Specifying Geographic Information - Ontology, Knowledge Representation, and Formal Constraints

Christensen, Jesper Vinther; Jacobi, Ole; Bjørner, Dines; Nilsson, Jørgen Fischer; Frederiksen, Poul

Publication date:
2007

Document Version
Publisher's PDF, also known as Version of record

Specifying Geographic Information

- Ontology, Knowledge Representation, and Formal Constraints

Jesper Vinther Christensen

Kongens Lyngby 2007
IMM-PHD-2007-178

Technical University of Denmark
Informatics and Mathematical Modelling
Building 321, DK-2800 Kongens Lyngby, Denmark
Phone +45 45253351, Fax +45 45882673
reception@imm.dtu.dk
www.imm.dtu.dk

IMM-PHD-2007-178
ISSN 0909-3192
ISBN 87-643-0148-4

Summary

This thesis deals with the specification of geographic information. On the basis of the role of geographic information as an infrastructure element, a method is developed for the making of specifications which are well-structured and ensure the connection between the data collections being part of a joint infrastructure.

The motivation for the presented work is to meet the need for topical geographic information at any time, so that the requirements for data content and quality are fulfilled, and the information can thus form actively part of the task performance in public administration as well as in the private sector.

The theoretical background is the establishment of a representational system, which ontologically comprises a representation of notions in the "real world" and notions which include the representation of these. Thus, the thesis leans towards a traditional division between modeling of domains and conceptualization of these. The thesis contributes a formalization of what is understood by domain models and conceptual models, when the focus is on geographic information. Moreover, it is shown how specifications for geographic information are related to this representational system.

The starting point of the thesis is an analysis mapping the elements in a specification for geographic information. The basis of this empirical investigation is TOP10DK's data content specification, version 3.2 of the National Survey and Cadastre. The basic idea is to view a specification as a collection of requirements and rules, building on terms from the domain and concept ontologies.

In combination with the theoretical basis the analysis is used for developing an underlying model of notions, which defines the individual elements in a specification and the relations between them. In the chapters of the thesis this underlying model is extended to include a number of components, which each contribute to the model being able to form the basis of a strong and productive specification tool for the making and maintenance of specifications for geographic information. These components among others include description of quality requirements and formalization of rules, so that they can be used for verification of produced information.

An essential contribution is a formal specification language dedicated to the formulation of formal rules to be observed by the information. The language is based on a formal semantic model which makes translation into other languages possible. In the thesis it is shown how statements can be translated into SQL and thus form the basis of direct implementation in the production environments where the geographic information is procured.

To be able to describe requirements for the quality of geographic information is an essential part of a specification. The thesis contributes a structure of quality descriptions by introducing two notions: "Acceptable Quality Levels" (AQL) and "Quality Element Requirements" (QER), which designate respectively the minimum quality requirements for information produced according to a given specification and the requirements for the quality parameters used to describe this information. The two notions are incorporated and related to the developed system of notions for specification for geographic information.

It is an important part of an infrastructure for geographic information that there is a connection between the individual data collections. This thesis argues for ensuring the connection by first and foremost describing these as an integrated part of the specification work. The thesis contributes a model which describes relations and dependencies by writing specifications in the context of one or more other specifications.

As an illustration of the applications of specifications written in the developed specification language, a concept is developed in the thesis to make possible a decentralized collection and distribution of information about changes to be used for updating geographic information.

Resumé

Denne afhandling omhandler specifikation af geografiske informationer. Med udgangspunkt i geografiske informationers rolle som infrastrukturelement udvikles en metode til udarbejdelse af specifikationer, der er velstrukturerede og sikrer sammenhæng mellem de datasamlinger, som indgår i en fælles infrastruktur.

Motivationen for det fremlagte arbejde er at imødekomme behovet for en til enhver tid aktuelle geografiske informationer, således at kravene med hensyn til dataindhold og kvalitet opfyldes, og informationerne dermed kan indgå aktivt i opgavevaretagelsen i såvel den offentlige administration som i den private sektor.

Det teoretiske grundlag tager udgangspunkt i opstillingen af et repr{aesentationssystem, som ontologisk omfatter repræsentationen af begreber i den "virkelige verden" og begreber, som omfatter repræsentation af disse. Dermed lægger afhandlingen sig op ad traditionel opdeling mellem modellering af domæner og konceptualisering af disse. Afhandlingen bidrager med en formalisering, af hvad der forståes ved domænemodeller og konceptuelle modeller, når fokus er på geografiske informationer, samt hvordan specifikationer for geografiske informationer forholder sig til dette repræsentationssystem.

Udgangspunktet for afhandlingen er en analyse, der kortlægger elementerne i en specifikation for geografiske informationer. Grundpillen i denne empiriske undersøgelse er TOP10DK's dataindholdsspecifikation ver. 3.2. fra Kort & Matrikelstyrelsen. Grundideen er at anskue en specifikation som en samling af krav og regler, der bygger på termer fra domæne- og konceptontologierne.

Analysen kombineret med det teoretiske fundament anvendes til at udvikle en grundlæggende begrebsmodel, som definerer de enkelte elementer i en specifikation samt relationerne mellem disse. Denne grundlæggende model udvides i afhandlingens kapitler til at omfatte en række komponenter, der hver for sig bidrager til, at modellen kan danne grundlaget for et stærkt og produktivt specifikationsværktøj til udarbejdelse og vedligeholdelse af specifikationer for geografiske informationer. Disse komponenter omfatter blandt andet beskrivelse af kvalitetskrav og formalisering af regler, således de kan anvendes til verifikation af producerede informationer.

Et væsentligt bidrag er et formelt specifikationssprog, der er dedikeret til at formulere formelle regler, som informationer skal overholde. Sproget er baseret på en formel semantisk model, hvilket muliggør oversættelse til andre sprog. I afhandlingen er det vist, hvordan udsagn kan oversættes til SQL og dermed danne grundlag for en direkte implementering i de produktionsmiljøer, hvori de geografiske informationer tilvejebringes.

At kunne beskrive krav til kvaliteten af geografiske informationer er en væsentlig del af en specifikation. Afhandlingen bidrager med en strukturering af kvalitetsbeskrivelser ved at indføre to begreber: "Acceptable Quality Levels" (AQL) og "Quality Element Requirements" (QER), som henholdsvis betegner mindste kvalitetskrav til informationer produceret efter en given specifikation, og krav til kvalitetsparametrene til beskrivelse af disse. De to begreber er indarbejdet og relateret til det udviklede begrebsapparat for specifikationer for geografiske informationer.

En vigtig bestanddel af en infrastruktur for geografiske informationer er, at der er sammenhæng mellem de enkelte datasamlinger. Der argumenteres i denne afhandling for, at denne sammenhæng først og fremmest sikres ved, at disse beskrives som en integreret del af specifikationsarbejdet. Afhandlingen bidrager med en model, hvor relationer og afhængigheder beskrives ved, at specifikationer kan skrives i konteksten af en eller flere andre specifikationer.

Som illustration af anvendelsesmulighederne for specifikationer skrevet i det udviklede specifikationssprog, udvikles der i afhandlingen et koncept, som muliggør decentral opsamling og distribution af informationer om ændringer til brug for ajourføring af geografiske informationer.

Preface

This thesis was prepared at GeoInformatics and Informatics and Mathematical Modelling at the Technical University of Denmark in partial fulfillment of the requirements for acquiring the Ph.D. degree in engineering.

The thesis deals with different aspects of modeling and specification of geographic information. The main focus is on studying the relations among domain descriptions, requirements and the design of geographic information.

The thesis is a summary report and is based on a collection of research papers written during the period 2003–2006, and published elsewhere.

Lyngby, February 2007

Jesper Vinther Christensen

Papers Included in the Thesis

1. "A Framework for modeling Quality Requirements", Symposium for Spatial Data Handling 2004. Anders Friss-Chrisensen, Jesper Vinther Christensen, and Christian Jensen.
2. "Formalizing Constraints for Geographic Information" in proceedings of Information System Development 2005". Jesper Vinther Christensen and Mads Johnsen
3. "Specifying Geographic Information", in proceedings of AGILE 2006. Jesper Vinther Christensen.

Acknowledgements

Many people have contributed to my Ph.D. project. I would like to greatly acknowledge this support and help.

I acknowledge the financial support of the National Survey & Cadastre - Denmark. Thanks for giving me the time and opportunity to pursue my ideas.

I would like to thank my supervisors at the Technical University of Denmark: Professor Ole Jacobi, Professor Dines Bjørner and Professor Jørgen Fischer Nilsson. Ole for introducing me to the concept of geographic information and especially the issues of information quality, Dines for introducing me to formal methods and for bringing the idea of domain engineering into the project. Jørgen for encouraging me to study logics and for showing directions within the science of knowledge engineering and representation.

Also thanks to all my colleagues at the National Survey & Cadastre. Poul Frederiksen, my internal supervisor, for providing me the time and opportunity to do the project, and especially Brian Pilemann Olsen, who has always been helpful and taken his time to discuss my ideas and guide me through the difficult periods of the project, thanks Brian.

I would also like to thank Anders Friis-Christensen for the work we have done together on quality requirements and for many interesting discussions. Also thanks to Mads Johnsen for the collaboration on the paper on formulating formal constraints and his work on the HLCL translator.

My family and I spent six months at ESRI in Redlands from February to August 2003. We would like to thank all for making this stay possible. Thanks to Klaus Gerlich and Kurt Andersen at Informi GIS for spending time and resources on arranging the contact to ESRI, and to all people at ESRI making us feeling welcome. The ESRI campus is a great place to stay. Thanks to Earl Nordstrand, who let me join his team in Product Development.

Finally, I would like to thank my family and friends for thier support and encouragement. Especially to Pernille, my wife, and my two sons Mikkel and Jeppe. Without your love, dedication, and understanding, finishing this project would never have been possible.

List of Figures

2.1 Aragos approach to domain engineering 10

2.2 The two signs at the escalator. 12

2.3 Jackson's distinction between the system and the world. 13

2.4 The two signs at the escalator 19

2.5 An entity-relationship diagram exemplifying simple inheritance. . 20

3.1 Communicating geographic information perspective. 26

3.2 The principle of co-operative design 28

3.3 ER diagram for the entity types and relations included in the running example. 39

4.1 Representation model. 46

4.2 Pattern for mapping domain and conceptual models. 64

5.1 Concepts of an "and/or" goal tree. 76

5.2 Concepts of an "and/or" goal tree. 77

6.1 Quality Requirements. 87

6.2 Quality information, elements, and subelements. 88

6.3 Quality assessment. 91

7.1 Ensuring consistency using a Validation Engine. 103

7.2 Examples of topologic relations which can be modeled by the nine-intersection matrix. 107

7.3 Some topological relations between to polygons 109

7.4 Allowed and disallowed relations between buildings and residential areas. 110

7.5 Entity-relationship for a network model. 110

7.6 Entity-relationship diagram of details of the road code format. . 113

7.7 Strategy for evaluation of "inner distance". 115

7.8 Relations between two time intervals (after [Ohlbach, 2004]) . . . 116

7.9 Left: two overlapping buildings, right: two road segments with identical start and end points. 119

8.1 Basic principle for the translation process 140

Contents

Summary **i**

Resumé **iii**

Preface **v**

Papers Included in the Thesis **vii**

Acknowledgements **ix**

1 Introduction **1**

1.1 Problem Description 2

1.2 Objectives, Aims, and Hypothesis 3

1.3 Contribution 4

1.4 Reading This Thesis 6

2 Contributing Disciplines **9**

2.1 Domain Engineering 10
2.2 Knowledge Engineering and Representation 14
2.3 Requirement Engineering 21
2.4 Geographic Information 22

3 Specifying Geographic Information 25

3.1 Introduction 26
3.2 The Process of Developing Specifications 27
3.3 Specifications for Geographic Information 28
3.4 A Small Example 31
3.5 Existing Approaches to Specifying GI 34
3.6 Design Goals for GeoSML 37
3.7 Summary 41

4 Structuring Specifications for Geographic Information 43

4.1 Introduction 44
4.2 Representation System 45
4.3 Domain Model 48
4.4 Conceptual Model 57
4.5 Mapping Domain and Conceptual Models 63
4.6 Summary 70

5 Requirement Specification 71

5.1 Introduction 72

5.2 Requirement Engineering 74
5.3 Grammar for Requirement Models 78
5.4 Example of Modeling Requirements 79
5.5 Summary 81

6 Specifying Quality Requirements 83

6.1 Introduction 84
6.2 Quality of Geographic Information 85
6.3 Modeling Quality 91
6.4 An Example 97
6.5 Summary 99

7 Constraints on Geographic Information 101

7.1 Introduction 102
7.2 The Basic Idea 103
7.3 Constraints on Geographic Information 105
7.4 Summary 121

8 Formalizing Constraints on Geographic Information 123

8.1 Introduction 124
8.2 Formalizing Constraints 126
8.3 Formal Description 133
8.4 Model-theoretic Semantics 136
8.5 Translating GeoSML Constraints to SQL 139

8.6 Examples - Translation of Constraints 140

8.7 Summary 146

9 Conclusion **149**

9.1 Results 150

9.2 Overall Conclusion 153

9.3 Future Work 155

Chapter 1

Introduction

Geographic information is today seen as an infrastructure element equal to roads, railways, and water supply systems. Geographic information is an important tool for managing our physical surroundings and supporting decision making. Geographic information (GI) is created and used by many organizations, public as well as private. The produced information constitutes the backbone of the Spatial Data Infrastructures (SDI's). The efficiency of SDI's depends on how well the content and the maintenance of the basic data collections are coordinated. Models and specifications are fundamental in the effort of doing so. How these models and specifications are developed and structured is the subject of the investigations in the present thesis. The aim is to provide a framework supporting the development of specifications for geographic information which can be used both by domain experts, in this case specialists in designing and producing GI, and computer experts responsible for developing production and distribution systems. The presented work is strongly influenced by the perspective of national mapping agencies (NMA's) and other producers of geographic information on the specification and production of geographic information. The examples used to identify requirements and illustrate the developed concepts are taken from topographic mapping. This does not mean that the presented framework cannot be used to specify other types of geographic information. The concept is meant to be general and the hope is that the framework can be used in a wide varieties of applications.

1.1 Problem Description

Specifying data content and cartographic design are recognized as major challenges by developers and producers of geographic information. Today most specifications are written in natural language with a loosely defined structure and content. The specifications may be supported by a number of data and conceptual models, using UML object type diagrams, ER diagrams, or one of the languages designated for the specification of geographic information. The experiences achieved by using these formal/semi-formal approaches in combination with natural language specifications have in some cases, at least for the National Survey & Cadastre - Denmark, been somewhat disappointing. There are a number of reasons for this:

- The use of modeling languages like UML seems to result in either too abstract models with properties of metamodels like the General Feature Model (GFM) [ISO, 2004] and UML's meta-model itself, or the models becoming too complicated drowning in a large number of topologic relations and constraints.
- Ontological commitments are often implicitly given in natural language specification. Design decisions may therefore be based on unexpressed understandings of the problem domains and user requirements. This leads to specifications that are more likely to be based on traditions and experiences capturing the requirements the users once had, rather than the requirements they will have in the future.
- Natural-language-based specifications are not directly coupled with the production systems. Rules and constraints expressed in the specifications must be interpreted and translated into a programming language before they can be introduced in the production system. This approach is error prone: information may be lost or misunderstood in the translation process, which results in implementations not necessarily reflecting the intended design.
- User requirements are not expressed in the traditional specifications, and are at best described in independent documents. Thus, there is no direct and documented relation between the representation of the problem domain, the stated requirements, and the decided design.
- Describing relations and dependencies among specifications written in natural language is difficult. Moreover, to use and reuse information for multiple purposes in an efficient way, the delegation of responsibilities for creating and maintaining the various parts of an SDI must be clear and unambiguous.

- Formulating requirements related to quality is a very complex task, and most specifications today either omit quality requirements or the descriptions are scattered throughout the specifications. Therefore, a systematic approach to embedding quality requirements in specifications is needed.

Approaches to modeling geographic information already exist. In Section 3.5 the most important are introduced and discussed in relation to the above requirements. The conclusion is that none of the existing approaches meet the requirement that producers of geographic information for a specification language, and that there is a need for a new approach for developing specifications that can be used both in the production geographic information and as support in the development of spatial data infrastructures.

1.2 Objectives, Aims, and Hypothesis

1.2.1 Objectives and Aims

The motivation of this thesis is the increasing need for geographic information representing reality so that actuality, content, and structure of the information constitute a stable and solid foundation for spatial data infrastructures (SDI's) as well as a reliable platform for making decisions about our physical surroundings. The aim is to develop a specification language that meets the requirements listed in the problem definition and enables domain experts, designers of geographic data collections, and computer engineers to participate in a cooperative design process. The idea is that a language dedicated to the development of specifications will enable information providers like national mapping agencies to develop and produce new products faster, and with a design and content that are more likely to meet the users' requirements that of existing approaches. To establish a foundation of the development of this language, concepts within the domain of specifying geographic information must be clarified, which is achieved by investigating various perspectives of the specification, production, and uses of geographic information.

1.2.2 Hypothesis

The thesis is based on two dogmas, one from computer science and one from the science of geographic information. The conjunction of the two is here called

geoinformatic, a term widely recognized within the research of geographic information.

> ***The hypothesis:***
>
> ***The knowledge embedded in natural language specification for geographic information can be represented in formal computational structures.***

This hypothesis shall be seen as a vision to be followed throughout the thesis, rather than a goal defining the success criteria for the project as a whole. We will investigate the hypothesis from three different viewpoints:

(i) Ontology and knowledge representation

Specifying geographic information is a discipline deeply involving ontological consideration. The notions of classification and conceptualization are the central concepts.

(ii) The concept of geographic information specification

The second perspective concerns the identification and definition of domain concepts in the field of specifying and producing geographic information. The notions of domain and representation of domain phenomena are central in the investigation and conceptualization of specifications and production of geographic information.

(iii) Validation and consistency checking

Computer science concepts and principles are introduced and used to formalize the domain concept of specifying and producing geographic information. The concepts of domain modeling, temporal interval logic, description logic, formal semantics, and mereology are introduced and treated in relation to the domain of specifying and producing geographic information. The production of geographic information strongly depends on specifications. We will investigate how specifications can be organized to support the production process.

1.3 Contribution

This thesis contributes to the understanding of the specification of geographic information. The overall contribution is a framework, which we call the Geo-

graphic Information Specification Markup Language (GeoSML). The contribution of this thesis has four major parts:

1. A method for structuring specifications for geographic information written in natural language.
2. A clarification of the role of ontologies in specifications.
3. A framework for specifying functional requirements
4. A framework for specifying quality requirements
5. A language to formalize constraints on geographic information as a part of developing a conceptual model.

The following sections summarize each of the five contributions.

1.3.1 Structuring Specifications Written in Natural Language

This thesis contributes with a syntax for marking up statements according to the role they play in the interpretation process, where entities in a domain are represented as objects in a geographic data collection. It is suggested that specifications include three models: (i) requirement model, (ii) domain model, and (iii) conceptual model. Statements in a specification are included either in one of the three models or in the description of the relationships among these models. Thus, a bridge is built between the knowledge of how the world is perceived and the representation of this knowledge as geographic information.

1.3.2 Ontological Commitments in Specifications

An important part of a specification is to describe the part of reality that must be represented in a geographic data collection. This thesis contributes to the understanding of domain descriptions by discussing what will be called the ontological commitment of specifications. The classification and modeling of geographic entities are discussed and related to existing approaches within knowledge representation and software engineering.

1.3.3 Structuring of Functional Requirements

The work contributes with a method for structuring functional requirements which the users demand from the data collections being developed. Functional requirements are related to elements in the conceptual model and thus justify the design of the data collection.

1.3.4 Specifying Quality Requirements

The quality elements necessary for adequately describing the quality of the geographic data collection are identified and described. Furthermore, elements are added to GeoSML which support a dynamic and flexible specification of requirements related to quality assessment. The result is a framework enabling designers and users to specify requirements related to quality and include these in the design.

1.3.5 Formalizing Constraints

The role of formal constraints in relation to designing and producing geographic information is discussed, and a formal constraint language to include formal constraints in specifications of geographic information is introduced in GeoSML. It is illustrated how statements written in natural language can be translated into this formal language and implemented by automatically translating the statements into SQL.

1.4 Reading This Thesis

1.4.1 Intended Audience

The intended audience of this thesis is both persons with a scientific interest in the specification and modeling of geographic information and professionals working with the development and production of geographic information. We also hope that the method for designing geographic data collections will be appreciated and used as a source of inspiration for project managers and team members responsible for developing new data collections or remodeling existing ones. The idea of dividing specifications into three distinct and related models,

and the explanation of the role of ontologies may be especially helpful. When designing production systems and case tools supporting the development of specifications, system architects and developers may benefit from the approach to formalizing constraints and introducing these in the production environment.

1.4.2 Structure of the Thesis

The nine chapters of this thesis are organized as follows:

Chapter 1 is an introduction to the subject of the thesis, including background, hypothesis, and contributions.

Chapter 2 gives an overview of the scientific disciplines that contribute to the presented work. The overview focuses on the theories of computing science and software engineering, including requirement engineering, domain engineering, and knowledge representation. Readers familiar with the theory of software engineering may skip this chapter.

Chapter 3 introduces concepts central for the specifications of geographic information and relates these to theoretic and philosophic issues of specification, including ontology and the classification of geographic entities. Furthermore, the role of geographic information seen as a basis for decision making is discussed.

Chapter 4 introduces the basic syntax of the Geographic Information Specification Markup Language (GeoSML). GeoSML is based on a representation system, which makes a clear distinction between the description of the world to be represented a data collection, and the conceptual design of the data collection.

Chapter 5 extends GeoSML with grammars for structuring functional requirements for the data collection being developed. Requirements can be related to elements at the conceptual level and thus document and motivate design decisions.

Chapter 6 discusses the notion of quality in relation to geographic information and suggests an approach to embedding quality requirements in specifications for geographic information.

Chapter 7 introduces a classification of constraints on geographic information. Each type of constraint is treated in the context of producing geographic information, and suggestions for formalizing constraints in predicate logic are given.

Chapter 8 adds a formal constraint language to GeoSML. This language is called the High Level Constraint Language and is designed to specify formal constraints in the context of a conceptual model, which can be translated into SQL.

Chapter 9 summarizes the results of the thesis and the conclusions to be drawn from the results achieved in the presented work. Furthermore, future work and research directions are discussed.

Chapter 2

Contributing Disciplines

Abstract: This chapter introduces the research areas that have contributed to and inspired the work presented in this thesis. References to the main material are given and the notions of domain, requirement, and knowledge engineering are discussed. The main references on domain engineering are due to Jackson and Bjørner, while the approach to requirement engineering is inspired by Lambsweerde.

2.1 Domain Engineering

Domain engineering is the process of identifying and modeling domain properties and capturing the knowledge of a domain in domain models. Domain models are descriptive by nature , as they deal with designating "real-world" things and phenomena and the relations among these. Domain engineering is inherently related to domain analysis. A definition of domain analysis is formulated by Prieto-Diaz, who elucidates its purpose as

> [Domain Analysis is] ... *a process by which information used in developing software systems is identified, captured, and organized with the purpose of making it reusable when creating new systems.*[Prieto-Diaz, 1990]

The original motivation for domain engineering is software reuse [Arango, 1989]. Arango and Prieto-Diaz presented a model of domain analysis summarized in the following SADT diagram:

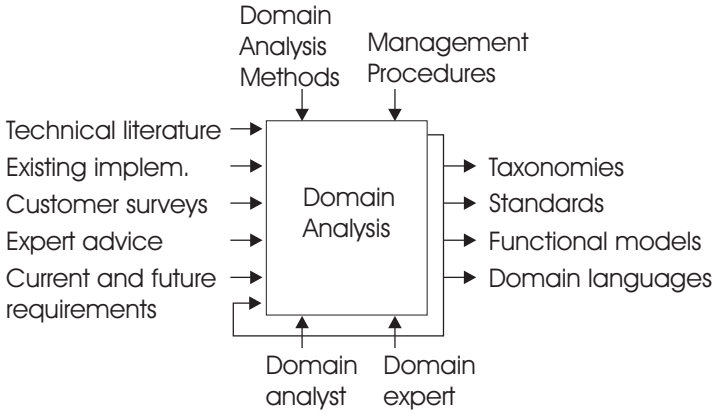

Figure 2.1: Aragos approach to domain engineering

This model describes domain analysis as an activity that takes multiple sources of input and produces many different kinds of output. The model shall be seen as an abstract paradigm for domain engineering, where a number of techniques and methods can be applied depending on the problem concerned. For example is the domain analysis method a parameter in the model. Domain analysis methods cover a variety of methods, e.g structured analysis, Jackson's JSD,

RAISE, and object-oriented analysis and development. Domain knowledge from relevant sources is used as a background to gathering domain knowledge and structuring this according to the selected paradigm. This is done, among others, by domain experts and analysts. The results of domain analysis are taxonomies, domain processes, standards, and logical architectures.

Domain models are a prerequisite to stating requirements for systems. Without an established understanding of the domain in which a system will operate, requirements will be loosely formulated and unrelated to the problems in the domain that the system being developed should seek to solve. During modeling it is important to bear in mind that the word "model" has at least to meanings in relation to system specification and design. It is important two distinguish the two meanings from each other.

The first sense of model is to consider a model as a description of a part of the world, e.g. a set of differential equations describing the movement of ground water or a set of classes and relations describing the ownership of properties and condition influencing the taxation of these. This kind of models is analytic by nature. The second type of models is called analogic or sometimes iconic [Ackoff, 1962]. A map is an example of an analogic model, it consists of objects representing real world phenomena. The specification for geographic information is an example of an analytic model. Such models are investigated in the present thesis.

Domain analysis is equally important to the specification of geographic information and to software engineering. One of the major contributions from this thesis is a paradigm for structuring domain knowledge, hence GeoSML includes methods for organizing the knowledge needed to design geographic information. While the motivation for domain modeling within software engineering is the possible reuse of software components, one of the arguments for creating domain models has been the need for combining geographic information from several sources, which is also denoted the capability of interoperability of geographic information. However, seen in relation to the specification and production of geographic information, domain models are motivated by the need to specify the requirements for the content of a geographic data collection. Here, domain models are developed to achieve an in-depth understanding of the domain the information will be representing [Worboys, 1995, Molenaar, 1998].

It is a characteristic of the process of domain engineering that the knowledge of the designers of the problem domain increases over time. From being weak ideas, growing into stable requirements, and finally turning into a design which can be used as the basis of an implementation. The first version of a specification may be a rough sketch, written in a natural language, which in the following process is enhanced and expanded, resulting in a precise and formal specification.

Formal specifications are necessary if a precise description of a domain should be achieved. The following example is from Jackson [Jackson, 1995] and is about two signs he once saw at the foot of an escalator in an airport. The example illustrates very well the reasons for a structured and formal approach of describing domains.

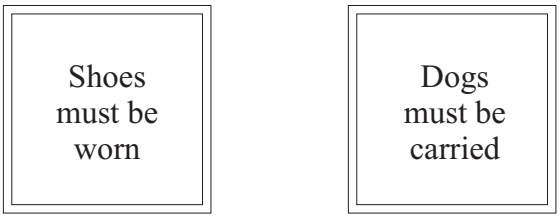

Figure 2.2: The two signs at the escalator.

Jackson asks the question: "What do they mean?" Must the two signs be interpreted in the same way, so that if you want to use the escalator you must wear a pair of shoes and carry a dog, leading to restrictions on who is qualified to use the escalator. Another interpretation could be that if you want to use the escalator and bring a dog, then the dog must be carried, and analogously for shoes, if you want to use the escalator and bring a pair of shoes, then they must be worn. Reasonable enough, but what if you just bought two pair of shoes, must they all be worn, which seems impossible?

