, 2001.

H.T.J.A. Uitermark. *Ontology-Based Geographic Data Set Integration*. 2001. URL `http://purl.org/utwente/fid/1367`.

Marc van Kreveld, Jürg Nievergelt, Thomas Roos, and Peter Widmayer, editors. *Algorithmic Foundations of Geographic Information Systems*, volume 1340 of *Lecture Notes in Computer Science*, 1997. Springer Verlag.

Andrej Včkovski, Kurt E. Brassel, and Hans-Jörg Schek, editors. *Interoperating Geographic Information Systems, Second International Conference, INTEROP'99 Zurich, Switzerland, 1999, Proceedings*, volume 1580 of *Lecture Notes in Computer Science*, 1999. Springer Verlag.

XML 1.1. *Extensible Markup Language (XML) 1.1*. W3C. URL `http://www.w3.org/TR/xml11/`.

Monica Wachowicz. *Object-Oriented Design for Temporal GIS*. Taylor & Francis, Inc., 1999.

Gerhard Weikum, Arnd Christian König, and Stefan Dessloch, editors. *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 2004. ACM Press.

Paul Werkmeister. *Lexikon der Vermessungskunde*. Herbert Wichmann Verlag, 1943.

Wikipedia. Wikipedia: The Free Encyclopedia. URL `http://www.wikipedia.org/`.

Heinz Wittke. *Geodätische Briefe: Ein neuzeitliches Selbststudium der Vermessungstechnik*. GEOS Verlag, 1954.

Mike Worboys. A Model for Spatio-Temporal Information. In Bresnahan et al. [1992], pages 602–611.