Formalization of rules and constraints is a strong tool for clarifying intended meanings. When it comes to describing domains, requirements, and writing data content specifications for geographic information formalizations can be a useful tool. Formal rules and constraints a more likely to be unambiguous than if written in natural-language. In predicate logic the meaning properly intended in the above example can be expressed like this:

$$\forall\ x\ (IsPerson(x) \wedge OnEscalator(x) \rightarrow \exists\ y\ (PairOfShoe(y) \wedge IsWearing(x,y))$$

"For all x where x is a person and is on an escalator they must wear a pair of shoes", or in the formal constraint language introduced in this thesis (see Chapter 8): "all person using escalator must wear shoe".

The meaning of the rule on the second sign could be formalized like this:

$$\forall\ x\ (OnEscalator(x) \wedge IsDog(x) \rightarrow IsCarried(x))$$

"for all x where x is on an escalator and is a dog must be carried", or in the mentioned constraint language: "all dog on escalator must be carried"

2.1.1 Jackson's World-Machine Distinction

Jackson has contributed to the understanding of the process of system development for the past four decades, and he has published a large number of papers and books on the subject. The books "Software Requirements & Specifications – A lexicon of principles, practices and prejudices" [Jackson, 1995] and "Problem Frames: Analysing and Structuring Software Development Problems" [Jackson, 2001] summarize the most important contributions.

Jackson makes a clear distinction between the real world and the system and thus defines the notion of domains. The domain is a part of reality in which the problem exists, and the system is the solution to the problem. Domain engineering is a prerequisite to requirement engineering.

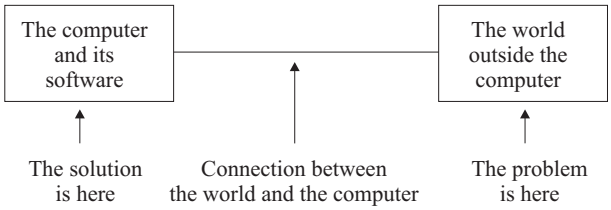

Figure 2.3: Jackson's distinction between the system and the world.

Jackson introduces the notions of *designations*, *definitions*, and *refutable assertions*.

Designations are identification and description of what Jackson calls the *ground terms*. Ground terms are terms that fix the relationship between the description and what is described. Ground terms form the basis of defining other terms.

Definitions add new terms to the domain description. Defined tems are defined on the basis of terms previously designated or previously formally defined terms. Formal definitions add no new knowledge of the description of a domain, but give easy access to complex concepts and thus provide a more convenient terminology for saying what we could have said less conveniently without them.

Assertions relate designated and defined terms by stating rules that constrain and explain the interaction among domain entities. Jackson emphasizes that

assertions must be *refutable*. He claims that if an assertion about a domain cannot be questioned, then nothing important has been said about the domain.

2.1.2 Bjørner: Triptych Approach

Bjørner's approach to domain modeling is to identify the fundamental *principles* of a domain. Bjørner calls these principles the *domain facets*, which cover among others: the intrinsics, the enterprize, the processes, the technology support, the management, and the rules and regulations. Throughout his research Bjørner investigates the methods for software engineering. A method is according to Bjørner *"a set of principles for selecting and applying techniques and tools in order efficiently to construct an artifact"* [Bjørner, 2006a].

Bjørner's method, which is called Triptych (from the Greek tri- "three" + ptyche "fold"), is a semi-formal approach to software engineering, where the methods and principles needed to design efficiently artifacts are divided into three: Domain engineering, requirement engineering, and software design. An introduction to Bjørners paradigm is found in the essay "What is a Method?" [Bjørner, 2003], and the complete theory can be found in *Software Engineering*, a three-volume book on system development and formal methods [Bjørner, 2006b]. Bjørner has applied the method to a number of domains, e.g. sustainable development [Bjørner, 1999], railways [Bjørner, 2004], and air traffic control [Bjørner, 1995], to mention a few.

Bjørner's approach draws on the theories introduced by Jackson. However, where Jackson methodologically is primarily based on textual descriptions, Bjørner intensively uses formal specification languages, primarily the RAISE specification language [George et al., 1992].

2.2 Knowledge Engineering and Representation

According to Sowa knowledge representation is a multidisciplinary subject which applies theories and techniques from three different fields [Sowa, 2000]:

1. *Logic* provides the formal structure and rules of inference
2. *Ontology* defines the kind of things in the application domain
3. *Computation* supports the applications that distinguish representation from pure philosophy

Representing knowledge is vital for the specification of most computer-based systems, and also for the design of geographic information. Domain knowledge and representation of domain knowledge are prerequisites to designing geographic information. Abstract representations of reality as geographic information, which project the user's perception of the domain entities, depend on an in-depth understanding of the domain the information seeks to represent.

Knowledge is assertions about individuals and phenomena and constitutes theories about abstract relationships among concepts, based on observations of reality, processed and interpreted by humans. Knowledge representations can, following [Davis et al., 1993], play five roles in the design and implementation of knowledge based system.

1. A knowledge representation is most fundamentally a surrogate, a substitute for the thing itself, used to enable an entity to determine consequences by thinking rather than acting, i.e. by reasoning about the world rather than taking action in it.

2. It is a set of ontological commitments, i.e. an answer to the question: In what terms should I think about the world?

3. It is a fragmentary theory of intelligent reasoning, expressed in terms of three components: (i) the fundamental conception of intelligent reasoning of representation; (ii) the set of inferences the representation sanctions; and (iii) the set of inferences it recommends.

4. It is a medium for pragmatically efficient computation, i.e. the computational environment in which thinking is accomplished. A contribution to this pragmatic efficiency is supplied by the guidance a representation provides for organizing information so as to facilitate the making of the recommended inferences.

5. It is a medium of human expression, i.e. a language in which we say things about the world.

The context in which this thesis uses knowledge representation resembles points 2 and 5. Developing specifications for geographic information is, seen from a knowledge engineering perspective, to describe a part of reality, as stated above.

Some steps in the semantic spectrum include the following:

2.2.1 Methods for Representing Knowledge

Several systems and languages have been suggested for knowledge representation, each with syntax and semantics designed for a particular application. Even though these formalisms are different in approach and syntax, they have common properties: they concern concepts and relationships among concepts. Concepts are called classes, entity sets, or object types, and relationships are called roles, relation, or slot depending on approach.

Traditionally, three meanings have been defined for binary relationships: associations, taxonomic and mereologic relations. *Mereologic relations* concern "part-whole" relations, e.g. a wall is part of a house and implies a strong dependency among the entities participating in the relation. *Association* indicates a weaker relation among the participating entities than for mereologic relations, hence they indicate a relation between two independent entities, e.g. that person owns that house. *Taxonomic* relations concern "is-a" relations, e.g. a car is a vehicle. To be more express full, different semantics can be applied to a relationship. A relationship is transitive if it exists between a and b, and between b and c, which also means that it exists between a and c, e.g. the "decent-from" relation. A relationship can be defined to be the inverse of another, e.g. "son-of" is inverse of "father-of. A relation can be antisymmetric meaning that if it exists between a and b, then it cannot exit between b and a, e.g the "father-of" relation. A relation can be symmetric meaning that if it exists between a and b, then it also exists between b and a, e.g. the "is-sibling-of' relation. Some formalisms also provide relationship inheritance ("is-a" relationships among relations), e.g. the relation between the "father-of" relation and the "parent-of" relation

Mathematically, relationships can be represented as binary predicates formulated in the context of set theory, and the semantic for the various kinds of relations can be explained by defining relations among sets. The following definitions is inspired by [Wikipedia, 2006, Smith, 2004].

An association between two concepts, e.g. PERSON and BUILDING, can be regraded as a binary relation R and defined as a triple (X, Y, G), where X and Y are arbitrary sets (X can symbolize a set of persons and Y a set of buildings), and G is a subset of the cartesian product X $\times$ Y. The sets X are called the domain of the relation and the sets Y are called the codomain or range of the relation. G is called the graph of the relation. Usually, a relation is denoted xRy or R(x,y), which is read as x is R-related to y. G can for example stand for the *own* relations that may hold between a person and a building.

Special kinds of binary relations can be defined by introducing axioms to the binary relation. For example a binary relation R over a set X is said to be transitive if it holds for all x, y, and z in X, that is if x is related to y and y is related to z, then x is related to z:

$$\forall\ x,y,z \in X\ (R(x,y) \wedge R(y,z) \rightarrow R(x,z))$$

A reflexive relation R on a set X is one where it for holds for all x in X that x is R-related to itself:

$$\forall\ x \in X \rightarrow R(x,x)$$

A binary relation R on a set X is irreflexive if it holds for all x in X, that x is never R-related to itself:

$$\forall\ x \in X \rightarrow \neg\ R(x,x)$$

A binary relation R on a set X is symmetric if it holds for all x and y in X that if x is related to y then y is related to x:

$$\forall\ x,y \in X\ (R(x,y) \rightarrow R(y,x))$$

A binary relation R on a set X is antisymmetric if it holds for all x and y in X that if x is related to y and y is related to x, then x and y are the same object:

$$\forall\ x,y \in X\ (R(x,y) \wedge (x,y) \rightarrow x{=}y))$$

The *is-a* or taxonomic relation can set-theoretically be defined by using set inclusion. If an is-a relation exists between two concepts, it means that all instances of the first concept are also instances of the second concept, and it is said that the set of instances of the first concept is a subset of the set of instances of the second concept.

$$\text{is-a}(A,B) = \forall\ x\ (\text{inst}(x,A) \rightarrow \text{inst}(x,B))$$

The *part-of* relation is semantically more difficult to explain than other binary relations, and has extensively been a subject of discussion in the literature (see [Varzi, 1996, Smith, 1996, Lambrix, 2000, Eir, 2004]). It has been suggested that the part-whole relation *P(x,y)* can be regarded as a binary relation where P is reflexive, antisymmetric, and transitive [Varzi, 1996].

Barry Smith has introduced a formal framework for describing *part-whole* relationships, which he calls mereotopology [Smith, 1996]. In [Smith, 2004] he suggest the following definitions of part-whole relations:

$$part\_for(A,B) = \forall\ x(inst(x,A) \rightarrow \exists\ y\ (inst(y,B) \wedge part(x,y)))$$

The above statement gives the information that instances of A only exist if there is an instance of B in which the instance of A is contained.

$$has\_part(B,A) = \forall\ y\ (inst(y,B) \rightarrow \exists\ x\ (inst(x,B) \wedge part(x,y)))$$

which states that instances of B only exist if there is an instance of A in which the instance of B is contained.

The need for precision in modeling and representing geographic entities has an influence on how expressive the chosen modeling language should be. The challenge is to find a balance between being able to express the wanted properties of a domain and not introducing a language that will be too difficult to apply in "real world" projects. The modeling language introduced in the thesis focuses on the usability, rather than on being capable of expressing all possible details of a domain. The language elements in our language primarily build on traditional entityrelationship diagrams (ER-diagrams) and and the formal constraint language introduced in chapter 8 is partly inspired by description logics, but without description logics reasoning facilities. The next two sections will introduce the syntax of these two approaches for knowledge representation.

2.2.1.1 Entity-Relationship Diagrams

One of the simplest approaches to modeling concepts and their mutual relationships is the entity-relationship model, which was introduced by Chen in 1976 [Chen, 1976] and extended by several other authors, e.g. the extended entity-relationship (EER) approach [Elmasri et al., 1985, Engels et al., 1992], and the spatial-temporal entity-relationship modeling language (STER) [Tryfona and Jensen, 1999].

Entity sets are drawn as a rectangle with names referring to the individuals included in the set, e.g. Person, House, Road or Forest. A relationship captures how two entities or concepts are related to one another. Chen introduced ER as a language for modeling the conceptual content of databases, and he included only simple binary relationships between entity types, which he suggested to be drawn as diamond shapes with the name designating the relationship between two (or the same) entity sets, e.g. own, work for, father of, and passed course (figure 2.4B). [Engels et al., 1992] extend ER with a syntax for taxonomic relations (Figure 2.4B), and mereologic relations (Figure 2.4C).

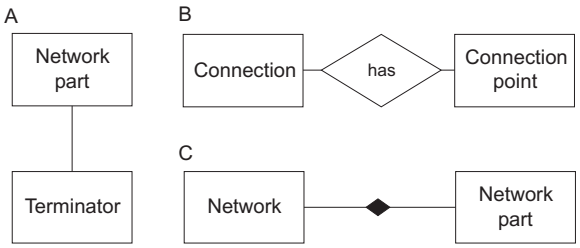

Figure 2.4: The two signs at the escalator

2.2.1.2 Description Logic

The role of ontology in a computer science perspective is to formalize structures that specify conceptualizations of specific domains. An ontology can be regarded as a controlled vocabulary that describes objects and their mutual relations in a formal way, and a grammar for combining the vocabulary terms to meaningful statements which make assertions about the domain of interest. Ontologies can include glossaries, taxonomies and thesauruses, but normally have greater expressivity and stricter rules than these tools. Formal ontologies are based on description logics [Baader et al., 2003], which are subsets of first order predicate logic. Several languages has been suggested to enhance the usability of description logics. The most prominent are oil+daml [D. Connolly et al., 2001] and OWL [Smith et al., 2004].

$$
\begin{array}{rrl}
\text{C,D} \rightarrow & \text{A} \mid & \text{(atomic concept)} \\
& \top \mid & \text{(universal concept)} \\
& \bot \mid & \text{(bottom concept)} \\
& \neg\ \text{C} \mid & \text{(concept negation)} \\
& \text{C} \sqcap \text{D} \mid & \text{(concept conjunction)} \\
& \text{C} \sqcup \text{D} \mid & \text{(concept disjunction)} \\
& \forall\ \text{R.C} \mid & \text{(value restriction)} \\
& \exists\ \text{R.C} \mid & \text{(existential quantification)}
\end{array}
$$

The above syntax introduces the minimum description logic of practical interest [Baader et al., 2003], the attributive language $\mathcal{AL}$. In the syntax A is an atomic concept, C and D concept descriptions, and the letter R stands for atomic roles. $\mathcal{AL}$ can be extended to be more expressive, e.g. union of concepts, full existential quantification, number restrictions, and negations (see [Baader et al., 2003] for details).

2.2.2 The Relations between ER and DL

Even though entity-relationship diagrams and description logics are two different approaches to knowledge representation, they do have semantic similarities. In general description logics are more expressive than the entity-relations approaches. Largely speaking the are two differences between description logics and entity-relationship: Descriptions logics support reasoning and negations, which entity-relationship diagrams does not. Furthermore, description logics are often used to develop knowledge bases on the open world assumption, which means that all facts must be introduced explicitly in the knowledge base.

The two following examples illustrate the corresponding between ER-diagrams and Description Logics.

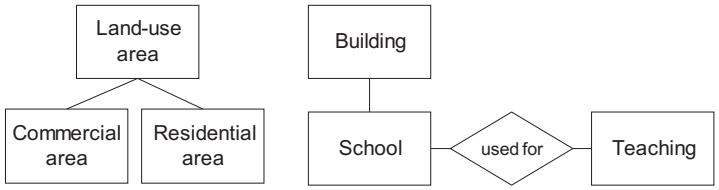

Figure 2.5: An entity-relationship diagram exemplifying simple inheritance.

The left diagram at Figure 2.5 illustrates a simple inheritance relation, which can be captured by description logic as:

$$Land - use\ area \equiv Commercial\ area \cup Residential\ area$$

The right diagram at Figure 2.5 illustrates an inheritance relation restricting the "child" entity class by require that the members of school must have at least one relation to the teaching entity type through the "usedfor" relation. This can be captured by description logic as:

$$School \equiv Building \cup \exists usedfor.Teaching$$

A general discussion on conceptual modeling using description logics is found in Chapter 10 in [Baader et al., 2003] and [Goodwin, 2005] exemplify the use of description logics within the domain of geographic information.

2.3 Requirement Engineering

Requirement engineering is a branch of computing science that deals with collecting and organizing functional and structural requirements for computer systems. The nature of requirement engineering is prescriptive, hence it deals with future systems, in contradiction to domain engineering, which is deals with capturing knowledge of real world phenomena and therefore is of a descriptive nature. A requirement is the effect in the problem domain that the users want the system, in this case a geographic data collection, to guarantee [Jackson, 2001]. Requirements are in this way the functionality to which the geographic information used in an appropriate geographic information system must lead.

The aim of requirement engineering is to identify the boundaries of the system, the stakeholders, and what sort of problems the system should solve. A number of techniques and methods for supporting requirement engineering have been suggested. Examples are the use-cases within the UML framework [Booch et al., 1999, Cockburn, 2000], Rollands CREWS, which is a scenario-based approach
[Rolland and Achour, 1998] and [Sutcliffe and Minocha, 1998], and the goal-oriented approach KAOS[1] suggested by Lamsweerde [van Lamsweerde et al., 1991].

In this thesis the KAOS approach to modeling and structuring goal-oriented requirements is preferred, and the following section introduces the approach

[1] KAOS is an acronym for Knowledge Acquisition in autOmated Specification.

adopted in KAOS.

KAOS is a semi-formal approach where requirements are formulated as goals that can be formalized and used to ensure that the implemented system meets the requirements. Goals can be related by a *and/or trees*. The most general goals are situated nearest the top of the tree and more specific requirements are situated at the bottom. Thus, goal trees will be organized so that subrequirements state how their parent requirement should be fulfilled, and vice versa the parent requirement explains why its sub-requirements must be met. The innovative idea of KAOS is to assign the responsibility of implementing goals to objects, which can either be agents, entities, events, or relationships.

2.4 Geographic Information

This section briefly describes the fundamental aspects of geographic information, to emphasize that geographic information has certain special characteristics which are not common to other types of information.

Geographic information is determined by three dimensions:

- *Space*. The spatial dimension defines the coordinates (x, y and z) for a location in the three-dimensional space with a given theme in time.
- *Time*. The temporal dimension defines the coordinates for a location with a given theme in time (t).
- *Theme*. The thematic dimension defines values of one or more thematic attributes having a location in time.

The structure (x, y, z and t) provides the framework for collecting thematic attributes [Veregin and Hargitai, 1995]. Geographic information is characterized as either entity-based or field-based. In the entity-based approach the real world consists of fully definable disjunctive entities, such as roads and buildings. In the field-based model the real world is considered as continuous fields, such as precipitation and temperature. During recording of entity-based information, each object[2] is usually based on a sequence/number of observations. A geographic object has a theme (the object type) with a number of thematic attributes, e.g., a building object which has attributes specifying the number of floors and total square meters. A geographic object will always have a location

[2]We use object for an entity represented in a database.

and extent in the space, hence, it has a spatial attribute. The spatial attribute may specify a point, line, or polygon. A line and a polygon consist of a sequence of connected points. This means that a spatial attribute value may contains a number of point objects (observations). Finally, a geographic object is dependent on time. It may exist in a given time interval as well as its spatial and thematic attributes may be valid or non-valid at a given time. Field-based information contain a number of single independent observations, e.g. observations made by a number of precipitation stations. Here, each observation has one or more thematic attribute values at a given location (x, y, z) at a given time (t). The most important component of geographic information, and what essentially makes geographic information different from most other kinds of information, is the geometric component. In vector-based systems geometries can in general be either points, polylines, or polygons. At least two points are necessary to form a polyline and at least one polyline to form a polygon. A point can either be a vertex or a node. Nodes are starting points and end points in polylines. Points not included in a polyline are also regarded as nodes and are sometimes called isolated nodes. Two other axioms that are not included in the drawing must hold: 1) A polygon has to be closed, i.e. there are no gaps in the boundary, and 2) The boundary of a polygon must not touch or intersect itself.

Topology is properties of geometry that are invariant in respect to scaling, rotation, and affine transformations. Two kinds of topology are used in geographic information systems: Arc-node topology and polygon topology. Herring was one of the first to give a formal description of topology in the context of geographic information using a point set approach [Herring, 1991]. [Clementini et al., 1993] named to standard topologic relations, while [Smith, 1996] introduced a logical system that defines the topological relations as predicates, based on a mereologic basis.

Constraints are an important aspect of specifying geographic information. Constraints state when information is consistent and ensure that the consistency requirements are satisfied. An approach to ensuring the quality of geographic data using constraints specified in Object Constraint Language (OCL) has been investigated previously [Casanova et al., 2002, Casanova et al., 2000].

Chapter 3

Specifying Geographic Information

Abstract This chapter discusses the specification of geographic information. The aim is to identify the requirements for a specification language designated for the development of geographic information specifications. The chapter also provides an overview of existing approaches to modeling and specifying geographic information. These approaches are discussed in relation to the development of specifications used in the production of geographic information. The final section presents the running example that will be used throughout the thesis.

3.1 Introduction

Geographic information is created to communicate facts about phenomena in reality. It is the intention to enable the receiver of the messengers embodied in the geographic information, whether it is a map or some information derived from a map, to make decisions. Several studies of the communication perspective of geographic information, geocommunication, have been conducted. An early reference is Kolacny [Kolacny, 1969]. Kolacny was inspired by Shannon's 1949 paper on the mathematical foundation of communication [CE and W, 1949], and he brought Shannon's communication model in the context of geographic information (or cartographic information in Kolacny's terminology), and added the notion of the cartographer's interpretation of the real world and the user's interpretation of the real world into the model. By introducing what he called the cartographic information process, Kolacny was able to relate the production and use of geographic information. The cartographic information process is determined by seven principal factors:

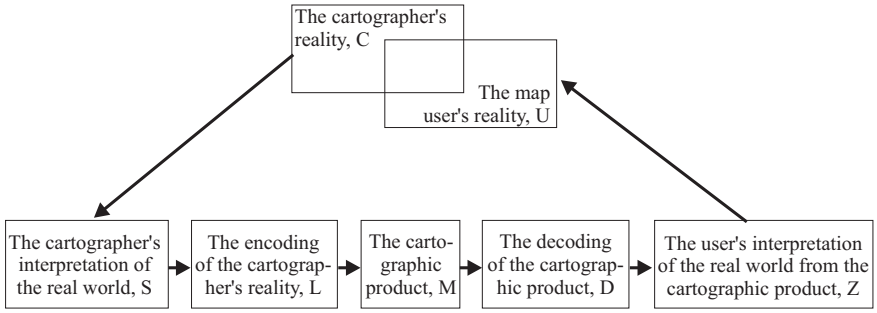

Figure 3.1: Communicating geographic information perspective.

1. The cartographer's reality (C)
2. The cartographer's interpretation of the real world (S)
3. The encoding of the cartographer's reality (L)
4. The cartographic product (M)
5. The decoding of the cartographic product (D)
6. The map user's interpretation of the real world from the cartographic product (Z)
7. The map user's reality (U)

The challenge of specifying and designing geographic information, is to create the basis for sending messengers which the receiver can rely on to make proper decisions and act accordingly. A successful communication depends on the sender and receiver using the same language and giving the same meaning to the terms and symbols used in the message. Developing clear and unambiguous specifications addresses this problem, hence one of the motivations of the specification is to establish terminologies accepted and understood by both the designers and the users of data collections.

This chapter discusses the nature of geographic information specifications, by addressing the process of developing specifications and describing the content of a specification. Section 3.2 discusses the processes leading to geographic information specifications and Section 3.3 introduces the concept of specifications for geographic information. Section 3.4 gives a small example of the content of a specification of a topographic map, just to illustrate the kind of specification we deal with. Section 3.5 lists some of the existing approaches to specifying or modeling geographic information. The properties of these approaches are discussed in relation to the requirements for developing specifications from which information can be produced. Section 3.6 lists identified design goals that the framework developed in this thesis must achieve. Finally, Section 3.6.1 introduces the running example which will be used in most of the examples throughout the thesis.

3.2 The Process of Developing Specifications

The design of geographic information is a challenging task which requires a diversity of knowledge. Typically, the design of a data collection is carried out by a group of people representing both the organization responsible for the development and the stakeholders who have an interest in the data collection that will be produced from the specification.

There are a number of different approaches to and methods for gathering and integrating the required knowledge for design of a system. Examples are Object-oriented Analysis and Design (OOAD), AGILE methods, Model Driven Architecture, and Rational Unified Process (RUP). A newer approach is the concept of *Co-design* or Cooperative Design. This approach presumes that the world consists of an infinite number of views of reality. These views or perspectives on reality must be expressed in the design an artifact, in our case a database consisting of geographic information.

The key idea is that the number of persons contributes to the development of specifications. End users and domain experts have knowledge of the problems in

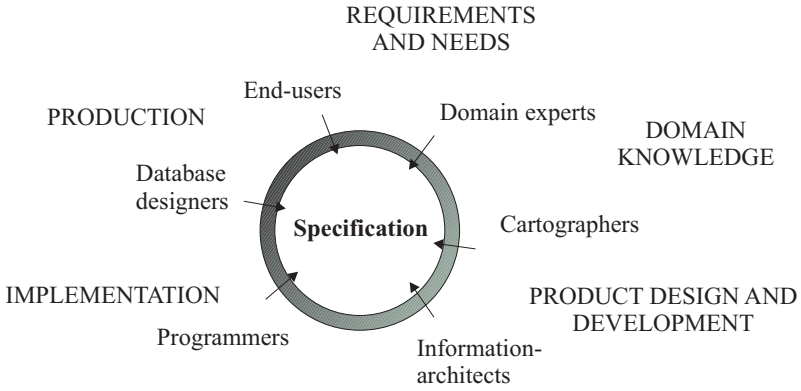

Figure 3.2: The principle of co-operative design

the domain, and so that they can contribute to the analysis leading to the identification of the requirements for the data collection being developed. Experts in geographic information, such as cartographers and geographic information architects, have a large knowledge of the representation of geographic entities and thus contribute to the design of the information at the conceptual level, such as the definition of object types, attribute types, and relationships. Computer engineers, programmers, and database experts are skilled in implementing software and databases which can facilitate the production and storage of the information specified at the conceptual level.

The point is that all the different kinds of knowledge are needed to develop specifications which meet the user's requirements and needs, have a well-defined structure and content, and which can be produced and stored in an appropriate way. To support cooperative design is a key principle for the GeoSML framework that will be developed in this thesis.

3.3 Specifications for Geographic Information

It is a challenging task to specify geographic information. The difficulties arise from three sources. (i) Geographic information deals with modeling and representing "real things" like lakes and forests. Everyone who has tried to define what a forest is, or to determine the boundary of a lake, or to answer questions like "can a lake be a part of a forest?" knows how difficult it is. (ii) Describing how the location and the extent of entities are represented by geometries is an

extensive process, which depends on scale and available sources of information. (iii) Specification of geographic data collections requires large amounts of information. Specifying a topographic map database may require several hundred pages, including detailed descriptions for 100 object types or more.

The content of a specification is influenced by a number of factors. Ideally, user requirements are decisive when the content and structure of a geographic data set are designed. User requirements may be contradictive or stated requirements not supported by a will to cover the cost for establishing the information that meets the stated requirements. Several other factors influence the content of a specification, among others politics, culture, traditions, the nature of domain, and last but not least available technologies and resources.

Specifications are basically created to describe a vision developed and owned by a group of individuals. Without writing down ideas and decisions in a specification, an in-depth understanding of the problem domain will never arise, and a possible design will be based on unsolid ground. In this way, the process of developing specifications forms consensus among the individuals, groups, or organizations that have an interest in the data collection produced from the specification. Seen from a user's point of view, specifications and the realization of specifications, seen in relation to the specific application, determine the perception of data quality. The value chain from user requirements, over written specification, to deliverable products is essential to creating products that meet the requirements stated by the users [Zeithaml et al., 1988, Parasuraman et al., 1985]. A second point that motivates the development of specifications, is to be able to communicate, in detail, what content a data set can be expected to have. Thus, specifications can be regarded as detailed metadata or more general metadata may be derived from a specification.

In general specifications deal with designing artifacts. The methods used for developing specifications vary depending on the kind of artifact to be designed but the methods may also have similarities. They must all include a collection and representation of knowledge of the domain in which the artifact will operate and describe the problems that it seeks to solve.

A specification for geographic information must answer two key questions: "Which entities in the domain must be given a representation?" and "How must this representation be?". A specification constitutes what is called a nominal ground or universe of discourse [Worboys, 1995, Friss-Christensen, 2003] and can be regarded as a mechanism that points out entities in a domain and defines how these are represented as objects.

Geographic information is an abstraction or representation of the real world. To be able to design appropriate representations, a fundamental understanding of

the entities and phenomena in geographic space is needed. A geographic entity, e.g. a house or a river, is characterized by its location in space and existence in time, and is described or classified by a set of properties. It is the representation of these three components that results in geographic information.

Classification of entities builds on the assumption that things in reality can be grouped according to number of properties. The classification of an observed entity decides if it should have a representation or not, by which attributes the representation should be described, the allowed values for each attribute, how the representation must change according to changes to the geographic entity, and finally which quality requirements there are to the representation.

There are a number of approaches to formal classification of entities. Formal Concept Analysis is a mathematic framework based on the assumption that a Galois connection exists between an ordered set of types and an ordered set of properties. This approach to classification is especially useful for analyzing the inheritance structure between a large number of types or concepts and to identify "hidden" concepts for which there are no denotations in natural language.

Data modeling using entity-relationship diagrams or UML can also, seen from a set-theoretical point of view, be regarded as a formal classification of types. In Section 2.2.2 the relation between entity-relationship diagrams and description logics is explained.

An important part of classification is to determine the **properties** of an entity type. Often there is a differentiation between **necessary** and **sufficient** properties. A property of an entity is necessary if it must hold for all entities of the type concerned. The necessary condition can therefore be seen as definitional, e.g. a building must have walls. Sufficient properties that may or may not hold for an entity. These properties can be used to describe properties may change, e.g. the color of a car.

Changes in reality may result in alteration of geographic entities that are represented as objects in a geographic data collection. When this happens a need for updating the representation of the entities may be in question. One of the predominant questions in relation to capturing changes and introducing these in the relevant data collections is the question of **identity** and **existence**, or in other words "how much can an entity change and still be perceived as the same entity?" and "when does an entity exist?". Producers of geographic information often ask themselves these questions, and without a well-defined basis for making decisions about the identity and existence of entities as they are born, changed, and die, the answers may be inconsistent.

Traditionally, changes of entities are modeled by use of state charts which are

basically petrinets [Sowa, 2000]. State charts describe allowed transitions between states. These models enable a process view of the life cycles of entities that includes a description of the situation before and after a change occurs, and less about what actually happened, or about the identity of the entity before and after it changed.

An entity occupies a volume in space. This volume can be described as the **location** of the entity and its **extent**. Entities can be either atomic or composite. Composite entities consist of other entities. The relation between a composite entity of which and the entities it consists is the part-whole relation (see Section 2.3). Thus, two entities may share the same volume in space. Representing composite geographic entities may be

3.4 A Small Example

The following statements are taken from the specification of TOP10DK [KMS, 1999]. The statements describe some of the details needed to represent buildings in reality as building outlines in the topographic map.

1. A building is defined as a house, foundations in connection with the construction of a house, or a ruin of building structural character.
2. A building is registered at the roof overhang/eaves of the house.
3. A building must be registered as closed polygons with a common start and end point.
4. A building should be registered on the outer extremity of the foundations or ruin.
5. As a general rule, all buildings larger than $25m^2$ must be registered in their fundamental form.
6. A building must be registered with as few points as possible but in such a way that the difference between the actual sequence and the registered sequence is never larger than 1 m in plan and elevation.
7. All building corners must be registered. However, buildings with overhangs and extensions of a side length of less than 3m and an area smaller than $10m^2$ must not be registered. Agricultural buildings smaller than $100m^2$ in connection with farms that are considered to be used for habitation must not be registered.

8. Houses built together must be registered as one building. However, multi-storey buildings with a difference of more than 5m must be registered as independent BUILDINGS with common 2-D geometry. Industrial buildings built together are registered as one `building` irrespective of large differences in elevation.
9. Storage barns larger than 100m^2 must be registered as building.
10. Fur farm buildings located closer than 2.5m to each other must be registered as one `building`.
11. Separate platform roofs between `railroad tracks` must not be registered as building.
12. `building` that seem to be right-angled must be rectified mathematically. However, no registered point must be moved more than 0.5m.
13. Telegraph mast and towers with the character of a building should be registered as `building`.
14. Buildings like terrace houses, blocks of flats and housing blocks in residential areas with a uniform appearance must be generalized homogeneously.
15. Buildings preserve their identity as long as the extent and the building parts do not change considerably.

The main purpose of such descriptions is to enable an agent, human or machine, to analyze the source material, e.g. aerial images, and to extract information from these to represent interpretations of identified entities as objects in the data collection. In its most abstract notion a source material is an observation of a property. Observations are evaluated by the agent using the appropriate statements from the specification as the basis. By this process the agent acquires new knowledge of the domain. When having sufficient knowledge the agent draws conclusions, which result in one or more objects representing a number of entities within the domain. The interpretation process is complicated and has been a research subject within philosophy and cognitive sciences since the times of Aristotle and Plato. In the following, we settle for a pragmatic approach by assuming that such things as entities and properties exist, and that the agents are capable of identifying entities and observing their properties.

3.4.1 Discussion of the Example

Even though the above example is a small fraction of a specification, valuable information about the fundamental parts and elements included in a specification can be extracted. Look at the following statement: *Fur farm buildings*

located closer than 2.5m must be represented as one building (statement 10). This statement includes a lot of information. There is something called *Fur farm buildings* which can apparently be located at a certain distance, and if the distance between some instances of fur farms is less than 2.5m, it has a consequence for the representation, namely that they must represented as one building. Several issues can be questioned, e.g. should fur farm buildings that are further away from each other also be represented as buildings? Or are fur farm buildings a kind of agricultural building, and if so, should they conform to the minimum area criteria in statement 7? And if they should, how is it determined if they do so: is it the total area of the fur farm building candidates for being represented as one, or is it the area of the individual fur farm building that must be larger than $100m^2$? All these questions should be answered before the agent can use the description to evaluate how fur farm buildings should be represented. The description as it is seems incomplete, which gives rise to the question: when is a description complete? The simple answer to this question is that descriptions are never complete. Naturally, this answer is not satisfactory, and we will throughout this thesis supply more adequate answers. The first step is to recognize that more effective methods for structuring the specifications are needed to ensure the quality of the specifications. The example in the previous section can serve as input to a first structure of specifications.

Two Vocabularies

The first important observation that to be made from the above example is that the statements are formulated by use of two sets of terms. One set describing real world entities and one set describing the objects that represent the entities. It is important to be able to distinguish between terms referring to entities and terms referring to objects.

Definitions

Definitions, and as we shall see later, designations are important components of a geographic information specification (or for all kind of specifications for that matter), hence terms need to be defined if they should give meaning to the specification.

Statement 1 is an example of a definition and it defines what a building is in the given context.

Criteria of Importance

To determine which entities should be represented in a data collection, it is not enough to list the types of entities that should be included. Often the product cost will increase dramatically, if there is not a set of rules restricting the entities that should be represented by some defined criteria of importance. Statements

5 and 9 are examples of criteria for importance. Statement 5 is a general rule for all buildings, and statement 9 imposes further restrictions on buildings used for agricultural purposes.

Rules for Instantiation of Objects

Statement 10 describe the instantiation of fur farms buildings as discussed above. In general instantiation rules are included in a specification for either cost or cartographic considerations.

Guidelines for Drawing Geometries

An important part of a specification is the guidelines for drawing geometries. Statements 3,4, and 6 describe the properties that are decisive for the representation of the location and the extent of the buildings by polygons.

Generally speaking, the guidelines for drawing geometries can be divided into two groups. One can be called *"internal"* guidelines describing requirements for the accuracy of the points and line segments and the number of details that must be included in the geometry. The second group of guidelines consists of *"external"* guidelines describing the impact the representation of other entities may have, e.g. how a stream is drawn if it is adjacent to a forest, or if a road crosses a river.

Life Cycles of Entities

When an entity changes it may have consequences for its representation. The most predominate question is if the identity of the entities is preserved during the change or not. Life cycle rules state under which condition the identity is preserved and under which condition it is changed. Statement 15 is an example of a life cycle rule stating that a building preserves its identity as long as no important building parts are added to or removed from the building.

3.5 Existing Approaches to Specifying GI

Using modeling languages like UML can help to get an understanding of the domain at hand. Geographic information, however, is very complex, and using existing modeling language may be impossible or at least very difficult, if all properties needed to describe fully a geographic data collection should be included in the model. Problems arising from modeling geographic information include: Definition of all appropriate topologic relations, including requirements for acceptable quality levels and measurements, specification of rules or constraints that formulate how objects must be registered and how the life cycles of these objects are, and describe the relations among object types at different

scales.

Several research projects have developed languages for modeling different aspects of geographic information. This section is a summary of preceding approaches, followed by a discussion if existing approaches possess the necessary properties, to establish and support a well-structured and efficient infrastructure for geographic information. This discussion is concluded by defining a number of research topics to be treated in this thesis.

STER

The STER [1] diagraming language [Tryfona and Jensen, 1999] extends in traditional Entity-Relationship approach by adding language elements to modeling of both spatial and temporal properties. STER includes symbols for indicating if an entity set has spatial properties.

IGN Approach

The Cosit laboratory at IGN in France has developed a framework for developing production specification for geographic information. This framework shares the fundamental idea to distinguish between concepts in a domain and geographic object types in a conceptual model [S. et al., 2003].

MADS

MADS [Parent et al., 1998, Parent et al., 1999] [2] combines the ER and object-oriented modeling approaches and supports modeling of spatial as well as temporal properties of spatial entities and relationships. This is achieved by introducing a set of abstract object types (labeled with icons for simple and complex geometries), which can be included in the definition of spatial attributes. MADS provides a number of predefined icons for modeling spatial relationships, and further more specification of simple constraints is supported. Recently, MADS has been extended with modeling constructs for multi-representational descriptions [Vangenot, 2004]. MADS is supported by some formal definitions, specifying the formal definition of the various constructs [Parent, 2004].

OMT-G

OMT-G is an object-oriented data model for geographic applications. OMT-G provides primitives for modeling the geometry and topology of spatial data, supporting different topologic structures, multiple views of objects, and spatial relationships. The language also contains constructs for specification of transformation processes and presentation alternatives, as well as multiple representation [Borges et al., 2001].

[1] STER is an acronym for Spatial-Temporal Entity-Relationship.
[2] MADS is an acronym for Modeling of Application Data with Spatio-temporal features.

MRSL
Multiple Representation Schema Language is a language specific to descriptions of relations and dependencies between multiple representations. The method includes the definition of integration classes, which are responsible for instantiating and keeping consistency among multiple representations of the same geographic entity [Friss-Christensen, 2003]

GeoFrame
GeoFrame is another extension of UML and serves as a framework for conceptual modeling of geographic information. The approach differs from other UML based extensions by using analysis patterns which focus on reuse [Ruschel et al., 2005].

3.5.1 Discussion of Existing Approaches

In this section the existing approaches to specifying geographic information are compared to the need for a specification language in the development of specifications used to support the production of geographic information.

As the example in Section 3.4 shows production specifications are composed of a diversity of types of descriptions. The most predominate drawback of the existing approaches is that none are particularly well-suited for structuring or formalizing these types of descriptions.

It also seems that existing approaches first of all address the conceptual modeling of geographic information. Thereby the focus is on describing the structure of the data collection-to-be, rather than the reality it seeks to represent. The result is that there are no direct relations from the specifications to the underlying ontological model describing the reality as the users see it.

Furthermore, the existing approaches lack a method for specifying formal constraints, which are accessible to non-programmers. It is believed that such a method would be of great importance if it should be possible to include formal constraints explicitly in the specifications.

Finally, the existing approaches do not incorporate mechanisms for stating requirements. Without the gathering and structuring of users' requirements and expatiations on the data collection, the design of the data collection may not possess the optimal properties compared to the users' applications. We suggest that the specification of requirements must be an integrated part of a language for geographic information specification.

3.6 Design Goals for GeoSML

Inspired by Bjørner's and Jackson's approach for domain engineering and Lamsweerde's approach to requirement engineering, this thesis suggests a new method for developing specifications for geographic information which is especially designed to meet the requirements for producing information from the specifications. The method, which is called the Geographic Information Specification Markup Language (GeoSML), divides the design process into three parts: (i) domain engineering - resulting in domain models, (ii) requirement engineering - resulting in requirement models, and (iii) information design - leading to conceptual models.

A specification can be regarded as a set of statements. Each statement can be classified according to its role in the interpretation process, representing geographic entities as geographic objects. Furthermore, statements can be related, according to their meaning and type, to form internal specification structures. A statement can assume three basic forms: (i) make assertions about a domain, (ii) describe a conceptualization of a domain, or (iii) state a requirement to a data collection. Furthermore, elements in a domain model can be related to elements in a corresponding conceptual model, and requirements can be related to elements in the conceptual model.

GeoSML addresses the problems with current approaches to modeling geographic information by focusing on the following language properties, and the following design goals are identified as important to the GeoSML framework:

Structuring of specifications written in natural language. GeoSML must support semi-formal specification and structuring of specification parts written in natural language. GeoSML must provide a classification scheme that can be used for markup statements according to their properties.

Support of cooperative design. The GeoSML framework must support a co-operative design process. The combination of knowledge from users of geographic information, cartographic experts, and computing engineers is necessary to design high-quality geographic information.

Separate domain descriptions and design decisions. Elements for making clear distinctions between descriptions of the reality that must be represented in a geographic data collection and the design of the data content must be incorporated in GeoSML. Designers must make a clear distinction between the ontological commitment of a specification and the design of the geographic information.

Inclusion of requirements in the specification. GeoSML must include elements that enable designers to motivate design decisions.

Formal constraints. It must be possible to specify formal constraints directly in the GeoSML framework, which enable designers to specify precise conceptual models.

Quality requirements. GeoSML must provide language elements for specifying quality requirements, which makes to possible for designers to integrate requirements to precision, accuracy, classification, and completeness directly into the conceptual models.

Each chapter in the remaining part of the thesis contributes to different parts of the GeoSML framework. Chapter 4 introduces language elements for developing domain models and conceptual models and relating the two model types. Chapter 5 introduces elements for specifying requirement models and relating requirements to the conceptual model. Chapter 8 develops a formal constraint language, and Chapter 6 supplies language elements for specifying quality requirements.

3.6.1 Running Example

This section presents an example that will be used to illustrate the developed concepts and elements of the GeoSML framework throughout the thesis. Since "real" specifications for geographic data collections include several hundreds of pages, the example cannot include all aspects of a specification. The challenge has been to develop an example that illustrates the complexity of specifying geographic information and still keeps the example as simple as possible.

The example focuses on the conceptual level of a specification, meaning the structure of the information which must be stored in a database. In the following chapters the example will be expanded to incorporate domain descriptions, functional requirements, quality requirements, and constraints.

The example is designed to include entity types which are typically represented by points, polylines forming a network, and polygons that the entity types represent, land-use areas, man-made objects, and administrative boundaries. From the above criteria the following object types have been chosen:

- Building
- Address

- Municipality
- Road
- Land-use area

The properties of and relationships between these obeject types are illustrated by the ER diagram at Figure 3.3.

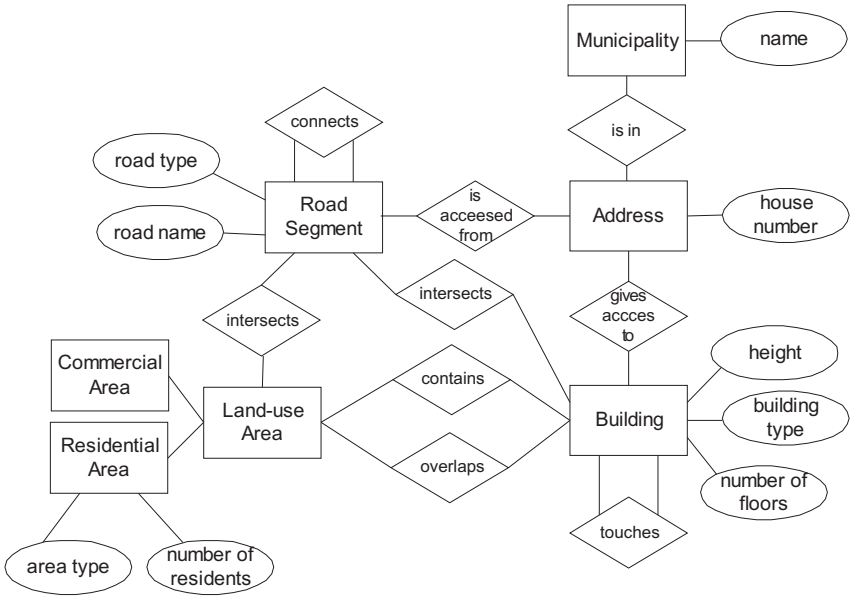

Figure 3.3: ER diagram for the entity types and relations included in the running example.

In the following descriptions of the object types are introduced.

Building
Definition: Buildings are man-made objects created for a particular purpose, e.g. residence, commerce, or production.

Properties: A building has a construction year, a number of floors, main usages, and can be owned by one or more persons or companies. The construction year is defined as the year in which the building was taken into use. The number of floors is defined as the number of different levels above ground separated by a staircase or an elevator. The main use of a building is defined as the predominate

activity carried out in the major part of the building, e.g. commerce, industrial production, or living. All buildings have one or more entrances. Each entrance may have an address.

The location and extent of a building are represented by a building outline. The outline must be registered with as few points as possible but in such a way that the difference between the actual sequence and the registered sequence is never larger than 1 m in plan and elevation. All corners in the building outline must be registered. However, buildings with overhangs and extensions of a side length of less than 3 m and an area smaller than $10m^2$ must not be registered. Agricultural buildings smaller than $100m^2$ in connection with farms that are considered to be used for habitation must not be registered.

Houses built together must be registered as on building. However, multistory buildings with a difference of more than 5m must be registered as independent buildings with common 2-D geometry. Industrial buildings built together are registered as one building irrespective of large differences in elevation.

Address

Definition: An address is a unique identification of an entrance to a building.

Properties: An address is characterized by its house number, road code, and municipality number. A house number is composed of a positive integer and an optional letter. The numbering of buildings on a road starts with 1 and 2 for the first buildings on each side of one end of the road. Buildings on the second side of the road are assigned even numbers in steps of two, while buildings on the first side are assigned odd numbers, also increasing the number with two. The optional letter is applied to e.g. group attached buildings or if a new building is constructed between two existing buildings already assigned a house number. A road code is a number uniquely identifying a road within a municipality. A municipality number is a number uniquely identifying a municipality in Denmark.

An address is represented as an address point with x- and y-coordinates, and located inside a building assigned a house number.

Municipality

Definition: A municipality is an administrative body which is responsible for local planning, providing educational institutions, well-fare, etc.

Properties: A municipality has a name, a number of inhabitants, and a municipality number.

Geometrically, a municipality is represented by a municipality border. A polygon represents the outer extremity of the municipalities. Municipality borders must not intersect each other or be self-intersecting.

Road

Definition: A road is a strip of land, smoothed or otherwise prepared to allow easier travel, connecting two or more destinations.

Properties: A road is described by a road code, the municipality number, and a classification code. The road code is a four-digit number, which together with the municipality number can be used to decide the name of the road. The classification of a road can either be "minor road", "major road", or "highway".

The location and extent of a road are represented by a center line. A polyline marks the approximate center of the road. Center lines representing roads meeting in an intersection must have a starting point or an end point in common.

Land-use Area

Definition: Land use is the pattern of construction and activity for which land is used. Patterns of land use arise naturally in a culture through customs and practices, but land use may also be formally regulated by zoning, other laws or private agreements such as restrictive covenants. For example, the setting aside of wilderness either publicly as a wilderness area or privately as a conservation easement [Wikipedia, 2005].

Properties: To illustrate the use of inheritance, land-use areas are divided into commercial areas and residential areas. Residential areas have a property indicating if the area consists of dense or scattered residences and the number of residents within the area.

The extent and location of land-use areas are represented by polygons indicating the outer extremity of the areas.

3.7 Summary

This chapter discusses various views on the specification of geographic information. The most important result is the identification of five design goals, which should be met by the design of the GeoSML framework. The design goals were primarily chosen because we want to develop a method for designing geographic information specifications which can be applied in the production of geographic information and which supports cooperative design processes.

Chapter 4

Structuring Specifications for Geographic Information

Abstract: This chapter introduces a framework for specifying geographic information, which is called the Geographic Specification Markup Language (GeoSML). GeoSML is based on the theories presented in Chapter 2 and the requirements identified in the previous chapter. The suggested framework has three important properties: (i) It enables designers of geographic information to structure and organize natural-language-based specifications, (ii) It supports a cooperative design process, and (iii) it is capable of formalizing natural language statements. It will be argued that a specification can be divided in tree separate parts: (i) a domain model, (ii) a conceptual model, and (iii) a descriptions that relate the two types of models. A grammar for these basic elements of GeoSML will be introduced in this chapter, including the notions of statements, terms, domain model, and conceptual model. The following chapters will contribute to GeoSML by adding new elements, each of which provides developers with new syntactic structures and guidelines for writing specifications for geographic information.

4.1 Introduction

The previous chapter discussed the specification of geographic information. It was described how national mapping agencies develop large specifications written in natural language, describing the content of data collections in such detail that geographic information can be produced from them. These specifications are developed in text editors offering little or no support for semantic markup or structuring of the specification elements. As argued in the problem description (Section 1.1), there are several reasons why a new approach to developing and managing specification for geographic information is needed. The main argument is that the ontological commitment is implicitly given in existing specifications. The consequence is that the description of domain properties may be unclear, and that in some cases it is difficult to recognize if a statement describes properties of the domain, or if it describes properties of the data collection. The design of a geographic data collection may therefore be based on an unexpressed understanding of the problem domains, which leads to specifications built on a vague and potentially faulty description of the problem domain. The specifications may also include several hundred pages, making it difficult to retrieve the information needed in a given situation, e.g. to be sure that all rules concerning the representation of a particular object type have been revealed in a given situation.

The major inspiration for the GeoSML framework is Bjørner's TripTych paradigm for software engineering [Bjørner, 2006b], Jackons approach to describing domains, and the STER modeling language suggested by Tryfona and Jensen. It is suggested that the development of specifications for geographic information involves three disciplines: Requirement engineering, domain modeling, and conceptual design. Practicing these three disciplines results in three types of models: Domain models, requirement models, and conceptual models. We will focus the introduced constructors on meeting the requirements for modeling geographic information. The aim is to minimize the complexity, resulting in a language that is targeting experts both within the area of designing geographic information and experts within developing computer systems and information technology.

Existing approaches to modeling geographic information primarily concern conceptual modeling focusing on the data structures in which information is stored, and less on the underlying phenomena and ontological commitment these structures seek to represent. The design of database structures is of great importance, but it is equally important to specify the nature and content of the information that will be included in the data collections. Approaches have been suggested to distinguishing between domain models and conceptual models based on UML [Larman, 2001]. It seems that these methods focus on drawing class diagrams,

rather than achieving an in-depth understanding of the problem domain by describing the domain facets. Moreover, there is no support for describing how concepts in domain models relate to concepts in the conceptual models, which is a must in specification of geographic information. Gesbert [Gesbert, 2004] and Mustière [S. et al., 2003] proposed a profile of UML that addresses these problems. Unfortunately, the specification of the elements in a profile is not complete and the grammar for structuring the statements in geographic information specification is missing.

The remainder of this chapter is organized as follows. Section 4.2 introduces a representation system that defines the basis of GeoSML. Section 4.3 defining specification elements for specifying domain models, Section 4.4 the elements for specifying designs using conceptual models, and section 4.5 specify the relationships among the two model types. The chapter is finished by a summary in Section 4.6.

4.2 Representation System

A specification can be regarded as a collection of statements that describe how geographic entities are identified, which ones should have a representation, and how this representation should actually be. These statements are built by use of two sets of vocabularies, one used for denoting *geographic entities* and their *properties*, and one for denoting *geographic objects* and their *attributes*[1]. If establishing a framework for specifying clear and unambiguous specifications for geographic information should be successful, a systematic approach to identifying and denoting phenomena in the real world and their representational counterpart must be introduced. The representation system suggested has four levels as illustrated in Figure 4.1: (i) The domain, (ii) a model of the domain, (iii) a conceptualization that describes which information should be stored about entities in the domain, and (iv) the representation of entities.

The **domain** and the **domain model** represent reality and an abstract descriptions of reality at terminology level. The **conceptual model** and the **representation of entities** represent the data collection, and a description of the data collection, i.e the two boxes to the left represent the reality and and a terminology enabling us to talk about reality, and the two boxes to the right represent data collections and a description of the structure and content of the data collection. The distinction between what is in reality and what is in the computer is analogous to the world/machine distinction made by Jackson

[1] Entities and properties denote real world phenomena and objects and attributes denote the representation of entities and properties.

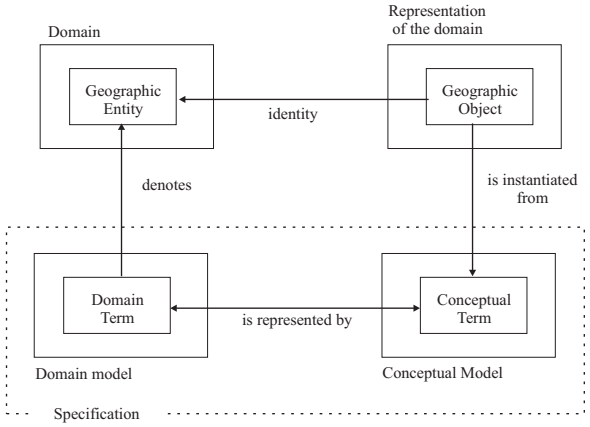

Figure 4.1: Representation model.

[Jackson, 2001] (see page 13). To the left the domain and the domain model represent the world and a terminology for describing the world. To the right the conceptual model and the representation of the domain represent the system or in our case a geographic data collection.

A **domain** is a part of the real world that is interesting for a particular problem [Jackson, 1995, Shlaer and Mellor, 1992]. The domains that are interesting to the specification of geographic information incorporate geographic entities. A geographic entity is a real world phenomenon with spatial properties, which in a given context can be distinguished from all other geographic entities. An entity is characterized by a number of properties. Each property may have a value determined by an observation. Distinguishing between entities and properties can in practice be challenging. In general, an entity is a phenomenon that should be exposed as an individual, while properties are unary predicates and functions, e.g. *the house is used for living* and *the height of the house is 10 meters* describe a particular entity. A geographic entity can be classified according to its properties, e.g. material, function, use, and spatial relation to other entities, but also location, orientation, size, and shape play a role in the classification classifying geographic entities.

A **domain model** is a conceptualization of a domain of interest. A domain model captures the characteristic properties of a domain that for certain applications are particulary interesting. For the domain of real estate, properties like number of bedrooms, distance to the nearest supermarket, and market value are interesting properties of a building, while the type of heating system and the consumption of energy are interesting properties of a building within the

domain of measuring the environmental impact from heating buildings in residential areas.

To capture interesting properties of a domain, a set of basic description techniques are needed, including the development of taxonomies and writing designations, definitions, and assertions.

A **conceptual model** expresses design decisions. Based on acquired requirements and the domain knowledge, the conceptual model formulates the structure in which entities must be represented and the constraints with which the represented entities must comply for geographic information. The conceptual model includes a number of geographic object types. Each object type is associated with a list of attribute types. Object types and attribute types are named by use of terms. Attribute types are furthermore described by a data type, a measure and a unit that are used when objects are instantiated. To be precise about the conceptualization of a domain, constraints can be included in the conceptual model.

Geographic entities can be represented as **geographic objects** stored in computable data structures. A set of geographic objects is often denoted a data collection or geographic information. The intention of geographic information is to establish models that simulate the properties and behavior of geographic entities and the relations among them. Objects are classified according to the type of the real world entity they represent, and the characteristics of the entities, i.e. the number of floors or the outline of a building, that are important to a given application are modeled as attributes, for which each object is assigned attribute values.

The **is represented by** relation between the domain model, and conceptual model in Figure 4.1 represents a mapping from descriptions in the domain model to elements in the conceptual model. In fact, this mapping constitutes the backbone of a specification and mapping defines at an abstract level the interpretation of domain concepts and binds these to corresponding terms in a representational model. The *is represented by* relation explains at the conceptual level how entities are instantiated.

While *is represented by* operates at the conceptual level, the **identify** relation operates at the instance level. It is a mapping between objects in the data collection and entities in the domain. The instantiation of this relation is made by humans when recognizing a map object, e.g. a building or a forest, as corresponding to their respective counterpart in reality, or the opposite case when it is recognized that an entity in reality is represented by an object in a data collection. The purpose of creating maps is often to enable users to establish this relation, for example when using a map for navigation. For some types of

objects the creation of *identify* relations is supported by reference systems.

As illustrated in Figure 4.1, specifications following the GeoSML concept include three components: a domain model, a conceptual model, and a mapping between two corresponding models. In the following sections, the content and structure of each of the three components are developed, and the syntax for these basic parts of GeoSML is presented.

4.3 Domain Model

The purpose of a domain model is to capture the domain knowledge needed to make proper design decisions, i.e. domain models are developed to describe the part of reality the geographic data collection must represent. Engineering domain models is challenging. One of the problems is to describe the domain with an appropriate level of details. Including too few descriptions leads to underspecified models, which may miss important knowledge of the domain. Including too much information leads to overspecified models, which in return result in complex models where the most important information may be difficult to identify. Frank [Frank, 1998] recognizes this problem and suggests that if none of the two can be avoided, underspecification is to prefer. This guideline is not really operational, and therefore the key to solve the problem is found in the requirement model (requirement models are discussed in Chapter 5). In general domain engineering is an activity that precede requirement engineering, since stating requirements is done in terms of domain models and the fundamental terms in the requirement model must be defined in the domain model. In practice domain engineering and requirement engineering are an iterative process, where domain terms, assertions, and requirements are defined as the knowledge and nature of the problem and solutions to these increase.

Domain models included in specifications of geographic data collections identify and describe sets of individuals or geographic entities that are particularly important ti the domain. These entity sets are named by relating them to what we call domain terms. Domain models also relate the named sets to each other and constitutes the ontological commitment of the specification. Domain descriptions in GeoSML build on Jackson's approach, and the basic methods are identification and designation of ground terms, defining new terms on the basis of designated terms, and making assertions that restrict or relate designated or defined terms [Jackson, 1995, Jackson, 2001].

Descriptions in a domain models can be more or less formal, spanning from text descriptions to formal ontology developed in description logics [Baader et al., 2003]

or algebraic specification and model-based languages like the Vienna Developing method [Fields, 1992] and the RAISE specification language [George et al., 1992]. Both description logic and algebraic specification languages are very expressive and can include detailed descriptions of the domain. The challenge is to choose a language that will be productive, i.e. it must be able to express the knowledge of the domain needed for an application and not be so complex that it is to difficult to learn and use. GeoSML adopts a semi-formal approach to describing domains. Descriptions are primarily written in plain natural language and structured according to Jackson's theories on domain modeling (*terms*,*designations*, *definitions*, and *assertions*). To model relations among domain terms, or more precise the entities they refer to, GeoSML includes a formalism based on Entity-Relationship (ER) diagram 2.2.1.1. There are several reasons why entity-relationship diagrams are chosen to represent the ontological relations in the specification. The main arguments are that we do not want to increase the complexity of the modeling language unnecessarily, and than the specifications of geographic information focus on representational issues rather that reasoning about the terms included in the specifications. It is believed that to be productive in real world specification development projects, simplicity comes before expressiveness, and most domain experts are not able to express their knowledge in description logics.

4.3.1 Domain Terms

Words are not given by nature, but created and defined by humans. The meaning of words may change over time, new words are invented, and the importance of others decreases. Some words may even be forgotten. A characteristic property of a language is that people using it give the same meaning to the words, or to be more precise, almost the same meaning. In some cases it is important to be precise about the meaning of words, e.g. in law texts, contracts, scientific reports, and also in specifications for geographic information. It is therefore a very important task to identify and define important terms in a specification, hence they constitute the fundamental terminology of a specification.

Domain terms are used to denote geographic entities and properties, which are identified to be of the same type, e.g. *Building*, *street*, and *forest* are terms to denote geographic entity types, and *number of floors*, *Street name*, and *Vegetation type* are terms denoting property types that may be associated with the listed entity types. Properties are semantically defined differently, e.g. the height of a building is not the same thing as the height of a telephone mast. Therefore, a property type can only describe one type of entities. This just means that terms may have several senses and that the representation system must be able to distinguish two senses from each other.

Terms may have several senses, but in general only one sense should be assigned to a term in a specification, since terms with more than one sense may lead to confusion and misunderstandings. The occurrence of terms that need more than one sense is, in the specification of geographic information often related to terms referring to properties, e.g. height, size, and color. When two properties are referred to by two syntactically similar terms with different semantics, a solution can be to include the term that the properties describe in the term referring to the property, e.g. the heights of buildings and poles can be referred to as building height and pole height.

4.3.2 Designations

The knowledge expressed in traditional data models developed in e.g. UML or entity-relationship diagrams is not sufficient if information should be produced from the models. Data models include classes, their properties, and relations among the classes. Thus, traditional data models focus on the structure of information rather than what things really are as phenomena.

To be able to include buildings or greenhouses in a data collection, they must be recognizable. The tool for this is designations. A designation has two parts: The designated term and a recognition rule. Terms are discussed in the previous section, and recognition rules are informal natural language rules that can be used to identify and classify real world phenomena. Exampled of designations from the domain of topographic mapping are:

Tree *A large, perennial, woody plant, having secondary branches supported on a single main stem or trunk. Compared with most other plant forms, trees are long-lived. A few species of trees grow to a height of 100m, and some can live for several millennia.*

Building *Man-made construction made for human activities, e.g. for living, sports, industrial production, and commerce. Generally, buildings have a foundations, a number of walls, and a roof. The various construction parts can be made from a number of building materials. Foundations are mostly made from concrete and walls can be made from bricks, glass, or wood.*

Road is a strip of land, especially suitable or prepared for car driving.

Designations say nothing about the domain, only that there are some phenomena that can be recognized. Relations and rules connecting the designated phenomena are applied by assertions, which is discussed later in the chapter.

From a phenomenological point of view the recognition rules are formulated in terms of the properties which are evidential for a given entity type. The perceived physical appearance is used to recognize an entity and to classify it as a particular type. Therefore, when designations are written, the physical evidence of the entities is used as recognizable properties. Examples can be the parts of which a particular type of entity normally consists, or physical property or combination of properties unique for a set of entities, e.g. color, height, orientation and distance to other entities.

4.3.3 Definitions

New terms can be defined on the basis of designated terms. A definition is a concise statement that includes as much information as it can in a minimum amount of space. It consists of three parts: (i) The term to be defined, (ii) the concept under which the entities the term refers to falls, and (iii) the properties differentiating the entities from all other entities of this concept. An example is *a greenhouse* (term) *is a building* (concept) *made for growing plants and vegetables*(differentiating properties).

A definition can be supported by mentioning the parts of which the reference term of the entities consists of, e.g. *a greenhouse is a building made for growing plants and vegetables and has walls and roof made from glass*. The concept used to define the term should be as "close" to the term as possible, e.g. *a greenhouse is a building made for growing plants and vegetables and has walls and roof made from glass* is better than *a greenhouse is a man-made construction for growing living organisms and has construction parts made from glass*. In general definitions should not be written by using negations, e.g. *windmills are not buildings*. Describing what a term is not can easily become exhausting. Besides, circular definitions should be avoided, e.g. *evergreen tree - tree with evergreen foliage*. Definitions can be formalized and thus differ from a designation. Moreover, definitions do not add anything new to the description of the domain other than a convenient way to refer to a set of entities which can be recognized by a designation, and which have some special properties separating them from all other entities identifiable with the same designation. Examples of formal definitions using predicate logic are given below.

$$SingleStandingtree(x) \rightarrow \forall t_1 tree(t_1) \neg \exists t_2 tree(t_2) \wedge close(t_1, t_2) \wedge t_1 \equiv \neg t_2$$

The example above defines single-standing trees. It can be interpreted as a single-standing tree is all trees for which it holds that no other tree is close to the tree.

$$\forall x\ forest(x) \rightarrow \exists s\ set(s)\ \forall a\ \forall b\ (isin(a,s) \wedge isin(b,s) \wedge indirectlyclose(a,b) \wedge count(s) > 200 \wedge partof(a,x) \wedge partof(b,x)$$

The next example defines the concept of forest. The definition of forests is somewhat complicated, since it includes a set of trees which must have particular relations to each other, namely that all pairs of trees in the set must be close or indirectly close to each other. The predicate close is evaluated to be true if the distance between two trees is within a defined threshold, and isdirectlyrelated is defined as:

$$\forall a \forall b indirectlyclose(a,b) \rightarrow close(a,b) \vee \exists c(close(c,b) \wedge indirectlyclose(a,c))$$

Descriptions of domains most likely include directly observable phenomena, like forests and residential areas, as designated. Definitions are more relevant phenomena that are only observable over a period of time, e.g. traffic density. On the other hand, geographic definitions may be relevant in the definition of new data collections in the context of others, or to support the specifications of constraints. In both cases the formal definitions are specified in terms of conceptual models, and not domain models. The problem of giving formal definitions to geographic entities is that they concern "real world phenomena". To find a term that can be designated and used to define terms like *hill* and *mountain* is difficult. The suggestion is that if definitions of domain terms are needed, it should be kept in natural language. The consequence is that GeoSML does not include capabilities to specify formal definitions at domain level, other than what can be achieved by defining relations and assigning attributes to subtypes.

4.3.4 Assertions

A domain is described by use of a set of assertions which form an abstraction that defines a particular view of the problem domain. Assertions are formulated in the context of designated and defined terms and express rules or regulations describing some properties of the domain and thus contribute to the development of a domain theory. Examples of assertions are: *Buildings may be owned by persons or by organizations*, *buildings of historic value are subject to a special reduced property tax*, and *the water level of a lake is determined by the intake and output for a given period.*

Jackson emphasizes that assertions must be refutable. Assertions that cannot be questioned gives no substantial knowledge of the domain. If an assertion states that highways have at least two lanes in each direction, someone may

come up with counterexamples: "What about the lanes that connect two intersecting high ways?", or "if a lane is closed because of road work?". This kind of argumentation is desirable. Only by "negotiating" the meaning of terms stable domain descriptions that are applicable to design processes can be developed. It is important to remember that descriptions which fully capture the meaning of words are difficult to achieve, and problems will always arise. Therefore, agreement on designations, definitions, and assertions is a necessity, without agreement formalization of the domain concept is pointless. As John Neumann wrote in The Theory of Games and Economic Behavior [von Neumann and Morgenstern, 1947]: *"There is no point in using exact methods where there is no clarity on the concepts and issues to which they are to be applied"*.

4.3.5 Domain Modeling

Using the basic description techniques presented in the previous section enables designers to capture domain knowledge in a structured, but yet informal manner. To be able to represent formally the meaning captured by the domain descriptions, relations among domain terms must be more precisely defined. The GeoSML syntax therefore includes elements for formalizing parts of domain models. The elementary method for doing this is to group properties according to the entities they describe, thus forming entity types. By defining entity types a commitment to the domain is made. It is postulated that a set of entities has so many similarities that it makes sense to treat them equally in the domain model.

Terms denoting entities, e.g. building, forest, and lake, and terms denoting properties, e.g. number of floors, building material, and height, can be related by what we call property relationships. Property relationships can either be mandatory or optional. Properties are implicitly understood as optional. If a property is mandatory is must be labeled with the `mandatory` keyword.

In Section 2.2.1 relations among individuals in sets are discussed and formalized. Since object types can be regarded as a set of entities, the defined relations can be used to describe formally the relations among geographic entities. This is achieved by introducing the notion of relationships in GeoSML. The first version of GeoSML supports: taxonomic relations, mereologic relations, and associations. Furthermore, optional inverse relationships can be specified.

4.3.6 Grammar for Domain Models

The following grammar describes the syntax of the part of GeoSML used for describing and modeling domains. The syntax is written in a context-free EBNF grammar [2] (see [Garshol, 2006] for an introduction to EBNF and [ISO, 1996] for a complete reference).

⟨domain model⟩::=**Domain Model**
(⟨domain term⟩)*
(⟨designation⟩)*
(⟨definition⟩)*
(⟨assertion⟩)*
(⟨entity type⟩)*
(⟨domain relationship⟩)*

⟨assertion⟩::=**Assertion:** ⟨assertionID⟩[**Concerning** (⟨domain termID⟩)*]⟨statement⟩

⟨definition⟩::=⟨definitionID⟩ **Definition:** ⟨domain termID⟩ **:**⟨statement⟩

⟨designation⟩::=⟨designationID⟩**Designation:** ⟨domain termID⟩**:**⟨statement⟩

⟨domain term⟩::=**Domain term:** ⟨domain termID⟩⟨term⟩

⟨entity type⟩::=**Entity type:** ⟨entity typeID⟩
Name: ⟨domain termID⟩
Properties::=(⟨propertyID⟩⟨domain termID⟩)*

⟨domain relationship⟩::=⟨drID⟩⟨drdef⟩

⟨drdef⟩::=⟨dr-is-a⟩ |⟨dr-part-of⟩ |⟨dr-association⟩

⟨dr-is-a⟩::=(⟨entity typeID⟩)* **is-a** ⟨entity typeID⟩

⟨dr-part-of⟩::=(⟨entity typeID⟩)* **is-part-of** ⟨entity typeID⟩

⟨dr-association⟩::=⟨entity typeID⟩⟨domain termID⟩⟨entity typeID⟩

⟨statement⟩::=⟨string⟩

⟨term⟩::=⟨String⟩

A domain model has a name, includes a set of designations, definitions, entity types, and relationships. An assertion is defined as a statement with the keyword "Assertion". A domain term is defined as a term with the keyword "Domain

[2] The Backus-Naur form was developed by John Backus and Peter Naur to describe the syntax of the Algol 60 programming language. Later the notation was extended with optional and repeated symbols.

term". Domain terms can either be defined or designated, by relating a term to a statement, and using the appropriate keyword. An entity type is defined by a name, which is a domain term and a set of properties, each being a domain term. Entity types can be related by the ⟨domain relation⟩ symbol, which can either be expressed as a taxonomic relationship (the ⟨dr-is-a⟩ symbol), an association (the ⟨dr-association⟩ symbol), or a mereologic relationship (the ⟨dr-part-whole⟩ symbol).

4.3.7 Domain Modeling - Example

To illustrate the use of GeoSML's domain modeling capacities this section gives an example of how domain descriptions can be marked up using GeoSML. The example has its origin in the domain of handling of building permissions. It is important to emphasize that a "real" domain model of these domains would require more pages than it would be appropriate to incorporate in this thesis. Therefore, the models only capture a few parts of the domains, - the aim is not to give extensive descriptions of the domains, but to include enough descriptions to illustrate the various language elements.

The example is inspired by the practice in Denmark regarding building permits. Building permissions are given by local authorities and governed by two sets of rules and regulations: local plans and general requirements for building materials and the assembly of building parts, e.g. requirements concerning inflammability or the dimension of building parts. The focus is on the regulations stated by local plans, so that geographic properties are mostly described.

```
Domain model: Building permits
    Domain term building: Building
    Domain term parcel: Parcel
    Domain term parcel_number: Parcel_number
    Domain term construction_Year: Construction_year
    Domain term floor: floor
    Domain term area: area
    Domain term number_of_floors: number_of_floors
    Domain term social_security_number: social_security_number
    Domain term owns
    Domain term is_owned_by
    Domain term is_situated_on
    Domain term contains
    Domain term Greenhouse
    Designation no1 Building: Man-made construction made for human
    activities, e.g. for living, sport, industrial production,
    and commerce. Generally, buildings have a foundation, a
```

```
number of walls, and roof. The various construction parts
can be made of a number of building materials.Foundations
are mostly made of concrete and walls can be made of
bricks, glass, or wood.
Definition no1 Greenhouse: is a building with roof and walls
made of glass and primarily used to grow plants.
Definition Build percent:  is the sum of the area of each
floor in the building divided by the area of the parcel
on which the building is situated, multiplied by 100.
Assertion no1 concerning Building
    New Buildings must be situated at least 3 meters
    from the boundary of the parcel on which the building
    is planed
Assertion no2 concerning Building
    The distance from planned Buildings to existing
    constructions must be larger than 5 meters.
Assertion no3 concerning building_percentage
    The building_percentage  must not exceed 25%
Assertion no4
    The owner of the parcel on which a construction less
    than 10 m2. is planned may begin the construction
    when an application is filled.
Assertion no6. Applications must be filed for all buildings,
that is houses, garages, winter gardens, and greenhouses,
of an area larger than 10 m2
Entity type Building
    Name: Building
    Properties:
        construction_year construction_year
        number_of_floors number_of_floors
Entity type Greenhouse
    Name: Greenhouse
Entity type Person
    Name: Person
    Properties:
        person_name person_name
        social_security_number social_security_number
Entity type Parcel
    Name Parcel
    Properties
        parcel_number parcel_number
Relation owns
    person owns parcel
Relation is_owned_by
    parcel is_owned_by person
Relation is_situated_on
    building is_situated_on  parcel
```

```
Relation contains
    parcel contains building
```

Note that the terms are used as the ⟨`domain termID`⟩. This is done to increase the readability of the example. In "real" specifications a numbering system using a unique integer-based numbering system should be applied. Such a numbering system is not described in the above specification or in following specifications.

4.4 Conceptual Model

The next component of a specification developed using GeoSML is a conceptual model. As explained in the beginning of this chapter, a conceptual model is a prescription of an internal representation structure which acts as a template for the creation of geographic objects. Conceptual modeling has been a subject of investigations by several research projects (see Section 3.5), each proposing its own method or language specially suited for the various aspects of conceptual modeling of geographic information.

Conceptual models express design decisions based on acquired requirements and the domain knowledge expressed in the corresponding requirement and domain models (requirement models are introduced in GeoSML in Chapter 5). The design of geographic information consists of two exercises: (i) development of a conceptual model, including deciding the object type to represent the entity types in the corresponding domain model and the constraints that restrict the values and relations of these objects. (ii) Description of the rules that define the entities to be represented in the data collection, and how the representation should be done. This section deals with the first part, while Section 4.5 introduces the GeoSML elements for binding domain descriptions to the conceptual model.

A conceptual model can be regarded as a set of constraints, which restrict the information stored in a database. The conceptual model includes a number of geographic object types. Each object type is associated with a list of attribute types. Object types and attribute types are named by use of conceptual terms. Attribute types are furthermore described by the data types used for instantiation of the objects. To be precise about the conceptualization of a domain, constraints can be included in the conceptual model. Constraints are statements assigned to guard the intended meaning of a requirement, an assertion, or a representation rule, and which the objects instantiated from the conceptual model must comply. Constraints are always formulated in the context of

conceptual terms. These constraints can either be simple or complex. The simple constraints are treated formally in this chapter, and the complex ones in Chapter 8.

Conceptual models for databases are often represented in entity relationship models (ER diagrams), which are a graphical notation for representing database structure and concepts. An introduction to ER diagramming rules and concepts is found in Section 2.2.1, and a more in-depth presentation can be found in [Chen, 1976]. The conceptual modeling of geographic information has been a subject of intensive research. Some of the suggested solutions and approaches have already been discussed in Section 3.5. The approach proposed here resembles most of the STER approach developed by Tryfona and Jensen, hence our approach and syntax also build on an ER modeling approach. Later the ER-like syntax will be extended with constructs for specifying complex constraints, which add elements similar to those of Description Logics, but without any reasoning facilities.

4.4.1 Object Types

Objects with common characteristics can be addressed as an **object type**. The characteristics, i.e. the number of floors or the outline of a building important to an application, are modeled as attributes, for which each object has an attribute value. Object types are patterns to be used to instantiate objects representing a specific entity. In GeoSML object types are described by a name and a set of attribute types. Each attribute type has an attribute name, unit, and data type.

There are three types of attributes [Tryfona and Jensen, 1999]:

- Spatial Attributes
- Temporal Attributes
- Thematic Attributes

4.4.1.1 Spatial Attributes

Instances of spatial attributes represent the physical location and extension of an entity. In the simplest case a spatial attribute can be either a point, line, or polygon, but more complex geometric data types can also be defined, see

e.g. the simple feature specification in [ESRI, 2003], where also multipoints, multilines and multipolygons are included together with more complex types of polygons, e.g. polygons with wholes.

4.4.1.2 Temporal Attributes

A geographic entity and its relations to other entities are dependent on time. It exists in a given time period, as well as its relations to other entities, and its properties e.g. a house may have new owners, or its properties, e.g. a car may be black in a given period and afterwards painted blue. If these time-dependent properties should be represented, they must be included in the conceptual model. The STER [Tryfona and Jensen, 1999] approach suggests three temporal aspects to be included in the conceptual model:

Existence time is used to represent the period from which an entity begins its existence to it stops to exist.

Valid Time is the period in time when a relation between two entities is valid or the period when a value for an attribute is valid.

Transaction Time is the period in which an element, either an object, attribute, or a relations instance, is in the database.

As suggested in STER, GeoSML uses the abbreviations "et", "vt", and "tt" for **existence time**, **valid time**, and **transaction time** respectively.

4.4.1.3 Thematic Attributes

Apart from spatial and temporal attributes, a geographic object may be described by a number of thematic attributes, e.g. a building object which has attributes specifying the number of floors and total square meters.

4.4.2 Relationships

As in domain models conceptual models may include associations, taxonomic relationships, and mereologic relationships. Furthermore, topologic relationships are added to this list. Spatial properties and relationships are naturally of the greatest concern in relation to the specification of geographic information. Section 7.3.1 gives an introduction to the foundation of topologic relations.

4.4.3 Constraints

Constraints are rules that restrict the creation of objects from the conceptual model and are introduced to ensure that the produced information possesses the desired properties. Constraints may regulate both the values of attributes and the instantiation of relations among objects. Constraints can be grouped in five types according to their function:

- Topology constraint, e.g. buildings must not overlap
- Format constraint, e.g. a date attribute must follow a predefined pattern
- Metric constraint, the distance between two forests must be larger than 30 meters
- Domain constraint, e.g. the value of an attribute must be one of the values included in a specified list
- Temporal constraint, e.g. some change must happen prior to another

In reality, constraints may be a mixture of the five categories, and therefore it does not make sense to subdivide the markup in the specification. The basic GeoSML syntax, as introduced in the next sections, provides elements for introducing natural language constraints in the specification. Chapters 7 and 8 are dedicated to the formalization of constraints and introduce a formal constraint language that extended the GeoSML syntax to conceptual models. Furthermore, the five types of constraints and the formalization of these are treated in detail.

4.4.4 Syntax for the Conceptual Model

The syntax of conceptual models specified in EBNF is the following:

⟨conceptual model⟩::= **Conceptual Model**⟨name⟩
    (⟨object type⟩)*
    (⟨conceptual relationship⟩)*
    (⟨constraint⟩)*
    (⟨conceptual term⟩)*

⟨object type⟩::=**Object Type**⟨object typeID⟩[**ET**][**TT**]
    Name:⟨conceptual termID⟩

Attributes
(**Name:** ⟨attributID⟩⟨conceptual termID⟩ [**VT**][**TT**]
Data type:⟨datatype⟩)*

⟨conceptual relationship⟩::=⟨crID⟩⟨crdef⟩

⟨crdef⟩::=⟨cr-association⟩ |⟨cr-is-a⟩ |⟨cr-part-whole⟩ |⟨cr-spatialrelation⟩

⟨cr-association⟩::=**Association:**⟨typeID⟩⟨conceptual termID⟩⟨typeID⟩

⟨cr-part-whole⟩::=**Part-whole relation:**(⟨typeID⟩)* **is part-of** ⟨typeID⟩

⟨cr-is-a⟩::=**Taxonomic relation:** (⟨typeID⟩)* **is-a** ⟨objecttypeID⟩

⟨cr-spatialrelation⟩::=**Spatial relation:**⟨named spatial relation⟩ |⟨defined spatial relation⟩

⟨named spatial relation⟩::=⟨typeID⟩⟨topooperator⟩⟨typeID⟩

⟨defspatialrel⟩::=⟨typeID⟩⟨topodef⟩⟨typeID⟩

⟨constraint⟩::= **concerning** (⟨object typeID⟩)***:** ⟨statement⟩

⟨conceptual term⟩::=**Conceptual term** ⟨conceptual termID⟩⟨term⟩

⟨term⟩::=⟨string⟩

⟨statement⟩::=⟨string⟩

⟨datatype⟩ ::= **string** | **bool** | **float** | **date** | **integer** | **point** | **polyline** | **polygon**

⟨topooperator⟩::=**overlap** | **touch** | **contain** | **inside** | **covered by** |**cover** |**disjoint** |**equal**

⟨topodef⟩::=
11:⟨val⟩ **12:**⟨val⟩ **13:**⟨val⟩
21: ⟨val⟩ **22:**⟨val⟩ **23:**⟨val⟩
31:⟨val⟩ **32:**⟨val⟩ **33:**⟨val⟩

⟨val⟩::=**0** |**1** |**2**

As seen some of the grammar is analogous to the one used to specify domain models (see Section 4.3). The major differences are that ⟨assertion⟩, ⟨designation⟩, and ⟨definition⟩ are omitted, and the ⟨constraint⟩ is introduced. Constraints can only be formulated in natural language in this version of GeoSML. Chapter 8 adds the syntax for specifying formal constraints. Furthermore, topologic relations are added to the list of possible types of relations (the ⟨spatialrelation⟩ symbol).

4.4.5 Conceptual Model - Example

To illustrate the usage of the grammar give supply an example based on the domain model developed in Section 4.3. In the example the conceptual model has four classes: Area, Residential Area, Commercial Area, and Building. Area is the superclass of residential area and commercial area. Residential area has two attributes: Residential area type and number of residents. Building has also attributes: Building type and number of floors. There are two topologic relations between area and building: contain and overlap. The contain relation indicates which buildings are contained in an area, and overlap indicates if there is an overlap between the boundary of an Area and the boundary of a building.

```
Conceptual model My Small Topographic Map
    Conceptual term building: building
    Conceptual term building_outline: building_outline
    Conceptual term construction_year: construction_year
    Conceptual term building_type: building_type
    Conceptual term number_of_floors: number_of_floors
    Conceptual term hight: hight
    Conceptual term road_segment: road_segment
    Conceptual term centerline: centerline
    Conceptual term road_type: road_type
    Conceptual term road_name: road_name
    Conceptual term area: area
    Conceptual term extent: extent
    Conceptual term area_type: area_type
    Conceptual term residential_area: residential_area
    Conceptual term number_of_residents: number_of_residents
    Conceptual term address: address
    Conceptual term house_number: house_number
    Conceptual term municipality: municipality
    Conceptual term municipality_name: municipality_name
    Object Type building
        name: building
        Attributes:
            building_outline building_outline polygon
            construction_year construction_year integer
            building_type building_type string
            num_of_floors number_of_floors integer
            height height float
    Object Type road_segment
        Name: road_segment
```

```
    Attributes:
        centerline polyline
        road_type road_type string
        road_name road_name string
Object Type area
    Name: land-use area
    Attributes:
        extent extent polygon
        area_type area_type string
Object Type residential_area
    Name: residential_area
    Attributes:
        num_of_residents number_of_residents integer
Object Type commercial area
    Name: commercial area
Object type address
    Name: address
    Attributes:
        house_number house_number string
Object type municipality
    Name: municipality
    Attributes:
        municipality_name municipality_name string
Taxonomic relation:
commercial_area and residential_area is-a area
Spatial relation: contain area contain building
Spatial relation: ab_overlap area overlap building
Spatial relation: bb_overlap building overlap building
Spatial relation: br_intersect building intersect road_segment
Spatial relation: ar_intersect  area intersect road_segment
Spatial relation: touch building touch building
Association: isaccessedfrom address isaccessedfrom road segment
Association: gives_access_to address gives_access_to building
Association: belongsto address belongsto municipality
```

4.5 Mapping Domain and Conceptual Models

The third part of the GeoSML concept is descriptions that relate elements in the domain model with elements in the conceptual model. By applying a map between entity types and object types a bridge between descriptions of the domain, which is constituting the ontological commitment of the specification,

and the design of a data collection is created. Thus, the descriptions of terms in the domain model are bound to descriptions of objects in the data collection, and enables designers to describe, how entities and relations in the domain actually should be represented in the data collection.

Mappings between a domain model and a conceptual model are facilitated by the `is-represented-by` relation, which is used to indicate that an `entity type` is represented by an `object type`, e.g *greenhouse is represented by building*. Binding the models together ensures that the design of a data collection builds on a well-formed ontology, and that the chosen representation schema can fulfill the stated requirements.

The `is-represented-by` relation not only binds entity types together with object types, but also provides a syntax for describing the interpretation of domain entities to form objects in the data collection, hence these rules are called interpretations rules. Interpretation rules define correct instantiation of objects by stating conditional relationships between observable properties in the domain and the representation of these. The rules formally follow the pattern:

$$\forall x_1 \dots x_n (\Psi(x_1 \dots x_n) \rightarrow \exists y_1 \dots y_m (\Phi(y_1 \dots y_m, x_1 \dots x_n)))$$

where $\Psi(x_1 \dots x_n)$ is a proposition over the geographic entities $x_1 \dots x_n$ and $Phi(y_1 \dots y_m)$ is a proposition over the geographic entities $y_1 \dots y_m$ and the geographic objects $x_1 \dots x_n$. If something holds true for the domain then something must hold true for the representation of the domain.

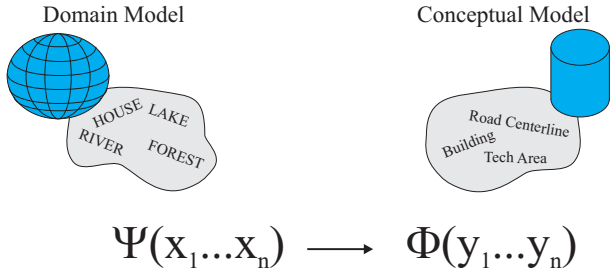

Figure 4.2: Pattern for mapping domain and conceptual models.

An example of a formula following this pattern is

$$\forall \text{ b1,b2 building}(b_1) \text{ building}(b_2)(\text{isneighbors}(b_1,b_2) \wedge \text{heightdiff}{<}5(b_1,b_2) \rightarrow \exists \text{ b (Building(b)} \wedge \text{represent(b,}b_1) \wedge \text{represent(b ,}b_2)$$

The above statement is to be read as *for all building, b_1 and building, b_2, if b_1 and b_2 are neighbors and the height difference between b_1 and b_2 is smaller than 5 meters, then there must exist a building, b where b represents b_1 and b represents b_2.*

In other words, if two buildings are neighbors and have a height difference less than 5 meters, then they must be represented by a building in the data collection. The "represent" predicate in the above formula is true if an object exists that represents an entity (and all constraints and representation rules are fulfilled).

Representation relations are in practice hard to specify, and even harder to implement. In general they, should be specified in natural language and leave the formalization to the constraints. In Chapter 8 it is shown how interpretation rules can be translated into constraints formulated in terms of conceptual models, and used to ensure that the produced information observes the interpretation rules.

It is possible to categorize interpretation rules according to the role they play in the interpretation process. In Section 3.4.1 we recognized five types of statements:

- Classification rules
- Selection rules
- Instantiation rules
- Representation rules
- Life Cycle rules

4.5.1 Classification Rules

Classification rules often have characteristics equal to definitions, e.g. "a building is a man-made construction used to facilitate human activities, such as living, production of goods, and sports". In this statement the term building is defined and it can be used to identify geographic entities that candidates for a representation in one or more data sets.

4.5.2 Selection Rules

The purpose of **selection rules** is to restrict the set of entities that should have a representation in the data collection. Selection rules can be formulated as rules that explicitly include entities with specified properties in the data collection (**inclusion rules**), or as rules explicitly excluding entities with specified properties from the data collection (**exclusion rules**). An example of a selection rule concerning roads is: *"roads shorter than 50 meters should not be given a representation, unless they connect two or more roads in the road network"*.

Inclusion and exclusion rules often describe required properties of entities in order to determine whether they should be represented in the data collection, but can also state restrictions on relations to other entities, i.e. "BUILDINGS in LAKES must be excluded".

4.5.3 Instantiation Rules

Instantiation rules describe how entities are instantiated as objects. It is not always the case that a one-to-one relation between entities and objects exists. A road is identified by its road name, while the objects representing the road are often divided in segments going from intersection to intersection. In the case of roads there are one-to-many relations between entities and objects. In other cases a set of entities is aggregated as one object. The motivation of aggregation is to reduce the work and thus the cost to instantiate a large set of entities, or if it is difficult to determine the exact boundary between two entities.

Examples from the TOP10DK specification are: *Two buildings must be instantiated as one building if they are neighbors and the height difference is less than 5 meters* and *two fur farm buildings must be instantiated as one building if the distance between them is smaller than 2.5 meters.*

In some cases it is difficult to distinguish between selection and instantiation rules. Take one of the above examples: *Two Buildings must be instantiated as one building if they are neighbors and the height difference is less than 5 meters*. If two buildings are neighbors and both have an area smaller than $25m^2$, but together have an area larger that $25m^2$, how must the two buildings be represented? Given the two rules it is clear that the two building should not be given an representation. However, if the number of buildings is ten instead of two, e.g. a group of garages. Then the total area of all buildings is much larger than the minimum requirement for one building, and we really want to have a representation of the buildings. Adding a selection rule we can ensure that:

Neighboring buildings of a total area larger than 50m^2 must be represented. Note that if a building of an area of 10m^2 is neighboring a building with an area of 50m^2, then both buildings should be represented, but instantiated as one `building` (according to the instantiation rule).

4.5.4 Representation Rules

When it has been determined how an entity or a group of entities is instantiated as an object, each attribute of the object must be assigned a value. The exact value is determined by applying a set of **representation rules**. For simple attributes the representation rules are often given in the definition of the corresponding property, e.g. if the construction year of a building is defined to be the year in which the construction of the building started, then it is obvious which value should be assigned to each building object.

For more complex attributes, and particularly for attributes whose values are geometric data types, several representation rules may be necessary to describe all the information needed to correctly assign values to an attribute.

For attributes of the geometric data type, registration rules often describe how generalization of the geometric representation is to be made, or how the different data sources are interpreted in order to extract the wanted information. For example an outline of a building is drawn from a digital image or a classification of vegetation is made. These rules are often difficult to formalize or even express in natural language. Therefore, illustrations or pictures can be used to describe a registration rule. An example of a registration rule is: *The corners of buildings must be registered with 90-degree angle if this does not compromise the requirements for a precision of 1 meter.*

4.5.5 Life Cycle Rules

When a domain changes from one state to another it may implicate a need for updating the representation of the domain. How and when these updates should be done is decided by a set of **life cycle rules**. The key question for life cycles rules is to decide wether a change in the state of a geographic entity is identity preserving or not. In other words, the task of life cycle rules is to define the criteria for identifying an entity as the same at two different points in time. An example of a life cycle rule is: "If building parts are added or removed from a building, then its identity is preserved only if the area of the resulting building has changed less than 30%.

4.5.6 Grammar for Mappings between Domain and Conceptual Models

The syntax for the "is-represented-by" relations in EBNF:

⟨repexp⟩::=⟨entity typeID⟩ **is represented by** ⟨typeID⟩
using (⟨selection rule⟩)*
(instantiation rule)*
(⟨life cycle rule⟩)*
Attribute Values::=(⟨attributID⟩ [**has default value** ⟨value⟩)] **is determined by** (⟨representation rule⟩*) |
⟨drID⟩ **is represented by** ⟨crID⟩
using (⟨selection rule⟩*)
(⟨instantiation rule⟩*)

⟨selection rule⟩::=⟨statement⟩

⟨instantiation rule⟩::=⟨statement⟩

⟨life cycle rule⟩::=⟨statement⟩

⟨representation rule⟩::=⟨statement⟩

Three sets of rules and a set of attribute value descriptions are applied to each mapping between an ⟨Entity Type⟩ and an ⟨Object Type⟩. The sets of rules include selection rules, instantiation rules, and life cycle rules. The ⟨Attribute Value⟩ part includes an optional default value and a set of representation rules for each attribute.

The `repexp` symbol is included in the conceptual modeling level.

4.5.7 An Example

To exemplify use of the mapping relation the following example is given. The example binds the domain model from Section 4.3 and the conceptual model from the previous section together.

```
Greenhouse is represented by Building
    Selection Rule
```

```
        The area of greenhouses must be larger than 25m2
        Attribute Values:
             Type is set to 'greenhouse'

Road is represented by road_segment
    setting roadtype to 'road'

Cycle path is represented by road_segment
    setting roadtype to 'cycle path'

Residential_building is represented by Building
    Attribute Values:
        type is set to 'residential'
        building outline is represented using
        Representation Rules:
            building outline is registered on the roof
            overhang/eaves of the building
        Selection Rules
            The area of building must larger than 10m2
        Instantiation Rules
            Two neighbor buildings of a height difference
            smaller than 5m must be represented as one building
        Representation Rule:
            building outline must be registered as closed
            polygons with a common start and end point.
        Representation Rule:
            building outline must be registered on the outer
            extremity of the foundation or ruin.
        Representation Rule:
            buildings outline must be registered with as few points
            as possible but so that the difference between the actual
            sequence and the registered sequence is never larger than
            1 m in plan and elevation.
        Representation Rule:
            All Building corners must be registered. However,
            buildings with overhangs and extensions of a side length
            of less than 3m and an area smaller than 10m2 must
            not be registered.
        Selection Rule:
            Agricultural buildings smaller than 100m2 in connection
            with farms that are considered to be used for habitation,
            must not be registered.
        Representation Rule:
            Houses built together must be registered as one building.
```

4.6 Summary

In this chapter a representation system for denoting phenomena in a domain and their representations as geographic entities has been introduced and a new principle for structuring specifications for geographic information written in natural language has been suggested.

The presented language developers are able to write specifications according to a predefined structure. By dividing a specification into the three suggested parts, it is clearer which part of the reality should be represented in the data collection being developed and how this representation should actually be.

The GeoSML approach to describing the domain is primarily informal, and only a few language elements are included to express formal properties of entities and objects. GeoSML as it is introduced in this chapter has little support for formalizing specification statements. The expressiveness of the formalism introduced to describe the domain may be insufficient for modeling all properties in the domain formally.

Chapter 5

Requirement Specification

Abstract: This chapter introduces the notion of requirement models into the GeoSML framework. By simple syntactic constructors, designers of geographic information are enabled to capture and structure the users needs for geographic information. The suggested approach to modeling user requirements is inspired by the KAOS requirement model [van Lamsweerde, 2001] and [van Lamsweerde et al., 1991]. Requirements are formulated as goals that the users want to achieve using a data collection. Goals can be refined or abstracted by ordering them in and/or trees. The major contribution from this chapter is specification elements that relate requirements to terms in the corresponding domain model and thus ensure that requirements are built on a formulated domain theory. Furthermore, requirements are related to elements in the conceptual model. By relating requirement and conceptual models, the motivation of design decisions formulated in conceptual models is documented.

5.1 Introduction

There is a gap between the way users state requirements and needs for geographic information and the way developers design data collections. While users specify their needs in terms of the domain in which they operate and goals they want to reach using the data collection, designers design data collections in terms of the object types, attribute types, and relationships they want to include in the data collection. To close this gap a framework for modeling user requirements, with the following properties, is suggested:

- requirements are formulated as goals
- goals are built by use of the terminology defined in a corresponding domain model
- goals can be refined by using refinement trees
- goals can be assigned to elements in a corresponding conceptual model

The elements in the framework are inspired by and partly based on the GOAL requirement model (see Section 2.3). While Lambsweerde's method is oriented toward a formalization of requirements, we use the idea of organizing requirements as and/or trees, and especially the idea of delegating the responsibilities for implementing goals to agents, human or computer-based, is of great value. The formalization is in our approach left to be formulated as constraints specified in the context of a corresponding conceptual model (see Chapter 8).

The contents of this chapter are organized as follows. Section 5.1.1 discusses the types of requirements for geographic information. Section 5.2 explains the approach for requirement engineering. Section 5.3 extends GeoSML to include syntax for structuring requirements and to related requirements to other specification elements, such as interpretation rules, object types, attributes, and constraints. Section 5.4 continues the running example by illustrating the usage of the language elements for defining and structuring requirements. The chapter is concluded by a summary.

5.1.1 Requirements for Geographic Information

The purpose of using geographic information and geographic information systems is in general to support the users in making spatial decisions according to predefined tasks. Examples of tasks are to transport some goods from one

location to an other, to manage a water supply system, or to deploy resources to prevent or reduce the consequences of natural catastrophes. All tasks are carried out to achieve a particular goal or set of goals. For the task of water supply system management these goals could be: *"All consumers must be supplied with the required amount of water"* and *"the water quality must be within the acceptable ranges"*. To operationalize these kind of high-level goals, they must be refined to goals with a lower level scope. For the first of the above examples this could be done as *"the capacity of the installed water pumps must be double the nominal water consumption"* and *"major nodes in the water supply network must have alternative sources"*. To relate these goals to requirements for geographic information, they must be further refined. For example the first of the above refined goals can be further refined as *"the capacity of all water pumps must be recorded"*, *"the estimated outtake must be recorded for each building which is supplied with water"* and *"the structure of the water supply system must be known"*.

Generally speaking, the goals directly stating requirements to a data collection can be divided into the following categories:

- Data content
- Topology and data structure
- Scale and resolution
- Cartographic design and representation
- Frequency of updates
- Quality

The usability of geographic information is first of all decided by the **data content** in terms of object types, attribute types and relationships. Also the **topology and data structure** are important to the application of a data collection. For example, if a data collection must be used for trip planning it is imperative to have access to a set of road segments forming a topologic network.

Requirements related to **scale and resolution** depend on which detail entities must be represented. If the goal for a task is to measure distances between buildings with a one-meter precision, then all details must be included in polygons representing the extents of the buildings.

Cartographic design and representation may also implicate the usability of a data collection. While data content and resolution are invariant properties

for a data collection, cartographic design and representation can be changed from one application to another.

The **quality** contributes to the usability of a data collection. The most important aspects of quality are completeness and accuracy. Some applications are sensitive to the completeness of a data collection, such as address databases used for dispatching ambulances, for others the accuracy is also important, e.g. databases describing the positions of pipes used for transportation of gas. Requirements related to quality are not treated as goal-oriented requirements, but as an integrated part of the conceptual model. Chapter 6 discusses in detail how requirements in relation to quality are introduced in a GeoSML specification.

A parameter correlated to quality is the **frequency of updates**. To ensure that a data collection is complete, it must be updated regularly. The more often a data collection is updated, the more actual is the information in the data collection. This is also a parameter in the evaluation of the usability of an data collection. Some applications require information which is updated often, e.g. weather forecasts and management of infrastructures, other applications need less often updated information.

5.2 Requirement Engineering

This section discusses aspects of requirement engineering important to the context of GeoSML and the specification of geographic information. First, the notions of requirements and goals are discussed, second, the structuring of goals is discussed, and third the relations between corresponding requirement, domain, and conceptual models are described.

5.2.1 Requirements and Goals

According to Jackson [Jackson, 1997] a requirement means *the effect in the problem domain that the users want the system to guarantee...*, in our case a geographic data collection. Requirements are in this way the functionality that some geographic information used in an appropriate geographic information system must deliver.

To capture the requirements for a system, various approaches can be adopted, e.g. investigating background papers, such as project proposals, cooperate business strategies, and work flow descriptions. Enquiring domain experts and users

is important in this process. Users know much more about solving problems and carrying out tasks within their specific domain than about designing geographic information. Therefore, to invite users to participate in a design process means that requirements should be formulated in terms of the user's domain.

Formulating requirements as goals is an effective method for doing this. By enforcing the users to thing in goal ensure that the collected requirements are directly inherited from actual needs and that they support the existing business processes, - implemented or being developed.

5.2.2 Structuring Goals

A goal can be refined as a subgoal or a set of subgoals, e.g. the goal *"the cost of distributing goods to customers must be minimized"* can be refined as the subgoals *"the travel distance must be minimized"* and *"the travel time must be minimized"*.

Goals can be related by a tree-structure where the most general requirements are situated in the top of the tree and more specific requirements at the bottom of the tree. The "requirement tree" is organized so that subrequirements state how their parent requirement should be fulfilled, and vice versa the parent requirement explains why its subrequirements must be met. The tree structure of the requirement model is implemented by adding a list of subrequirements to a requirement.

The innovative idea of KAOS is to assign the responsibility of implementing goals to objects, which can either be agents, entities, events, or relationships. The approach adopted to stating and structuring requirements is inspired by the KAOS model, where requirements are formulated as goals the system must ensure that the user can reach by using the system.

Goals may be conflicting. This means that the satisfaction of one goal may prevent other goals from being fulfilled. In GeoSML conflicting goals are related.

Goal A goal is a state of affairs or a state of a concrete activity domain which a person or a system is going/tends to achieve or obtain.

Subgoal Goals can be refined as a set of subgoals. This is done either by linking goal and subgoals by "and-links" or by "or-links".

and-link If a goal is related to a set of subgoals through and-links, it means that the goal is achieved if all subgoals are achieved.

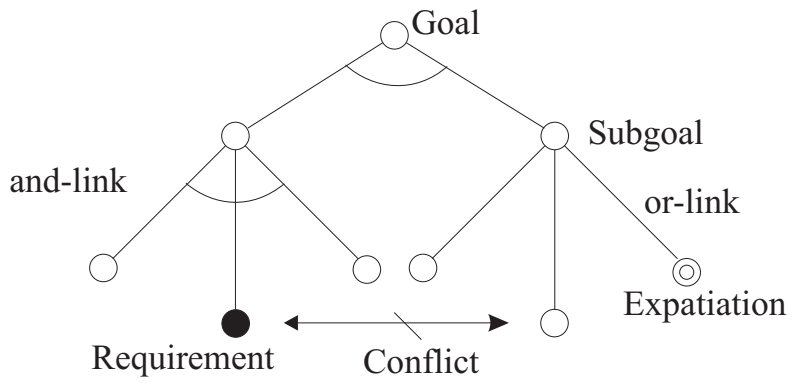

Figure 5.1: Concepts of an "and/or" goal tree.

or-link If a goal is related to a set of subgoals through or-links, it means that the goal is achieved if one of the subgoals is achieved.

Requirement A requirement is a goal that has been assigned to an element in a conceptual model.

Contradiction Two goals are said to be contradictive, if the fulfillment of one of the goals excludes the fulfillment of the other goal.

Expectation An expectation is a goal which is expected to be achieved by an agent in the domain. In the case of specifying requirements for geographic information, this agent is likely to be a function in a geographic information system.

Figure 5.2 shows how an and/or goal tree can support the organization of goals. The most general goal states that the system being developed must support the transportation of goods between distributions centers and retail stores. This goal is refined as two subgoals linked by and-links. The two subgoals state that the higher level goals can be achieved if the location of the distribution center and the location of the store are known. The two subgoals are further refined by or-links, to two pais of subgoals stating that the locations can be found by either an address or a coordinate.

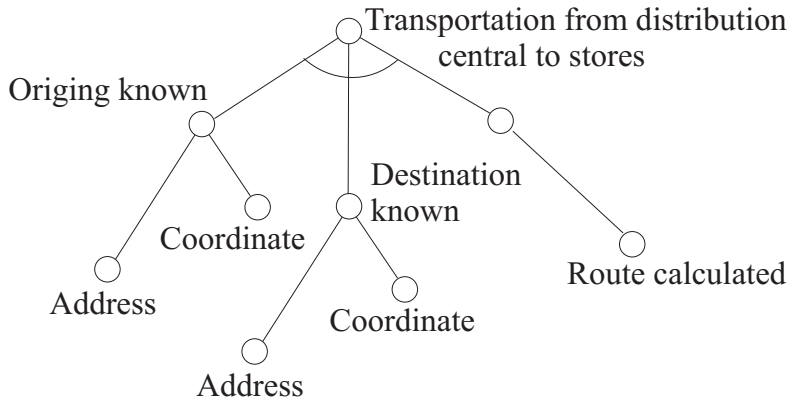

Figure 5.2: Concepts of an "and/or" goal tree.

5.2.3 The *Implement Relation*

The relation between corresponding domain, requirement, and conceptual models can theoretically be described by the *implement relation* [][Jackson, 1997]: Let $\mathcal{D}$, $\mathcal{R}$, and $\mathcal{S}$ stand for a domain model, a related requirement model, and a conceptual model. For $\mathcal{S}$ to be an implementation of $\mathcal{R}$ - based on $\mathcal{D}$ - the following relation must hold:

$$\mathcal{D}, \mathcal{S} \models \mathcal{R}$$

In other words, a system design, represented by a conceptual model, must implement the requirements stated in a requirement model and be based on the domain knowledge captured by the related domain model. It is in fact very difficult to prove that an implementation meets all requirements stated in a requirement model, and that it represents the domain facets in an appropriate way. To conduct a formal prove that the *implement relation* holds for a given domain model, requirement model, and conceptual model is difficult - if not impossible.

It could be argued that if the implement relation could be mathematically proved, the consequence would be that a design could automatically be derived from the domain and requirement models. We believe that design of geographic information requires human interaction to make efficient designs. Therefore,

domain descriptions, requirements, and design are handled in separate and integrated models.

The integration of goals is facilitated by two element in the specification language. (i)Goals are formulated in the context of terms existing in the problem domain, and are thereby bound to the theory of the domain. In practise this is done by listing the domain terms that are relevant for a given goal. (ii) Requirements are also strongly related to the conceptual model, hence the design decisions expressed in the conceptual model depend and is motivated by requirements in the requirement model. The knowledge of which requirements motivate which design elements is captured by assigning requirements to the elements in the conceptual model.

5.3 Grammar for Requirement Models

In this section a grammar that builds on the theory of goal-orientation of user requirements and the above analysis on requirements for geographic information. The grammar includes elements for structuring requirements and relating these to corresponding domain and conceptual models.

Requirement model:::= ⟨requirementmodel⟩

⟨requirementmodel⟩::=(⟨goalexp⟩)*
(⟨contradiction⟩)*
(⟨goalassign⟩)*

⟨goalexp⟩::=⟨goal⟩[**is refined as** ⟨operator⟩⟨goalexp⟩*

⟨goal⟩::=**goal**⟨goalID⟩[**concerning** (⟨domain termID⟩)*]⟨statement⟩

⟨contradiction⟩::=⟨goalid⟩ **is conflicting** ⟨goalid⟩*

⟨operator⟩::=**and** |**or**

⟨goalassign⟩::=⟨element⟩ **is motivated by** ⟨goalID⟩*

⟨element⟩::=⟨object typeID⟩ |⟨attribute typeID⟩ |⟨constraintID⟩ |⟨representationruleID⟩

The top symbol in this grammar is ⟨Requirement model⟩. This symbol marks the beginning of a requirement model in a GeoSML specification. A ⟨Requirement model⟩ is defined as three sets: a set of ⟨goalexp⟩ (short for *goal expression*), a set of ⟨contradiction⟩, and a set of ⟨goalassign⟩ (short for *goal assignment*).

The ⟨goalexp⟩ symbol is used to define refinement trees by adding the optional *"is refined as"* which has an ⟨operator⟩ and a set of ⟨goalexp⟩ on the right hand side. The recursive definition of ⟨goalexp⟩ enables the creation of and/or trees having ⟨goal⟩ as nodes. The ⟨goal⟩ symbol defines the format for specifying goals that can be included in a refinement tree and is defined as an identifier, an optional list of keywords from a domain model, and a ⟨statement⟩. The ⟨statement⟩ symbol is defined in Section 4.4.4 and is the carrier of the text describing a particular goal.

The ⟨contradiction⟩ symbol is used to mark two goals as conflicting. The ⟨contradiction⟩ symbol can be is a symmetric binary relation between two goals. Therefore, if it is stated that $goal_1$ is contradictive to $goal_2$ then it is given that $goal_2$ is also contradictive to $goal_1$.

The ⟨goalassign⟩ symbol are used to describe relations between goals in the requirement model and a corresponding conceptual model. This relation is facilitated by the *is motivated by* relation, which has an `element` from the conceptual model on the left-hand side and a ⟨goalID⟩ on the right hand-side. An ⟨element⟩ is a reference to either an object type, an attribute type, a constraint or a representation rule.

5.4 Example of Modeling Requirements

To illustrate the usages of GeoSML grammar for specifying requirement models three requirement models. It must be emphasized that the models to some extent are "naive" and that their main purpose is to show have the grammar is used. The examples should be self-explanatory.

```
Requirement model: building permits.
    1  Application must be analyzed to determine if the filled
    application can be approved or not.
    1.1 No requirements in the small-build regulation must be broken
    1.1.1 No building parts must be taller than 8.5 meters.
    and
    1.1.2 The area of a building must be smaller than 25% of the areal
    on which the plan building will be situated.
    or
    1.2 a dispensation must be given for the rules that are not meet.
Requirement model: school district planning
    1 School district must be
    planned each year
```

```
        Goal 1.1 The route which the students travel to school
            must be the best possible
            Goal 1.1.1 The students must have as short way to
                school as possible.
                Goal 1.1.1.1 Home addresses and school locations
                    must be recorded
                and
                Goal 1.1.1.2 Road network must be available
            and
            Goal 1.1.2 The way to school must be as safe as possible
                1.1.2.1 The student must crosses as few roads
                    as possible
                and
            Goal 1.1.2.2 crosses no major roads
                1.1.2.1.1 There must be access to the classification
                    of roads
        and
        Goal 1.2 The number of elver must be relatively similar
            each year
        and
        Goal 1.3 The school districts must be as stable as possible

    1.2 is conflicting 1.3
    1.1.1 is conflicting 1.1.2

    1.1.1.2 is implemented by Roadnetwork
    1.1.1.1 is implemented by Address
    1.1.2.2.1 is implemented by Roadtype

Requirement model: navigation
    Goal 1 The system must enable pedestrians, car drivers,
        and cyclists to find their way around
        Goal 1.1 It must be possible to calculate the travel
            distance and time between two locations
        and
        Goal 1.2 Generation of traveling directions based on
        road names the leads like "turn left at next intersection"
        and "use the second leg in the roundabout".
        and
        Goal 1.3 Start points and destinations should be found by
        addresses, place names, and points of interest,
        like hospitals and museums.
```

5.5 Summary

In this chapter a method for stating and organizing requirements for geographic data collections is developed. GeoSML, which was introduced in the previous chapter, has been extended with grammar elements for including what is called a *requirement model*.

In the proceeding chapter requirements for the quality of geographic information will be investigated in further detail, and the conceptual model will be extended to include the specification of acceptable quality levels and the quality parameters of a data collection will be presented.

Chapter 6

Specifying Quality Requirements

Abstract: This chapter extends GeoSML with structures for specifying requirements related to quality. Two concepts are introduced: (i) Acceptable quality level (AQL), defining requirements for minimum and maximum values of quality parameters, such as precision and completeness, and (ii) quality element requirement (QER), which specifies the quality parameters that must be used to describe the quality of a data collection. The work presented in this chapter was carried out in collaboration with Anders Friss-Christensen and published at the International Symposium on Spatial Data Handling 2004 [Friss-Christensen and Christensen, 2004]. The originally introduced work was made as extensions to the metamodel of the Unified Modeling Language. In this chapter the language is defined as a context-free grammar and adjusted to the GeoSML framework.

6.1 Introduction

A geographic data collection is by nature an abstraction of the domain sought to be represented and therefore "only" offers approximate descriptions of the entities within the domain. Quality descriptions are needed to determine *how good* the approximation actually is. Explained in the context of the representation model presented in Chapter 4 (see Figure 4.1 on page 46) quality is related to the *identify* relation between *geographic entity* and *geographic object*. The quality can be regarded as the loss of information going from the "real" entity to its representational counterpart.

Users who need certain object types from a certain area should receive additional quality information customized to their specific application. Hence, quality information is metadata for instances of object types. This differs from the traditional approach where quality information is provided at a more general and higher level. Typically, the fitness for use is mainly evaluated by studying metadata, but there is a need for a more dynamic approach to quality management in which quality information is an integrated part of the geographic data model. Here, quality can be assessed directly when sources of information are selected, as opposed to having to study a separate and general metadata report.

Producers of geographic information assess the quality to ensure that the produced information conforms to the underlying specifications, and users need quality evaluation reports to determine *how good* the information actually is, and to decide if the information is appropriate for a given application.

The contributions of this chapter are twofold: First, we identify and describe quality elements that are necessary for adequately describing the quality of any geographic data collection. The elements identified are based on requirements from the National Survey and Cadastre, but should satisfy most applications that utilize geographic information. Second, we extend GeoSML to incorporate geographic information quality elements. The result is a framework that enables designers and users to specify quality requirements in a geographic data model. Such a quality-enabled model supports a more application specific distribution of geographic information, e.g. one using web services.

Geographic information quality has been a research topic for more than a decade, and several aspects of data quality in relation to GIS have been in focus. The fundamental elements of quality have been investigated in previous work [Chrisman, 1984, Goodchild and Gopal, 1989, Guptill and Morrison, 1995, Veregin, 1999]. Furthermore, the International Standardization Organization's TC211 is close to the release of standards for geographic information quality [ISO, 2001b, ISO, 2001a]. Its work focuses on identifying, assessing, and

reporting relevant quality elements for geographic information, but does not consider integrating information quality requirements into conceptual models. We believe that conceptual modeling is important when making the management of geographic information quality operational. Another aspect of geographic information quality is error propagation [Heuvelink, 1998], which is the description of how quality and errors evolve in GIS analysis. Whereas error propagation focuses on determining the quality when calculations are applied to the information, we focus on the modeling of the fundamental elements of quality. However, our work establishes a framework which can be used in error propagation in GIS.

The remainder of the chapter is organized as follows. Section 6.2 presents the quality elements and investigates how to classify quality requirements. It also describes how quality is assessed in a production process. Section 6.3 develops a framework for modeling quality by extending GeoSML to include quality elements. In section 6.4 the running example is used to illustrate how requirements to quality can be included in conceptual models.

6.2 Quality of Geographic Information

The notion of geographic information quality is a complex and open-ended concept covering a wide range of characteristics related to the perception and usage of geographic information. The approach taken in this chapter seeks to operationalize the quality of geographic information by suggesting that the quality of a geographic data collection can be described by a finite set of quality elements. A set of quality elements is termed *quality information*. This section describes each such quality element. Furthermore, the relevance to applications of geographic information is explained. Finally, we describe how quality can be assessed as part of a production process.

6.2.1 Quality and Quality Requirements

Two overall approaches may be adopted to describe quality: A user-based approach and a product-based approach [Garvin, 1988]. The former emphasizes fitness for use. It is based on quality measures evaluated against user requirements dependent on a given application. The product-based approach considers a finite number of quality measures against which a product specification can be evaluated. While the product specification is based on user requirements, it is often a more general specification that satisfies the needs of a number of dif-

ferent customers. The ISO 9000 series (quality management) defines quality as *the totality of characteristics of a product that bear on its ability to satisfy stated and implied needs* [ISO, 1994]. This definition is used in the ISO standards on quality of geographic information [ISO, 2001a, ISO, 2001b], and it covers both the user-based and the product-based approaches.

As quality is a relative measure, users and database designers are faced with two challenges: how to specify their quality requirements and how to determine whether the information satisfies the requirements. The first challenge concerns the specification of implied costumer needs, which are often not expressed directly. Introducing quality elements in conceptual models is helpful in meeting this challenge, because it then becomes easier to capture quality requirements in the design phase. The second challenge concerns the possible difference between actual and required quality of the information. A solution is to make it possible for users to evaluate their requirements against the information, by including quality information about each object, attribute and association in the data collection.

To be able to support quality requirements in conceptual models the characteristics of such requirements must be identified. In doing so, the requirements are divided into two. First, it should be possible to specify which of the quality sub-elements are relevant to a given application. These requirements are termed *quality element requirements*, or QER. Second, it should be possible to express requirements related to the values of the actual quality, i.e. express specifications or standards that the information should satisfy. These requirements are termed *acceptable quality level*, or AQL. The two levels of requirements are similar to the notions of information quality requirements and application quality requirements [Wang et al., 2001].

The quality element requirements can be characterized as the necessary elements making it possible to assess and store quality information. Thus, quality element requirements are at a detailed design level. At a more abstract level, we find requirements such as 90% of all objects must be present in a data collection, which is the acceptable quality level. It is important to be able to separate these two levels of requirements. The acceptable quality level does not necessarily have to be specified, but if any quality information is needed, the quality element requirements have to be specified.

Figure 6.1 shows the two levels. The requirements are specified as user-defined, which means that the designers are specifying the requirements, but they should reflect the requirements from the application users. As seen in the figure, the quality elements require that different evaluation methods are assessed: internal and external methods, which methods are described in Section 6.2.3.

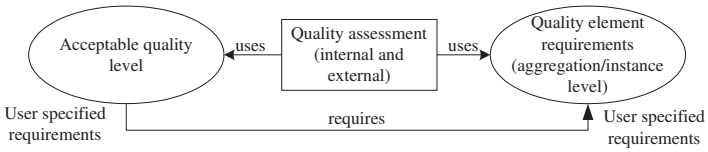

Figure 6.1: Quality Requirements.

In the internal approach, quality is assessed immediately when information is inserted in the database using internal tests and, hence, requires no external source to be carried out. For example, topologic constraints are evaluated immediately after information has been inserted into the database. If the information does not satisfy the constraints it is rejected. In the external approach, information is checked against an external reference that is believed to be the "truth." For example, if 90% of all forest objects are required to be present in the information, this has to be measured against some kind of reference. Hence, quality elements assessed by external methods need to be assessed at a later stage, by means of an application running on top of the database.

Finally, the quality elements are divided into two levels: the aggregation level and the instance level. These levels refer to whether quality is measured/specified for each instance of an object type or the quality is aggregated at, e.g. an object type level. It should be possible to specify requirements related to both levels. As an example, a building object (instance level) has a spatial accuracy which is different from the aggregated spatial accuracy of all buildings (aggregation level). Both levels may be important to an application user.

6.2.2 Quality Elements

This section describes elements of quality which are adequate to compose a quality report that can be used in applications of geographic information. These elements are covered in most descriptions of geographic information quality. Existing work characterizes quality [ISO, 2001a, ISO, 2001b, Veregin, 1999, Guptill and Morrison, 1995], but does not completely agree on the elements being relevant and necessary. However, they agree on most quality elements. We include the quality elements specified by ISO. However, in addition we include precision and metric consistency [Guptill and Morrison, 1995]. The choice of which elements to include is based on requirements from the National Survey and Cadastre, but should be applicable to geographic information in general. In Figure 6.2, quality information is divided into elements and subelements.

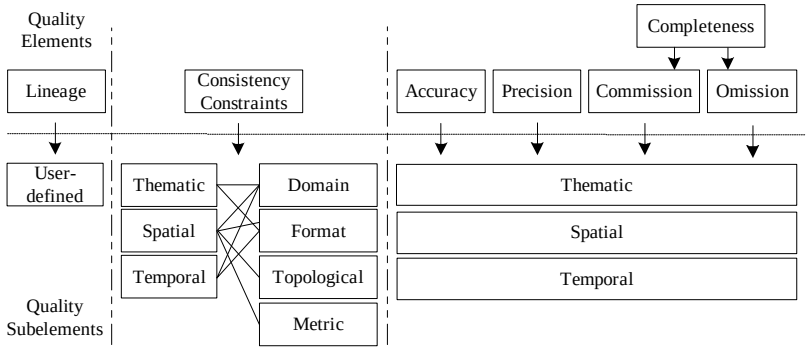

Figure 6.2: Quality information, elements, and subelements.

In the following sections each quality element is described in further detail.

6.2.2.1 Lineage

Lineage is a qualitative aspect of quality and refers to the process of recording and keeping track of the origins of the information. All elements of a geographic object may have lineage information, e.g. lineage can be specified for both objects and attributes. Lineage information is related to the measurement and creation of the information and may contain information about the person or company that has registered a given object or attribute. However, it can also be information stating that an object is derived from, e.g. another data collection. Lineage is used as a basis when expressing how reliable the information is. The actual recording of lineage information and the kind of information to be stored depends on the given application. An observation has spatial, temporal, and thematic attributes, which may all contain lineage information. An example is that a specific GPS model is used to measure the spatial properties of an object. The observation itself may also contain lineage information, e.g. the name of the person who has registered the information. Finally, a geographic object, which consists of a number of observations, may also contain lineage information, e.g. that an object is derived from another object in another data collection.

6.2.2.2 Consistency

The consistency quality element includes the following:

- Domain consistency expresses whether information belongs to a given value domain, which may specify numeric types or enumerated types. An example is that an attribute value must have a value existing in the list: red, green, blue.

- Format consistency expresses to which degree attribute values satisfy a specified format, e.g. a temperature should be specified as a decimal number with two digits.

- Topologic consistency gives rules for how spatial attribute values should be related. An example is rules specifying that two polygons are not to overlap. Valid topologic relationships include *disjoint, contains, inside, equal, meet, covers, coveredBy,* and *overlap.* These eight relationships are valid for polygons in two-dimensional space and for lines in one-dimensional space [Kainz, 1995].

- Metric consistency refers to the relative positions of objects related to each other. An example is that an instance of one object type is to be at least 10 meters from an instance of another object type.

6.2.2.3 Completeness

Completeness concerns the absence or presence of some information. It includes descriptions of whether the information is in excess (i.e. objects are present in the information collection, but not in reality) or whether information is missing compared to the universe of discourse. These subelements are also termed commission and omission, respectively. An example of a completeness requirement is that the omission of buildings in a data collection must be less than 2%. To be able to assess completeness, some kind of reference information is required.

6.2.2.4 Accuracy

Accuracy covers a range of quality subelements and can be characterized according to the three dimensions of geographic information (theme, space, time).

- Thematic accuracy concerns thematic attributes and classifications of objects. The measurement of thematic accuracy can be to determine the classification correctness of object types against some kind of reference believed to be the truth. Typically, this is reported in an error matrix as seen in Table6.1.

Table 6.1: An error matrix.

	ClassA	ClassB
ClassA	95%	5%
ClassB	2%	98%

- Spatial accuracy concerns spatial properties of objects. Spatial accuracy can be either absolute or relative. The absolute spatial accuracy is the closeness of a stored position compared to some kind of reference believed to be true, whereas the relative spatial accuracy is the relative mutual closeness of objects compared to a reference.
- Temporal accuracy concerns the quality of temporal attributes. An example is the accuracy of the construction year for a building.

6.2.2.5 Precision

Precision is the level of known uncertainty of each registration and it is not to be confused with accuracy. As with accuracy, precision can be separated in three subelements that are related to the three dimensions of geographic information. Thematic precision refers to the precision of a unit used for registration. Spatial precision refers to the uncertainty of a measured position. Temporal precision refers to uncertainty of a time measurement. An example is that the expected spatial precision in a stereoscopic model is 0.5 meters in aerial photos on a scale of 1:25,000. Precision can be recorded when an object is instantiated.

6.2.3 Quality Assessment

Quality assessment is the process which generates the quality information and is always related to specifications. In addition to the standard application requirements, the specifications specify quality information that should be accessible, rules for the information that must be kept true, and the level of quality to which the produced information must conform. This implies that the quality is measured against the specifications. There are two overall approaches to assess quality: internal and external quality assessment. In the internal approach quality is assessed immediately after the information has been inserted into the database by using internal tests and, hence, requires no external source to be carried out. For example, topologic constraints can be evaluated immediately after the information is inserted into the database. If the information does not satisfy the constraints it are rejected. In the external approach information is

checked against an external reference believed to be the "truth". For example, if 90% of all forest objects are required to be present in the information, it has to be measured against some kind of reference. Hence, quality elements assessed by external methods need to be assessed at a later stage by means of an application on top of the given database. In Figure 6.3, a production process is depicted. It is seen that geographic information is captured from a given source

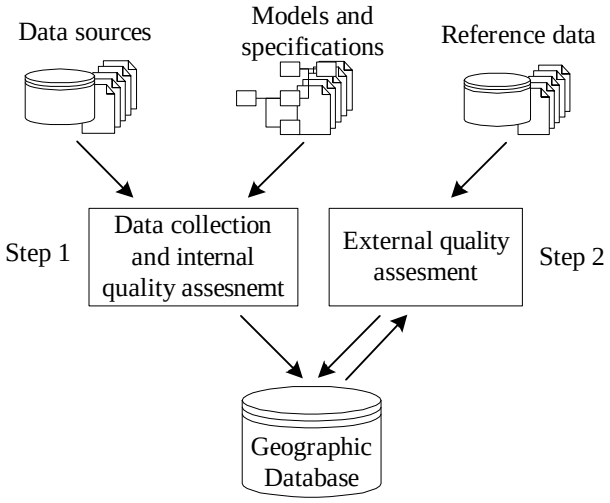

Figure 6.3: Quality assessment.

(e.g. aerial photos) and an internal quality assessment is performed based on the quality element requirements specified in the specifications. When the geographic data collection has been established, an external quality assessment can be performed. This requires a reference against which information can be measured. The result is a geographic data collection including quality information, which can be accessed by application users.

6.3 Modeling Quality

We proceed to present a framework for conceptual modeling of geographic information quality. We first describe how conceptual models can be extended to support quality requirements. Then, the various quality subelements are related to model constructs. This sets the stage for the quality-enabled model, which is covered last.

6.3.1 Quality in Conceptual Models

Conceptual models typically support constructs such as classes, associations, and attributes. No special notation is offered for the modeling of information quality, and database designs often do not capture information quality, because producers often neglect the importance of useful quality information. The fact that database designs do not offer appropriate quality information hinders the application users from considering quality for their applications.

In certain cases this can reduce the applicability of the data collections: to be able to use the information properly, it is important for the user to have access to quality information. Integration of information quality subelements into the common model constructs is a first step to make it possible to express quality requirements in the conceptual design phase. This would enable users of the information to have access to quality information.

Table 6.2 relates conceptual model constructs to data quality elements and is used to extend GeoSML with data quality elements. As seen from the table, quality subelements related to model constructs are divided into an aggregation level and an instance level. The aggregation level is the aggregated quality subelement measure for all objects of a class. The instance level is the quality subelement measure for each object. As described previously, quality requirements can reside at both levels.

Furthermore, the quality subelements are classified according to their required assessment approach. Completeness and accuracy sub-elements require external data to be assessed, whereas the remaining quality subelements only require internal quality assessment. Precision and lineage information are closely related to the production process and are attached to the instance of the model construct. Precision can later be assessed at the aggregation level. The consistency requirements are related to the specification of the universe of discourse. They specify requirements to be satisfied and are usually expressed as constraints in the conceptual models. Consistency requirements can be specified for instances, but also for collections and sets (aggregation level).

At the instance level, the omission subelement of completeness is not included for objects. This is because information cannot be associated with objects that do not exist in the data collection. On the other hand, commission error can be stored, because this means that an object exists in the data set, but not in the universe of discourse. For attributes and associations, we can assess both omission and commission. When a class is instantiated, it is possible to determine whether an attribute has a value or not. The same applies to explicit associations, as they can be implemented as foreign keys or object reference

Table 6.2: Quality related to model constructs.

Aggregation level

	Quality elements				
	Completeness (external)	Accuracy (external)	Precision (Internal)	Lineage (internal)	Consistency (Internal)
Class	Omission Commission	Thematic	Thematic	—	Format
Attr.	Omission Commission	*Absolute:* Spatial Thematic Temporal	*Absolute:* Spatial Thematic Temporal	—	Domain Format Topologic Metric
Assoc.	Omission Commission	*Relative:* Spatial	*Relative:* Spatial	—	Domain Format Topologic Metric

Instance level

	Quality elements				
	Completeness (external)	Accuracy (external)	Precision (internal)	Lineage (internal)	Consistency (Internal)
Object	Commission	Thematic	Thematic	Source	Format
Attr. value	Omission Commission	*Absolute:* Spatial Thematic Temporal	*Absolute:* Spatial Thematic Temporal	Source	Domain Format Topologic Metric
Assoc. instance	Omission Commission	*Relative:* Spatial	*Relative:* Spatial	Source	Domain Format Topologic Metric

attributes. Only spatial accuracy and precision of associations are possible. This is the relative distance, which may be relevant in certain situations. Based on requirements, the relative temporal and thematic accuracy/precision is not found to be relevant.

6.3.2 Quality Measures

Quality cannot be expressed by a single value as it depends on the application and on a range of subelements all expressing a measure for a single aspect of quality. The value of each subelement can be expressed differently. Here, some practices are presented that can be used in the determination of how to express the values of the subelements. The value domain of each subelement has to be determined before any implementation can take place.

Omission and commission. At the aggregation level omission and commission can be measured as a total number of objects, attributes, or associations either missing or in excess in the data set. It may also be presented as a percentage. For example 10% of the building objects are missing from the data set. At the instance level omission (only on attributes and associations) and commission can be represented as a Boolean value.

Accuracy is a complex quality element and can be measured and assessed in numerous ways [Drummond, 1995, Goodchild, 1995]. Spatial accuracy is the discrepancy (or error) measured between the "true" coordinates and the measured coordinates. A common way to measure accuracy at the aggregation level is to express the root mean squared error (RMSE). It requires a set of checkpoints to be assessed. For a point at the instance level, the error is recorded for each coordinate, either by recording the "true" value or by recording the discrepancy. Temporal accuracy can also be expressed by the RMSE. Thematic accuracy can be of numeric or nominal value. At the aggregation level, standard deviation can be used for the numeric values and percentage classified correctly (PCC) can be used for nominal values, usually presented via a misclassification matrix. At the instance level for nominal values, either a Boolean value or the "true" value can be recorded. For numeric values the "true" value or the discrepancy can be recorded. A reason for storing the "true" value can be that errors found are not necessarily corrected in the same process as the assessment.

Precision is related to the measurement techniques used and can be expressed by standard deviation (e.g. ± 2 meters). Both the instrument used and the operator influence precision. Hence, if different instruments or operators are used for registering a complete data set, the value can vary from the instance level to the aggregation level.

Consistency is specified by constraints and can be expressed, e.g. in natural language or as formal constraints (see Chapter 8). A consistency requirement at the aggregation level specifies a requirement valid for all instances

of a model construct, e.g. the sum of all values of a specific attribute must not exceed a certain number.

Lineage information describes the origin of the produced information, and it can include many types of information. Therefore, lineage must be described by a set of parameters defined by the users.

Naturally, the acceptable quality level has to be specified in the same value domain as the subelements. Which value domain to choose depends on the application and is to be decided by the designers.

6.3.3 Extending GeoSML with Quality Requirements

This section extends the grammar for conceptual models introduced in Section 4.4 to include quality requirements. Object type and relationship are redefined to include quality subelements used to describe relevant *quality element requirements* and need to be assessed at the aggregation and instance levels. The *acceptable quality level* is included in the conceptual model to specify requirements for the values of quality subelements for both the instance level and the aggregation level. This is done by extending the GeoSML language element for conceptual modeling with "AQL aggregation level", "AQL instance level", QER aggregation level", and "QER instance level" for ⟨object type⟩, ⟨attspec⟩, and ⟨association⟩.

⟨conceptual model⟩::= **Conceptual Model**⟨name⟩
(⟨object type⟩)*
(⟨conceptual relationship⟩)*
(⟨constraint⟩)*
(⟨conceptual term⟩)*

⟨object type⟩::=**Object Type**⟨object typeID⟩[**ET**][**TT**]
Name:⟨conceptual termID⟩
AQL Aggregation Level:(⟨objtypeAQL⟩⟨operator⟩⟨value⟩)*
AQL Instance Level:(⟨objtypeAQL⟩⟨operator⟩⟨value⟩)*
QER Aggregation Level(⟨AQualityElement⟩)*
QER Instance Level(⟨IQualityElement⟩)*
Attributes
(**Name:** ⟨attributID⟩⟨conceptual termID⟩ [**VT**][**TT**]
Data type:⟨datatype⟩
AQL Aggregation Level:(⟨attAQL⟩⟨operator⟩⟨value⟩)*
AQL Instance Level:(⟨attAQL⟩⟨operator⟩⟨value⟩)*
QER Aggregation Level(⟨QualityElement⟩)*

QER Instance Level(⟨QualityElement⟩)*)*

⟨conceptual relationship⟩::=⟨crID⟩⟨crdef⟩

⟨crdef⟩::=⟨cr-association⟩ |⟨cr-is-a⟩ |⟨cr-part-whole⟩ |⟨cr-spatialrelation⟩

⟨cr-association⟩::=**Association:**⟨typeID⟩⟨conceptual termID⟩⟨conceptual termID⟩⟨typeID⟩⟨relqualiyelement⟩

⟨cr-part-whole⟩::=**Part-whole relation:**(⟨typeID⟩)* **is part-of** ⟨typeID⟩⟨relqualiyelement⟩

⟨cr-is-a⟩::=**Taxonomic relation:**(⟨typeID⟩)* **is-a** ⟨objecttypeID⟩

⟨cr-spatialrelation⟩::=**Spatial relation:**⟨named spatial relation⟩ |⟨defined spatial relation⟩

⟨named spatial relation⟩::=⟨typeID⟩⟨topologicoperator⟩⟨typeID⟩⟨relqualiyelement⟩

⟨defined spatial relationship⟩::=⟨typeID⟩⟨topologicdef⟩⟨typeID⟩ ⟨relqualiyelement⟩

⟨relqualiyelement⟩::=
AQL Aggregation Level:(⟨attAQL⟩⟨operator⟩⟨value⟩)*
AQL Instance Level:(⟨attAQL⟩⟨operator⟩⟨value⟩)*
QER Aggregation Level:(⟨QualityElement⟩)*
QER Instance Level:(⟨QualityElement⟩)*

⟨objectypeAQL⟩::=**Omission** |**Commission** |**Accuracy** |**Precision**

⟨attAQL⟩::=**Omission** |**Commission** |**Accuracy** |**Precision**

⟨relAQL⟩::=**Omission** |**Commission**

⟨qualityElement⟩::=⟨Lineage⟩ |**Omission** |**Commission** |**Accuracy** |**Precision**

⟨lineage⟩::=**linage:**⟨user-defined quality element⟩⟨statement⟩

⟨user-defined quality element⟩::= **Producer** |**User** |**Software version** |**Production date** | ...

The grammar shown below extends the GeoSML to support geographic data quality using Table 6.2 as a basis.

The syntax enables designers to specify both requirements for the acceptable quality level (AQL) and requirements for the elementsused to measure the information quality (QER). Some of the elements, such as `accuracy`, `precision`, `omission`, and `commission` are associated with a GeoSML element that carries quality information, so that their values belong to the object type and, hence, apply to all instances. They enable specification of quality requirements at the aggregation level.

6.4 An Example

In this section our example from the previous chapters is continued by adding requirements related to quality to the conceptual model by use of the suggested syntax.

The example includes a data collection with the four basic classes, `Building`, `Road`, `Road_Segment`, and `Municipality`. The model contains no explicit information on quality requirements, though some are represented implicitly in the model. An example is that the cardinalities of associations express consistency requirements. Apart from the implicit quality requirements, several additional quality requirements exist. We proceed to state the quality requirements that must be included in the model by describing each requirement in natural language.

1. The address is a characteristic of a building; thus, it should be located inside the building it belongs to (consistency requirement).
2. All road segments must have an average maximum root mean squared error of maximum 1.7 meters (accuracy requirement).
3. The name of a road object should be correct (accuracy requirement).
4. A road object should have a name (completeness requirement).
5. At least 99% of all roads in the domain must be represented in the data, and no more than 1% of all roads must be represented in the data without being in the domain (completeness requirement).
6. At least 99% of all road segments must be classified correctly (accuracy requirement).
7. For all roads, buildings, and municipalities, there should be information about who has digitized and created each object (lineage requirement).
8. The height of buildings must be measured by an accuracy of 1 meter (accuracy requirement).

This list exemplifies the variety of quality requirements that exist. We would like to be able to express these requirements in our conceptual model. Adding these quality requirements to the existing conceptual model from Chapter 4.4results in the following specification (only elements necessary for incorporating the quality requirements are included):

```
Conceptual model My Small Topographic Map
    Object Type building
        name: building
        AQL Aggregation Level:
            Commission < 0,01
            Omission < 0,01
        QER Instance level
            Operator
            Producer
        Attributes:
            building_outline polygon
            construction_year integer
            building_type string
            number_of_floors integer
            hight float
            AQL Aggregation Level:
                Accuracy < 1m
    Object Type road_segment
        Name: road_segment
        AQL Aggregation level:
            commission < 0.01
            omission < 0.01
        Attributes:
            centerline polyline
            AQL Aggregation level
                accuracy < 1.7m
            road_type string
            AQL Aggregation:
                thematic accuracy = 1
            road_name string
            AQL instance level
                accuracy = 1
    Object Type address
        Name: address
        Attributes:
            house_number housenumber
    Association: address gives access to building
        AQL Aggregation level
            Completeness = 1
```

6.5 Summary

This chapter discusses the notion of quality in the context of specifying geographic information. Two important concepts has been introduce (i) Acceptable quality level (AQL) which are used to describe requirements to the intervals or values measured quality parameters must be within and (ii) Quality parameter requirement (QER) which are used to list the requirements with which the quality of a data collection must be measured. Furthermore, the GeoSML framework has been extended by language elements for including the two at the conceptual modeling level.

Chapter 7

Constraints on Geographic Information

Abstract: The aim of this chapter is to identify the requirements for a declarative langauge built on first order predicate logic, which can specify formal constraints on geographic data collections. Five types of constraints are identified and it is discussed how each type can be formalized by first order predicate logic. The requirements and analyzes made in this chapter are used in the following chapter to develop a constraint language. This constraint language extends GeoSML to include formal constraints, which can be evaluated to check whether the produced information conforms to the specified requirements.

7.1 Introduction

Ensuring the quality of produced information is a significant task in the production of geographic information. Generally speaking, the quality is checked either by sampling where a number of objects are chosen for inspection and compared with a superior data set, or the quality is checked by ensuring that the produced information conforms to a set of predefined constraints.

We shall in the following two chapters focus on constraints for geographic information and the formalization of these. GeoSML, as it is defined in the previous chapters, regards constraints as statements formulated in natural language. To evaluate the conformance to predefined constraints, the produced information must be evaluated in a software system capable of constraint checking. It is a requisite to this process that the constraint must be suitable for evaluation in a computer-based system. In general, this means that the constraints must be specified by a formal language and that they must be formulated in the context in which the information is computationally handled and stored.

The objective of this chapter is to establish a basis for the formal constraint language, which is introduced in the next chapter. The aim is to identify classes of constraints that can be used to ensure the consistency and to create a platform for developing a declarative rule language with resemblance to natural language and which can be automatically translated into queries in ordinary query language, such as SQL.

Constraints for geographic information have not been given much attention in research. Cockcroft discusses the classification of constraints and the relation between business rules and constraints on geographic information [Cockcroft, 1997]. Frank discusses constraints in the context of ontologies and specifications of geographic information [Frank, 1998]. In [Hoel et al., 2003] a framework for defining and storing topology constraints is presented. One of the few examples treating formal approaches to constraints on geographic information is [Brisaboa et al.,], here the Unified Modeling Language is used in combination with the Object Constraint Language to test the ability of the two languages to specify formal constraints on geographic information.

The remainder of this chapter is organized as follows. Section 7.2 explains the basic idea of formalization and evaluation of constraints and Section 7.3 introduces five types of constraints that are relevant to geographic information and the formalization of these.

7.2 The Basic Idea

Geographic information is produced by use of geographic information systems. These systems are specially designed to create, store and handle spatial information. The requirements for these systems depend on the type of information the systems must produce. In most cases the system, to some extent, must be able to validate if the produced information conforms to the set of rules stated in the specification. Today these rules and constraints are mostly hard-coded into the software itself, using proprietary customizing languages delivered with most commercial systems, e.g. MapBacis for MapInfo or ArcObject for the ArcGIS family. Some systems accommodate functions for specifying topology rules in the user interface, e.g. ArcInfo.

A disadvantage of hard coding or defining constraints in proprietary systems is that it becomes difficult to use more than one production system or to change from one to another. It is also a problem that the implemented rules and constraints exist independently of the specifications, making it difficult to maintain when specifications change.

Figure 7.1 suggests an architecture separating the validation of produced information from the production software by delegating the control of the validation process to a *validation engine*. This results in a loose coupling with other parts of the production environment. The construction has several advantages compared to traditional approaches, since it

1. makes it easier to maintain the constraints
2. enables the reuse of constraints in several production systems
3. makes it easier to replace parts of the production systems

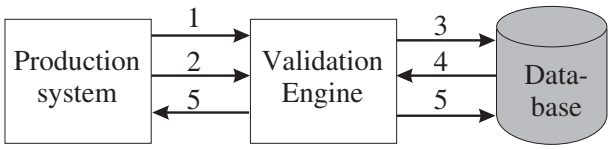

Figure 7.1: Ensuring consistency using a Validation Engine.

The following workflow can be applied to validation of the produced information

1. Initialization of validation engine

2. Sending information to the validation engine
3. The validation engine starts a transaction on the database, inserting the information in the database and "firing" the appropriate constraints at the database
4. The database returns the result of the queries
5. The validation engine analyzes the results and responds to the production system and the database
    - If the information passes the tests, then the changes are committed to the database and the production system is given a "passed test" message.
    - Otherwise a rollback is issued to the database and the production system receives a "failed test" plus a list of the failed constraints

The realization of this idea depends on a medium or language to communicate the rules and constraints to the validation engine. This kind of languages is addressed as formal constraint languages.

Wagner identifies three types of constraints [Wagner, 2002]:

- Integrity constraints
- Derivation rules
- Reaction rules

An **integrity constraint** is an assertion that must be satisfied in all states and state transitions of a system, e.g. a database. Two types of constraints exist: *state constraints* and *process constraints*.

State constraints must hold at any point in time. An example of a state constraint is: *The buildings in a topographic database must not overlap.*

A **derivation rule** is a statement of knowledge derived from other knowledge by an inference or a mathematical calculation. Derivation rules allow to capture terminological domain knowledge about concepts whose extension is not stored explicitly. An example of a derivation rule is the following definition of a connected pipe in water supply system: *A connected pipe is a pipe that is connected to two other pipes, one in each end.*

The third kind of rule is **reaction rules** or event-condition-action rules, as they are also called. This type of rules is concerned with regulating the behavior of a system. They describe patterns for which actions should be started in response to events, by stating the conditions under which the actions must be taken. An example of a reaction rule from the domain of topographic mapping is: *When a new building is inserted in the database, then it must be checked if a new residential area also must be created, or if an existing must be expanded.*

In the remainder of this chapter the focus will be on what Wagner calls integrity constraints. In the following, production or updates of a data collection will be regarded as transactions bringing the database from one valid state to another valid state. What determines if a state is valid or not is if a predefined and finite set of constraints "fired" toward the database returns an empty set of records or not. The constraints formulate the well-formed conditions for the objects in the database.

In order to elicit the requirements for a constraint language suited for validating geographic information, the various types of constraints are discussed in the next section.

7.3 Constraints on Geographic Information

To be precise about the conceptualization of a domain, constraints are included in the conceptual model. Constraints are statements assigned to guard the intended meaning of a requirement, an assertion, or a rule describing the interpretation of domain entities. If for example a selection rule states that buildings in the domain must be larger than 10 m^2 to be represented in the data collection, then the corresponding constraint could be that the area of building outlines must be larger than 10 m^2.

The difference between the two statements is that the first statement binds terms from the domain model to terms in the conceptual model, while the latter statement only includes terms from the conceptual model. This is a characteristic property of constraints – they must only include terms, or more precisely references to object types and relations, in the conceptual model. Constraints are divided into five types:

- Topology constraints
- Domain constraints
- Format constraints

- Temporal constraints
- Metric constraints

The following sections describe the five types of constraints, discuss possible formalizations and evaluations of the constraints, and give examples using predicate logic of the various types.

7.3.1 Topology Constraints

A topology constraint specifies consistency requirements for topologic relationships among objects, e.g. *buildings must not intersect residential areas*; *routes must be on top of a road network*; and *buildings must not be inside a lake*. They are all examples of topology constraints.

Topology constraints are important for geographic information. By formulating and applying these in the production process three goals are supported: (i) "nice" cartographic representation of the geographic information is achieved and e.g. overlapping polygons avoided. (ii) the relations among objects do not violate the users perception of how entities relates in reality. (iii) topological structures in e.g. road networks are guarantied.

Topology constraints uses topological relations among geometries. The theory of topological relations for geographic information originates from point-set theory. Egenhofer [Egenhofer and Herring, 1990] introduces a 9-intersection model to compare the spatial relation between two geometries (A and B) in the two-dimensional space, by studying the nine intersections between As interior (A^o), boundary (*partial*A), and exterior (A^-) with Bs interior (B^o), boundary (∂B), and exterior (B^-).

$$\Gamma_9(A,B) = \begin{pmatrix} A^o \cap B^o & A^o \cap \partial B & A^o \cap B^- \\ \partial A \cap B^o & \partial A \cap \partial B & \partial A \cap B^- \\ A^- \cap B^o & A^- \cap \partial B & A^- \cap B^- \end{pmatrix}.$$

The values in the matrix are determined by two rules: If the intersection is the empty set ($\emptyset$) the value is set to zero, and if the intersection is the non-empty set ($\neg\emptyset$) the value is set to one. Thus, the topologic relation between

two geometries can be described by a three-by-three matrix with either the value "1" representing the non-empty set, or the value "0" for the empty set. Figure 7.2 illustrates threetopologic relations which can be modeled by the nine-intersection matrix. The corresponding matrices are given in table 7.3.1.

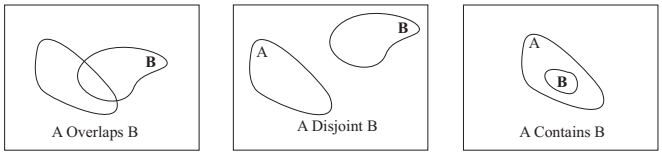

Figure 7.2: Examples of topologic relations which can be modeled by the nine-intersection matrix.

$$\Gamma(A,B) = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} \quad \Gamma(A,B) = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix} \quad \Gamma(A,B) = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix}$$

Three topologic relations described by the nine-intersection matrix.

To describe topologic relationships more precisely, the dimensionally extended nine-intersection Matrix is introduced by applying the function *dim(x)* to each intersection geometry in the tree-by-three intersection matrix [Clementini et al., 1993]. This function returns the highest dimension of the intersection geometry, i.e. it returns the value 0 if highest dimension of all geometries in the resulting intersection geometries is zero-dimensional, 1 if it is a one-dimensional geometry, and 2 if x is a two-dimensional geometry. -1 if the intersection geometry is the empty set:

$$\Gamma_9(A,B) = \begin{pmatrix} dim(A^o \cap B^o) & dim(A^o \cap \partial B) & dim(A^o \cap B^-) \\ dim(\partial A \cap B^o) & dim(\partial A \cap \partial B) & dim(\partial A \cap B^-) \\ dim(A^- \cap B^o) & dim(A^- \cap \partial B) & dim(A^- \cap B^-) \end{pmatrix}.$$

Investigations of the values of the dim(x) function for all combinations of intersections between interiors, exteriors, and boundaries of two geometries, give a detailed description of the topologic relations between the two geometries.

Intersection matrices can describe topologic relations among all combinations of points, lines, and polygons if the interior, boundary, and exterior are defined. For points the interior and boundary are defined as the point itself and the exterior as all other points. For lines the interior is defined as the line excluding the end points, the boundary as the end points, and the exterior as all other points.

Intersection matrices can also be used to define allowed topologic relations among geometric objects, e.g. two building objects. For this purpose the predicate, R (or relate) is introduced [ESRI, 2003]. This predicate takes two geometries and the allowed values for the nine values of the intersection matrix and returns TRUE, if the values for the intersection matrix correspond to the acceptable values, otherwise it returns FALSE.

The allowed values for each cell in the intersection matrix can be any subset of {-1,0,1,2}. To ease the description of allowed values, the symbol p is introduced. This symbol is assigned the following values, depending on the allowed values of dim(x):

$$p = \begin{cases} T & \text{if } \dim(a) \in \{0,1,2\} \\ F & \text{if } \dim(a) = \text{-1} \\ * & \text{if } \dim(a) \in \{-1,0,1,2\} \\ 0 & \text{if } \dim(a) = 0 \\ 1 & \text{if } \dim(a) = 1 \\ 2 & \text{if } \dim(a) = 2 \end{cases}$$

The following definitions exemplify the use of the dimensional extended nine-intersection matrix:

Disjoint: disjoint(a,b) $\leftrightarrow$ relate(a,b,"FF*FF****")

Touches: touches(a,b) $\leftrightarrow$ relate(a,b,"FT*******") $\vee$ relate(a,b,"F***T****") $\vee$ relate(a,b,"F**T*****")

Line crosses polygon: crosses(a,b) $\leftrightarrow$ relate(a,b,"T*T******")

Line crosses line: crosses(a,b) $\leftrightarrow$ relate(a,b,"0********")

Within: within(a,b) $\leftrightarrow$ relate(a,b,"T*T**F***")

Two points or polygons overlap: overlaps(a,b) $\leftrightarrow$ relate(a,b,"T*T***T**")

Two lines overlap: overlaps(a,b) $\leftrightarrow$ relate(a,b,"1*T***T**")

Intersects: intersect(a,b) $\leftrightarrow$ disjoint(a,b)

Contains: contains(a,b) $\leftrightarrow$ within(b,a)

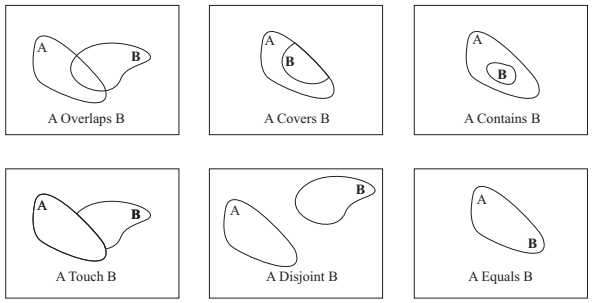

Figure 7.3: Some topological relations between to polygons

Examples of formulas including topology predicates is the following:

$$\forall b(building(b) \rightarrow (\neg \exists ra : residentialarea(r) \wedge overlap(b, r)))$$

This formula can be interpreted as: for all buildings (b) there must not exists a residential area (r) so that the building and residential area overlap.

$$\forall b_1, b_2(building(b_1) \wedge building(b_2) \rightarrow disjoint(b_1, b_2))$$

This constraint states that all pairs of buildings must be disjoint, i.e. no buildings must be overlap.

$$\forall ra, b(residential - area(ra) \wedge building(b) \rightarrow \neg overlap(ra, b))$$

The above constraint states that all pairs of buildings and residential areas must not be overlap.

Topology constraints are not only restricted to descriptions of intersections among geometries.

Topology constraints can also express connectivity rules in networks, e.g. *"Butterfly valves can only be connected to a pipe with a diameter larger than 14*

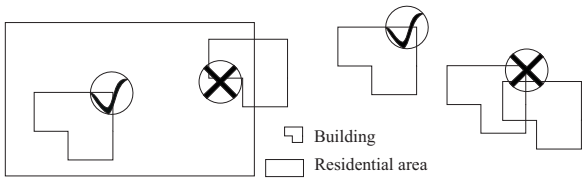

Figure 7.4: Allowed and disallowed relations between buildings and residential areas.

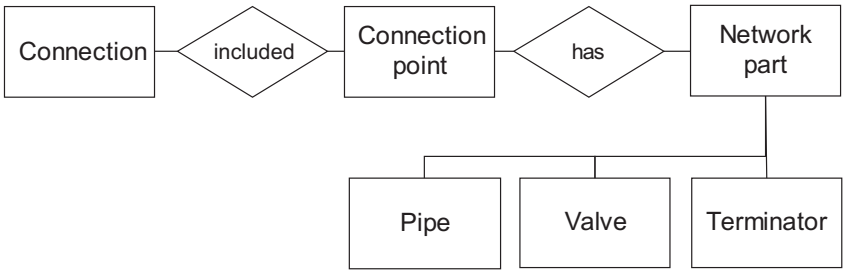

Figure 7.5: Entity-relationship for a network model.

inches". Figure 7.5 shows a small conceptual model for a database representing a water supply system.

A topology constraint including subtyping of pipes and valves, requiring that pipes must be of a certain type to be connected to a certain type of valves, can be stated as follows:

$$\forall v, p(valve(v) \wedge pipe(p) \wedge isconnected(v,p) \wedge sizelargerthan(p,'14') \rightarrow havetype(v,'butterfly'))$$

Another example of a network constraint is that road segments included in a road network are not allowed to have pseudonodes, which means that at least three end points must meet in an intersection.

7.3.2 Domain Constraints

The term domain in domain constraint must not be confused with the term *domain* in the domain model. For historical reasons we use the term domain constraints for the kind of constraints that restrict the values of attributes, which is widely used in the database literature, e.g. [Elmasri and Navathe, 1996] where a database relation is defined as a subset of the Cartesian product of the possible domain values of each attribute:

$$r(R) \subseteq (dom(A_1) \times dom(A_2) \times \ldots \times dom(A_n))$$

Also in the area of geographic information systems, there is a tradition for the use of domain constraints, e.g. the Modular GIS Environment for Intergraph uses the notion of list domain and range domain for attributes and the ArcGIS software family also supports predefined domains for attributes.

7.3.2.1 Coded Value Domains

A coded value domain is a kind of domain constraint that defines a finite list holding allowed values for an attribute. An example is that the `roadtype` attribute of a `road` object must be either `minorroad`, `majorroad`, or `highway`. By use of predicate logic this can be be formalized by the following formulas:

$$\forall r(road(r) \rightarrow hastype(r,'minorroad') \vee hastype(r,'majorroad') \vee hastype(r,'highway'))$$

$$\forall r(road(r) \wedge hastype(r,'minorroad') \rightarrow \neg hastype(r,'majorroad') \wedge \neg hastype(r,'highway'))$$

$$\forall r(road(r) \wedge hastype(r,'highway') \rightarrow \neg hastype(r,'minorroad') \wedge \neg hastype(r,'major'))$$

$$\forall r(road(r) \wedge hastype(r,'majorroad') \rightarrow \neg hastype(r,'minorroad') \wedge \neg hastype(r,'highway'))$$

There is no doubt about the insufficiency of this approach to specifying coded list domains, and some languages have introduced an abbreviated syntax for this construct. E.g. the *oneof* provided by some description logics [Smith et al., 2004]. The above example can be expressed in owl as

```
<owl:oneOf rdf:parseType="Collection">
  <owl:Thing rdf:about="#minorroad"/>
  <owl:Thing rdf:about="#majorroad"/>
  <owl:Thing rdf:about="#highway"/>
</oneOf>
```

7.3.2.2 Range Domain

Range domains define allowed intervals for values of attributes. Examples are temperatures in kelvin, which must always be larger than zero, or the width of a road those minimum value could be defined to be 2 meters and maximum value could be set to 100 meters.

$$\forall\ r(road(r) \rightarrow widthlargerthan(r,'2') \wedge widthsmallerthan(r,'25'))$$

7.3.2.3 Functional Dependencies among Domain Constraints

A value of an attribute may reduce the valid domain for other attributes. E.g. if it is given to which a building is located within a certain municipality, then the possible zip codes that the building can belong are restricted to the ones intersecting the municipality.

$$\forall b(building(b) \wedge isinmunicipality(b,'Smalltown') \rightarrow isinzipcode(b,'1111') \vee isinzipcode(b,'2222'))$$

This type of relationship is called non-deterministic functional dependencies. A constraint on butterfly valves and allowed sizes of connected pipes is another example of this kind of domain constraint.

7.3.3 Format Constraints

A format constraint specifies consistency requirements that constrain a value to some kind of format. An example is the use of road codes in Denmark. These codes are used to unique identification of roads and their names in a municipality. The specification states that road codes must consist of precisely four characters, and each of the four characters must be one of the digits from 0 to 9. Thus, an allowed code is "0034", whereas "34" is a disallowed code.

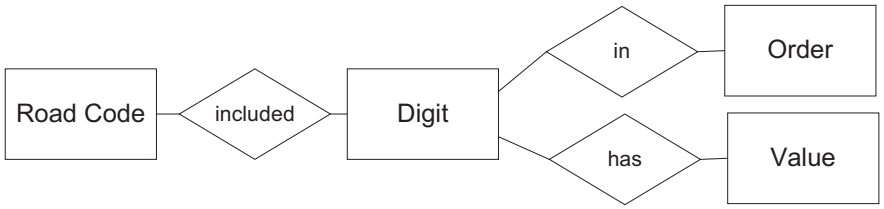

Figure 7.6: Entity-relationship diagram of details of the road code format.

Figure 7.6 is an example of a conceptual model capturing the requirements for the format of road codes.

- Road codes must have four digits
- Order, can have the values 0, 1, 2, or 3
- One digit included in a road code must be of the order 0
- One digit included in a road code must be of the order 1
- One digit included in a road code must be of the order 2
- One digit included in a road code must be of the order 3
- Value can have the value 0, 1, 2, 3, 4, 5, 6, 7, 8, or 9

Formulating formal constraints on the basis of conceptual models is properly not the best seen in relation to specifying format constraints: It would require that the conceptual model should be included in the conceptual model.

Another approach more suitable for specifying format constraints is to see them as grammars restricting the value of an attribute. If the road code example is used again the grammar could be:

⟨road_code⟩::=⟨symbol⟩⟨symbol⟩⟨symbol⟩⟨symbol⟩
⟨symbol⟩::=**0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9**

In the suggested approach the subsystem (the database or some software working together with the database) will be responsible for the evaluation of the format constraints, and the specified constraints will use what is called a user-defined predicate to ensure format consistency, e.g.:

$$\forall b(building(b) \rightarrow hascorrectformat(b))$$

This approach ensures a loose coupling from the production software and makes it possible to formulate the formal format constraints in a declarative style. At the specification level the formal constraints must be formulated by natural language.

7.3.4 Metric Constraints

A metric constraint specifies the consistency requirement for measures of length and size. It could be a required minimum distance between buildings or a minimum size of lakes. Metric constraints can be formulated for one object, e.g. the minimum area of buildings, or for two objects, e.g. the minimum distance between two forests.

Metric constraints require some sort of calculation to be evaluated, because the required information is not necessarily is stored in the database. It is possible to make this sort of calculation using predicate logic by introducing simple arithmetic operators and building more complex expressions from them. This approach results in complex formulas and may be difficult to implement in production systems. A more sufficient approach is to leave the evaluation of the formulas to the subsystems, e.g. databases or code written in traditional imperative programming languages.

The execution of these calculations can be activated by including predicates in the constraints, which are evaluated by subsystems. An example is a constraint the stating that all lakes must be of an area larger than 100 m^2, which can be expressed by the following formula:

$$\forall l_1, l_2(lake(l_1) \wedge lake(l_2) \rightarrow distancemustbelargerthan(l_1, l_2, 2))$$

The `distancemustbelargerthan(object, object, threshold)` must be implemented by the system and evaluated, when a lake is inserted in the database.

Some metric constraints may be computationally challenging. An example is a constraint stating that the distance from a point on the boundary of a residential area to a building located inside the area must not exceed 50 meters. By predicate logic this can be expressed as

$$\forall ra(residentialarea(ra) \wedge \forall p(point(p) \wedge touch(p, ra) \rightarrow \exists b(building(b) \wedge inside(b, ra) \wedge distancelessthan(p, b, 50))))$$

Seen from an implementation point of view this formula suggests that it should be checked if all points on the boundary of residential areas are less than 50 meters away from at least one building inside the residential area.

If this problem is regarded as a continuous function defining the distance from a point on the boundary to the nearest building, it can be realized that even though the number of points on the boundary is infinite it is possible to solve it computationally. The problem is, from is conceptual point of view, that the points on the boundary are not included in the conceptual model, which would require access to points at database level. A more operational approach would be to introduce a predicate to be evaluated at application level and then formulate the constraint in the following way:

$$\forall ra(residentialarea(ra) \rightarrow \exists s(set(s) \wedge \forall b(isin(b, s) \wedge (building(b) \wedge (inside(b, ra) \wedge hasdistancetoinnerpart(ra, s))))))$$

The `hasdistancetoinnerpart` must be evaluated by an algorithm at application level, e.g. by requiring the intersection of the 50 meter buffers of the buildings within an residential area to contain completely the residential area (see Figure 7.7).

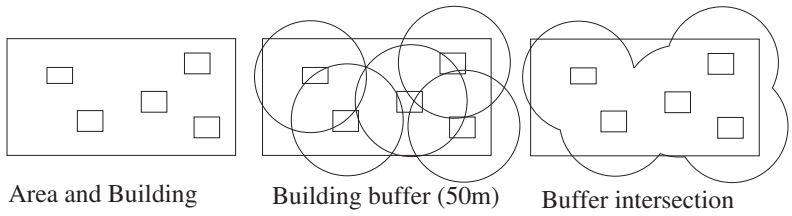

Figure 7.7: Strategy for evaluation of "inner distance".

The proposed approach suggests that constraints can be specified in a generic constraint language, based on predicate logic and translated into one or more tagged languages with two tasks: (i) implementing the user-defined predicates, and (ii) executing the constraints when appropriate.

7.3.5 Temporal Constraints

Temporal constraints are applied in order to restrict the values of temporal attributes and temporal relations among objects. As in the case of topologic relations, predicates determining the binary relations between two time intervals can be defined. Allen defines thirteen relations among time intervals, of which twelve are pairwise inverse [Allen, 1983] (see Figure 7.8).

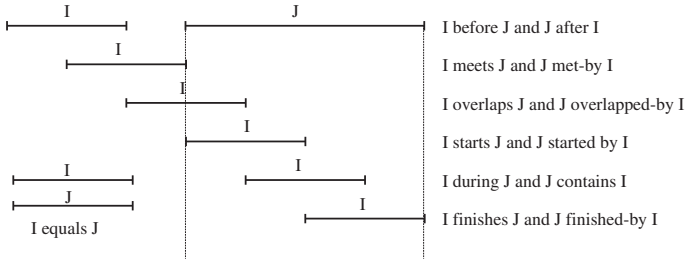

Figure 7.8: Relations between two time intervals (after [Ohlbach, 2004])

The relations can be used to formulate constraints between events (see e.g. [Campos and Hornsby, 2004]. Examples are: *"A bus must arrive before it can leave from the bus station"*, or *"The construction of walls must not begin before the foundations have been finished"*.

To capture the meaning of the second statement using predicate logic, a predicate, *interval(x)*, representing the fact that x is a time interval is introduced.

$$\forall b, ws, f(building(b) \wedge setofwalls(ws) \wedge foundation(f) \wedge part-of(ws, b) \wedge part-of(f, b) \rightarrow \exists i_1, i_1(interval(i_1) \wedge interval(i_2) \wedge constructedin(ws, i_2) \wedge constructedin(f, i_1) \wedge before(i_1, i_2)))$$

The life cycle of geographic entities can be described by a set of rules is called life cycle rules, which are used to create and maintain references to entities. The aim is to create a platform to define how much an entity can change before it turns into another entity. Or in other words, to decide whether a transition from one state to another is identity preserving.

Constraints can be applied to ensure that versioning of objects conforms to the formulated life cycle rules. The idea is that if an entity changes from one state to another and the identity is to be preserved, the database is updated inserting a record using the same identifier as the representation of the entity had before it was changed.

In a conceptual model the specification of details about versioning of objects in the database is to be avoided. On the other hand access to concepts that can be used to refer to preceding and proceeding versions of an object is needed. Therefore, a number of predicates which can be used to determine if two objects are representations of the same entity are introduced. If they are, it is said that they are two versions of an object representing the same entity.

preceding_version(A,B) is a predicate that holds true if A is the preceding version of B.

preceding_versions(A,B) is a predicate that holds true if A is one of the preceding versions of B.

proceeding_version(A,B) is a predicate that holds true if A is the proceeding version of B.

proceeding_versions(A,B) is a predicate that holds true if A is one of the proceeding versions of B.

neighbor_version(A,B) is a predicate that holds true if A is one of the proceeding versions of B or if B is one of the proceeding versions of A.

neighbor_versions(A,B) is a predicate that holds true if A is one of the proceeding versions of B or if B is one of the proceeding versions of A.

current_version(A) is a predicate that holds true if there is no B for which it holds that it is the preceding version of A.

Proceeding_vesions(A,B) can recursively be defined in the context of proceeding_version(A,B), as

$$\forall a, b(proceeding\_versions(A, B) \rightarrow preceding\_version(A, B) \vee (preceding\_version(C, B) \wedge proceeding\_versions(A, C)))$$

and the neighbor_version(A,B) can be defined as follows

$$\forall a, b(neighbor\_version(a, b) \rightarrow preceding\_version(a, b) \vee preceding\_version(b, a)$$

and the neighbor_versions(a,b) can be defined as

$$\forall a, b(neighbor\_versions(a, b) \rightarrow \\ preceding\_versions(a, b) \vee preceding\_versions(b, a)$$

Finally current_version(A) can be defined as the following:

$$\forall a(current\_version(a) \rightarrow \neg \exists b preceding\_versions(a, b))$$

These predicates can be evaluated by the database management system, either by creating views or as implemented functions. If for example the versioning of objects is managed by taking snapshots each time changes are made to an object and the valid time is recorded, then it can be evaluated if the state transactions conform to the stated life cycle rules.

An example of a life cycle rule that describes some of the properties influencing the identity of buildings is:*"A building is regarded as the same as long as its area is not extended or reduced considerably compared to the area the building had before it was changed."*. This rule can be rewritten as a constraint by the following rule:

$$\forall b_1, b_2(building(b_1 \wedge building(b_2)) \wedge overlaplargerthan(b_1, b_2) \rightarrow \\ neighbor\_version(b_1, b_2)))$$

The above expression can be read as: Ror all pairs of buildings which overlap more than 70% it must hold that they are neighbor versions.

Sequential changes can be applied to an entity, e.g. a building may be extended several times. If the above rule is changed to the following: *"A building is regarded the same as long as its area is not extended or reduced considerably compared to the area the building had when it was constructed"*, then the neighbor_version must be substituted by the neighbor_versions predicate:

$$\forall b_1, b_2(building(b_1 \wedge building(b_2)) \wedge overlaplargerthan(b_1, b_2) \rightarrow \\ neighbor\_versions(b_1, b_2)))$$

Another example of constraints that can be used to guard the intended meaning of life cycle rules concerns the identity of roads. To the right in Figure 7.9 the rerouting of a road is illustrated. The question is if the road is perceived as the

same as before the changes were made. A life cycle rule could state: *"When roads are rerouted they are regarded as the same as long as the connections to other roads do not change"*, and thus it can be decided that they do.

Expressing this life cycle rule as a constraint requires an additional condition to the one stating that road segments with the same start and end points must be versions of the same object, namely that the two objects must also intersect. The reason is that more than one road may start and end at the same intersection.

$$\forall rs_1, rs_2(roadsegment(r_1 \wedge roadsegment(rs_2)) \wedge have - same - start - point(rs_1, rs_2) \wedge have - same - end - point(rs_1, rs_2) \wedge intersects(rs_1, rs_2) \rightarrow neighbor\_versions(rs_1, rs_2))$$

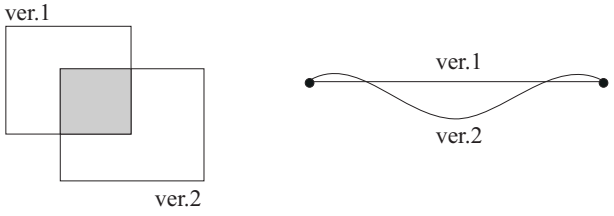

Figure 7.9: Left: two overlapping buildings, right: two road segments with identical start and end points.

Another life cycle rule concerning to change of the pavement or surface of a road does not affect the identity.

Thus, there must not be two geometrically identical road segments with different surface materials that are not versions of the same object.

$$\forall rs_1, rs_2(roadsegment(r_1 \wedge roadsegment(rs_2)) \wedge equal(rs_1, rs_2) \wedge differentsurfacematerial(rs_1, rs_2) \rightarrow neighbor\_versions(rs_1, rs_2))$$

If a road changes its name it is regarded as a new road:

$$\forall rs_1, rs_2(roadsegment(r_1 \wedge roadsegment(rs_2)) \wedge equal(rs_1, rs_2) \wedge differentname(rs_1, rs_2) \rightarrow \not neighbor\_versions(rs_1, rs_2))$$

7.3.6 Combining Different Types of Constraints

The categorization of constraints into five types is mostly relevant for presentation purposes. In reality, many constraints will include parts that are combinations of two or more of the five types. Often topologic and metric constraints are combined to form more complex expressions. An example is: *"Road segments that are not connected must be longer than 50 meters"*. Here the word *"connected"* indicates a topologic relationship, and the word *"longer"* indicates a metric constraint.

Constraints can also be included in compound statements using the "or"/"and" operators. In declarative language the "and" operator is implicitly given by line breaks, e.g. the statement *"All houses must have at least 1 floor and a height larger than 3m"* can be formalized by the following two statements:

$$\forall b building(b) \rightarrow (numberoffloors(b) \geq 1)$$

$$\forall b building(b) \rightarrow (height(b) > 3)$$

The *"or"* operator must be explicitly used: *"The length of a road segment must be smaller than 2m or longer than 100m"*

$$\forall r roadsegment(r) \rightarrow ((length(r) < 2) \vee (length(r) < 100))$$

7.3.7 Auxiliary Predicates

To ease the formulation and the reading of constraints, it is suggested that complex constraints are broken into smaller pieces by using auxiliary predicates in the definition of concepts which can be used to form complex constraints. Auxiliary predicates are especially useful when the same structure is used in several constraints. To illustrate the application of auxiliary predicates the above example of water supply systems can be concidered example. First, two new predicates are defined, after which they are applied to a constraint.

(i) A connected pipe is defined as a set of all pipes having all their connection points included in one connection:

$$connected - pipe(p) \equiv$$
$$\forall p(pipe(p) \wedge (\exists p_1, p_2 pipe(p_1) \wedge pipe(p_2) \wedge isconnected(p, p_1) \wedge isconnected(p, p_2)))$$

(ii) A terminated pipe is defined as a set of all pipes with only one connection point included in a connection where the other connected to a pipe in one end and to a terminator in the other end:

$$terminated - pipe(p) \equiv$$
$$\forall p(pipe(p) \wedge (\exists p_1, p_2 pipe(p_1) \wedge pipe(p_2) \wedge isconnected(p, p_1) \wedge isconnected(p, p_2)))$$

The definitions can be used to make the constraints much easier to read than if the full expressions has been included instead. E.g. the two statements can be included in a constraint which states that *"All pipes in service must either be a terminated pipe or a connected pipe"*:

$$\forall p(pipe(p) \wedge inservice(p) \rightarrow (connected - pipe(p) \vee terminated - pipe(p)))$$

7.4 Summary

In this chapter constraints on geographic information have been discussed in the context of formalization into first order predicate logic. It has been shown by the examples that the majority of constraints can be specified in first order logic.

Chapter 8

Formalizing Constraints on Geographic Information

Abstract: This chapter deals with the formalization of constraints on geographic information. The specification language presented in Chapter 4 is extended by adding language elements which can be used to formulate complex constraints on conceptual models, or more precisely on objects instantiated by a conceptual model. Five types of constraints on geographic information are identified, and the formalization of these into predicate logic is discussed. The syntax and model-theoretic semantics of the constraint language is presented. Furthermore, examples illustrating how constraints can be expressed formally as GeoSML statements and automatically translated into SQL statements are given.

8.1 Introduction

An important aspect of the production of geographic information is to ensure that the produced information is consistent and homogeneous. Data content specifications are used to describe entities to be represented in a data collection, and to evaluate whether the quality of the produced information is acceptable or not. In the previous chapters we have discussed informal and semi-formal specifications, and a markup language supporting a structured development of specifications was introduced. As mentioned, parts of these specifications can be implemented as constraints in production systems, and used to evaluate, whether the produced information has the intended content. For this purpose informal statements must be translated into a formal language.

Today constraints on geographic information are specified by cartographers and topographers in natural language and subsequently hard-coded into the production systems by programmers. There are several reasons why this approach is infeasible to maintain. Firstly, products are defined or redefined on a regular basis, resulting in changes in the number and content of constraints. This requires programming each time a new constraint is needed, or an existing should be changed. Secondly, the sources of information change from being only collected from aerial photos to including a variety of new sources, e.g. administrative updates from municipalities, and changes posted on web pages. Information from new sources is delivered in multiple structures and content, which makes it impossible, or at least very difficult, to require that all information must pass through a single application for validation. Thirdly, the existing approach requires the constraints to be defined at implementation level, meaning that they have to be redefined if changes to the database schema are introduced.

If there was a method cartographers and computer experts could use to define and maintain constraints in cooperation, the risk of misleading translations between specification and implementation languages would be reduced. Such a method is suggested in this chapter: a constraint language built on top of the grammar for conceptual models introduced in Section 4.4. The constraint language has three important properties, which make it able the solve to problems listed above: (i) it is close to natural language, so that it is easy to use, (ii) statements can be parsed and translated automatically into SQL, which makes validation of information possible, and (iii) the formulation on the basis of conceptual models ensures that statements are loosely coupled from the actual database implementation. The constraint language in GeoSML is an elaboration of the High Level Constraint Language developed in collaboration with Mads Johnsen [Christensen and Johnsen, 2005], who as a part of his master thesis project [Johnsen, 2005] specified and implemented a parser that can automatically translate constraints into SQL. The parser is implemented in

SWI-Prolog [Wielemak, 2007].

Constraints and integrity constraints are well-established topics within research on conceptual modeling and databases, and other attempts to create constraint languages can be found in the literature. The COLAN language [Bassiliades and Gray, 1995] has properties similar to those of GeoSML, but works on a data model called "P/FDM", which is a functional data model built on Prolog, and it therefore seems to be lacking a clear correspondence with the relational data model. Existing approaches to modeling geographic information are all restricted to expressing constraints as restrictions on binary relations between objects. Few attempts to specify complex constraints for geographic information in context of conceptual models have been made. An example is Brisboa et al [Brisaboa et al.,], where constraints are formulated by use of OCL [1] statements. OCL has a syntax with a programming language style, which makes it difficult to use for product specialists. Some geographic information systems offer functionality for specifying constraints. An example is the ArcGIS software family, where some capabilities of specifying topologic rules [Hoel et al., 2003] are available, but only a predefined number of topologic relations can be included in the constraints.

The organization of the chapter is as follows. In the next section we present the basic idea of introducing a formal constraint language, and the intended use and role in production systems are discussed. Section 8.2 discusses the requirements for specifying constraints on geographic information, and it is shown how constraints can be formalized in predicate logic. In Section 8.2.1 the properties of the constraint language included in GeoSML are described. The mathematical details of Pierce algebras will not be presented in the thesis, but left to a paper in preparation [Nilsson and Johnsen, 2007]. Section 8.6 is an elaborate example that goes through the process of describing a conceptual model, writing constraints in GeoSML, mapping the conceptual model to an implementation model, and shows how constraints are translated into SQL.

[1] OCL is an acronym for the Object Constraint Language, and is an extension to UML. See [Demuth et al., 2001] for a specification of OCL.

8.2 Formalizing Constraints

This section presents a formal constraint language for specifying constraints on geographic information. The language has four important properties:

1. A syntax close to natural language, which makes it easy to use.
2. Expressions are formulated in the context of the underlying conceptual model, so that domain experts can intuitively formulate constraints. Hence they build on terms from the business domain and ensure that constraints are loosely coupled with the actual database implementation.
3. The language is based on formal semantics and has a number of well-formed criteria. Thus, basic syntax checking can be performed.
4. Statements which can be parsed and translated automatically into SQL, so that it is possible to embed the constraints in production systems and enables validation of the produced information.

There are several reasons for the introduction of a new constraint language:

(1) Predicate logic is not very easy to read or write, and the relation to the underlying conceptual model is not clear. The general idea of the constraint language is to specify restrictions on object types included in the conceptual model.

(2)The aim is to bridge the gap between constraints or business rules formulated in natural language and their implementation by use of e.g. SQL. The constraint language is designed to have syntax as close to natural language as possible, but still with a clear and unambiguous underlying semantic model.

(3) By introducing a map between the conceptual model and the logical model, changes in the low-level database schema can be accommodated in the map, thus leaving the conceptual model and ultimately the GeoSML constraints unchanged. The constraints can be updated by recompiling the constraints in the context of the new mapping and schema.

8.2.1 Basic Constructs

The basic constructs used in the formulation of formal constraints in GeoSML involve two top predicates: the `all-must` construct and the `no-may` construct. An example of the `all-must` construct is:

```
all greenhouse must building
```

which specifies that *"all greenhouses must be buildings"*, and means that it must be checked if a taxonomy relationship in the conceptual model actually holds between objects at database level. In general, the left-hand side of the "must" part specifies to which object type the constraint applies and the right-hand side specifies restrictions on the object type.

```
All residential area must contain building
```

The above expression is another example of the `all-must` construct, but including a relational path `contain building` on the right-hand side, which should be understood as *"contain at least one building"*, hence there is an implicitly given existential, quantification just after the "must" keyword. The corresponding predicate logic formula including the existential quantified building object at the right-hand side is as follows:

$$\forall\ a\ area(a) \rightarrow \exists\ b\ building(b) \wedge contain(a,b)$$

Using the `no-may` construct expresses class disjointness, meaning that the intersection between the sets on each side of the "may" keyword must be empty. An example of the `no-may` construct is:

```
no lake may contain building
```

The above constraint expresses that *"no lake may contain any building"*.

The two basic forms can be extended to express more complex requirements for the relations of the extensions of object types and the values of attributes. In the following sections these extensions are introduced and explanatory examples are given.

8.2.2 Paths

Paths of any length can be added to both sides of the "must" and the "may" keywords in the `all-must` and `no-may` expressions respectively. This is simply achieved by adding more relations and classes, and multiple paths can be bundled by using the "or" disjunction and the "and" conjunction operators. The only requirement is that the included paths must traverse predefined paths in the conceptual model. The reason is that translating constraints into SQL requires a map to accommodate relations between the conceptual model and the

logical model. If paths in constraint do not follow the predefined relationships in the conceptual model, then the parser will produce faulty SQL statements. Four examples of adding paths and using compound statements are given below:

```
all area type residential must
 contain building of-type residential
```

This first example states that *"residential areas must contain residential buildings"*. Note that the constraint does not restrict buildings of other types than residential to being within a residential area. To express this kind of restrictions the `no-may` pattern must be issued:

```
no area of-type commercial must
 contain building of-type residential
```

The above constraint expresses that *"commercial areas are not allowed to contain residential buildings"*.

Two statements that illustrate the usage of compound statements are:

```
All building has historicvalue and within cityborder must
 have taxation-type reduced
```

which states that *"buildings of historical value and that are inside a city are subject to a reduced taxation"*.

```
All building within commercial area must
 be owned-by company and type commercial
```

8.2.3 Alternative Quantifiers

As mentioned the object types in expressions on the right-hand side of the must keyword are implicitly existential quantified. Other quantifications can also be used in relational paths, but these need to be defined explicitly. There are three ways to express alternative quantifications over sets:

- the `All` keyword expressing universal quantification
- the `Solely` keyword expressing limiting relations to a single object type
- one of the `atleast`, `atmost`, and `exactly` keywords expressing numerical quantification

8.2.3.1 The `all` Keyword

The `all` keyword expresses universal quantification, i.e. that all objects with some specified properties must participate in the relations issued just before the `all` keyword. An example of the use of the `all` keyword is:

```
no area must contain all building
```

Expressing that *"no areas are allowed to contain every building"*. By continuing the path expression after the object type containing the `all` keyword, the objects on the right-hand side of the **all** keyword can be further restricted by adding relation paths to the class, e.g.:

```
No area must contain all building type residential
```

This constraint expresses that *"no areas should contain all buildings which are of the type residential"*.

8.2.3.2 The `solely` Keyword

The `solely` keyword limits the objects participating in a relation to being of a particular type, e.g.:

```
all residential area must contain solely building type residential
```

This constraint expresses that *"all residential areas must contain solely residential buildings"*. The "solely" keyword expresses that residential areas must only contain residential buildings and nothing else.

The `solely` keyword may also be used as an "exclusive or" operator:

```
all area must contain solely building type 'residential'
 or contain solely building type 'commercial'
```

which states that *"areas must either contain residential buildings or contain commercial buildings, and not combinations of the two types"*.

8.2.3.3 The `atleast`, `atmost`, and `exactly` Keywords

Numerical quantifications are used to restrict the number of objects which are allowed to participate in a relation. Examples of the use of numerical quantifiers are:

```
All area type residential must contain exactly 3 building
```

which states that *"all residential areas must contain three and only three buildings"*.

```
all area type residential must contain atleast 4 building of
 type residential and contain atmost 1 building type commercial
```

which states that *"all residential areas must contain four residential buildings or more, and no more than one commercial building"*.

8.2.4 User-defined Variables

In natural language references to the same object may be implicitly given in two parts of a sentence. This kind of sentences is called anaphora, also known as *donkey-sentences*, after the most popular example of an anaphoric statement: *"All farmers that own a donkey beat it"* [Benthem, 1986].

Constraints with anaphoric elements are difficult to formalize and to translate into SQL. There are two reasons for these difficulties: first the used formalism must be to able express that it is the same donkey, in the case of the donkey-sentence, that the farmer owns and beats, second the scoping of the variables that must be introduced to solve the first problem is not straight forward. A constraint formalizing the donkey-sentence could be:

```
all farmer own donkey must beat donkey
```

it is tempting to translate this constraint into the following predicate formula:

$$\forall x farmer(x) \wedge \exists y(donkey(y) \wedge own(x, y)) \rightarrow \exists y(donkey(y) \wedge beat(x, y))$$

but this does not really express the intended meaning, hence the statement does not capture that it is the same donkey that is both owned and beaten (the y is locally scoped on both sides of the implication). Therefore, we need

to introduce universal quantified variables and the scope of the variables must reach both sides of the `must` keyword. The constraint with variables:

```
all farmer own donkey D must beat donkey D
```

which could be translated into the following expression:

$$\forall x, D(farmer(x)donkey(D) \wedge own(x, D) \rightarrow beat(x, D))$$

An example from the geographic information domain is the sentence : *"All areas intersected by a road segment must also contain a building which is intersected by the road segment"*. By use of variables it can be expressed in GeoSML as

```
all area intersectedby road R must
 contain building intersectedby road R
```

Another example is:

```
All building B must equal building B
```

which expresses that *"all buildings must be equal to themselves"*. It is the user-defined variable "B" that enables the designer to express that the two buildings mentioned in the constraint should be the same.

8.2.5 Value Comparison

Simple value comparison of attributes can be included in constraints. Value comparison is used to specify domain constraints. There are four keywords supporting the specification of domain constraints: `lessthan`, `greaterthan`, `equalto`, and `oneof`. The first three for range domains and the fourth for coded value domains. Examples are:

```
all building must have height greaterthan 2
```

```
all roadsegment must have numberoflanes greaterthan 1
```

```
all building must have type oneof residential, commercial, industrial
```

8.2.6 User-defined Predicates

Relationships and attributes indirectly given in the geographic information must be calculated before they can be included in constraints. Implicit information cannot be accessed by the constructs introduced in the previous sections, because implicit relations and attributes cannot be directly mapped from the conceptual model, which prevents to the logical model the parser from producing useful SQL statement.

The tool for including implicitly given information in the constraints is user-defined predicates. The evaluation of these predicates is delegated to the underlying database management system or to an application layer just "above" the database.

An example of a constraint with a user-defined predicate is as follows:

```
      all building A hastype T touch Building B hastype T
must havezdifferencelargerthan(A,B,5)
```

This constraint states that neighboring buildings of the same type must have a height difference of more than five meters.

If the height differences were stored in the database, e.g. by creating a view including references to two touching buildings, and their mutual height difference, and the view included a reference to the view in the conceptual model, then the corresponding constraint could be something like:

```
      all touchingbuildingwithzdiff firstbuilding building type T and
secondbuilding building type T must have value atleast 5.
```

Whether user-defined predicates or views should be used depends on the situation. The advantage of user-defined predicates is that they can be reused, if they are implemented correctly, e.g. the havesdifferencelargerthan(a,b,z) can be issued also for forests if an analogous constraint is needed. The advantage of using views is that is minimized the need for variables.

8.2.7 Definitions

As illustrated in Section 8.2.7 constraints can be formulated more compactly to enhance the readability if definitions of complex parts are separated in indepen-

dent definitions. To support formal definition GeoSML includes the `is-defined-as` keyword, which keyword can be included in expressions of the form:

```
defined term is-defined-as expression
```

where `expression` is a path in the conceptual model including the optional constructs presented in the above sections, e.g. numerical quantifications and user-defined predicates. The example from Section 8.2.7 can then be reformulated as:

(i) A connected pipe is defined as *"a pipe having all their connection points included in a connection"*:

```
connected-pipe is-defined-as pipe is-connected-to
atleast 2 pipe
```

(ii) A terminated pipe is defined as *"the pipes that connected to a pipe in one end and to a terminator in the other end"*:

```
terminated-pipe is-defined-as pipe is-connected-to exactly 1
pipe and is-connected-to exactly 1 terminator
```

The definitions can be used in constraints which are much easier to read than if the full expressions were included instead. E.g. it can easily be stated that *"All pipes in service must either be a terminated pipe or a connected pipe"*:

```
all pipe hasstatus 'in-service' must connected-pipe
or terminated-pipe
```

8.3 Formal Description

In this section a formal definition of the constraint language explained in the above sections is given. Both a concrete and an abstract syntax is given. The concrete syntax serves as a reference to the constraint language, describing the structure of allowed statements, while the constructs used in the abstract syntax will be used in the semantics and the translation strategy of constraints. In this thesis we will account for the formal semantics.

8.3.1 Concrete Syntax

The following grammar defines the syntax of the constraint language embedded in GeoSML.

⟨expression⟩::=**all**⟨typeexp⟩ **must** ⟨typeexp⟩ |
no ⟨typeexp⟩ **may** ⟨typeexp⟩

⟨typeexp⟩::=⟨object typeID⟩[⟨reltype⟩] |⟨user-predicate⟩ |⟨vartypeexp⟩

⟨vartypeexp⟩::=⟨object typeID⟩ ⟨variable⟩ [⟨reltype⟩]

⟨reltype⟩::=⟨relationID⟩ [⟨int quant⟩] ⟨typeexp⟩ [⟨operator⟩ ⟨reltype⟩] |
⟨relationID⟩ ⟨vartypeexp⟩ [⟨operator⟩ ⟨reltype⟩] |
⟨attributeID⟩ ⟨value⟩ |
⟨attributeID⟩ ⟨numerical relation⟩ ⟨integer⟩ ⟨value⟩

⟨int quant⟩::=**all** |**solely** | ⟨numerical relation⟩⟨integer⟩

⟨operator⟩::= **and** | **or** | **andnot** | **ornot**

⟨numerical relation⟩::=**exactly** |**at least** |**at most**

⟨variable⟩::= **A** | ... |**Z**

⟨integer⟩::= **1** |**2** | ...

...

⟨user-predicate⟩::=⟨user-predicate name⟩⟨parameter list⟩

⟨value⟩::= ...

The values in ⟨object typeID⟩, ⟨relationshipID⟩, ⟨attributeID⟩ and ⟨value⟩ depend on the corresponding conceptual model, while values in ⟨user-predicate⟩ depend on the database model.

Statements made in the constraint language must conform to the above grammar and to a number of other requirements which are not expressed in the grammar. These requirements include amongst other the following items:

- Expressions must use paths that already exist in the conceptual model
- Each variable must appear at least twice.
- Occurrences of a variable must refer to the same type of class/attribute.
- Expressions are only allowed to start with a class expression.

- Expressions are not allowed to have compound operators on the left-hand side of the expression.

A complete list of requirements for well-formed expressions can be found in [Johnsen, 2005].

8.3.2 The Impact on the GeoSML Grammar

To include formal constraints at the conceptual level and to relate other specification elements to formal constraints a few changes is made to the the grammar needs to be extended. This is done by changing the `constraint` symbol to include both natural language statements and formal constraints. Furthermore, the ⟨goalassign⟩ symbol from the requirement modeling level is substituted by adding alternative definitions to the d⟨relexp⟩. This is done by adding a definition which relates curtain elements in domain and requirement models to elements in conceptual models.

⟨constraint⟩::= [**concerning** (⟨object typeID⟩)*][**Natural Language:**]
⟨statement⟩ [**Formal:** ⟨expression⟩]

⟨repexp⟩::=⟨entity typeID⟩ **is represented by** ⟨object typeID⟩
using (⟨selection rule⟩)*
(instantiation rule)*
(⟨life cycle rule⟩)*
Attribute Values::=⟨attributID⟩ [**has default value** ⟨value⟩] **is determined by** (⟨representation rule⟩*) |
⟨drID⟩ **is represented by** ⟨crID⟩
using (⟨selection rule⟩*)
(⟨instantiation rule⟩*)

⟨selection rule⟩::=⟨statement⟩

⟨instantiation rule⟩::=⟨statement⟩

⟨life cycle rule⟩::=⟨statement⟩

⟨representation rule⟩::=⟨statement⟩ |
⟨drelement⟩ **is implemented by** ⟨conceptual element⟩

⟨conceptual element⟩::=⟨object typeID⟩ |⟨attribute typeID⟩ |⟨constraintID⟩
|⟨crID⟩

⟨drelement⟩::=⟨definitionID⟩ |⟨designationID⟩ |⟨assertionID⟩ |⟨entity typeID⟩
|⟨drID⟩ |⟨propertyID⟩ |⟨goalID⟩

8.4 Model-theoretic Semantics

This section is a formal specification of the model theoretic semantics for GeoSML. The model uses the constructs introduced in the abstract syntax in Section 8.4.1 as a basis for the set-theoretic model. The model focuses on the two elementary forms and their extension into relation paths. Variables and user-defined predicates will not be treated.

8.4.1 Abstract Syntax

To be able to describe the semantics of the constraint language, an abstract grammar is introduced. This grammar introduces an abstraction between the concrete syntax and the semantics which is defines in the following sections.

⟨E⟩::=**allmust(**⟨CE⟩,⟨CE⟩**)** | **nomay(**⟨CE⟩,⟨CE⟩**)**

⟨CE⟩::= **class(**C_{id},⟨RC⟩**)** | **varclass(**C_{id},V_{id},⟨RC⟩**)** | ⟨F⟩ | ⟨RC⟩

⟨RC⟩::=**exrelclass(**R_{id},⟨CE⟩**)** | **allrelclass(**R_{id},⟨CE⟩**)** | **solelyrelclass(**R_{id},⟨CE⟩**)**
| **numrelclass(**Int,⟨Comp⟩,R_{id},⟨CE⟩**)** | **attribute(**A_{id},⟨Comp⟩,Val**)**
| **or(**⟨RC⟩,⟨RC⟩**)** | **and(**⟨RC⟩,⟨RC⟩**)** | **ornot(**⟨RC⟩,⟨RC⟩**)** | **andnot(**⟨RC⟩,⟨RC⟩**)**
| []

⟨F⟩::= **unarypredicate(**P_{id},V_{id}**)** | **binarypredicate(**P_{id},V_{id1},V_{id2}**)**

⟨Comp⟩::= **eq** | **ge** | **le**

8.4.2 Basic Form

Two basic sentence forms which are represented by the two predicates *"allmust"* and *"nomay"* can be used to write constraints. The *"allmust"* defines a class inclusion, meaning that the set on the left-hand side must be a subset of the set on the right-hand side. The *"nomay"* defines that no elements in the left-hand side set must be a member of the set on the right-hand side, therefore the intersection between the two sets must be the empty set:

$$[\![\ \text{allmust(P,Q)}\ ]\!] = [\![\ \text{P}\ ]\!] \subseteq [\![\ \text{Q}\ ]\!]$$

$$[\![nomay(P,Q)]\!] = [\![P]\!] \cap [\![Q]\!] = \emptyset$$

8.4.3 Relational Paths

By a recursive definition of class expressions "CE" the basic forms can be extended with relational paths. The formal semantics of relational paths can be defined by using the two-sorted Peirce Algebra suggested in [Brink et al., 1994].

Class expressions defined as a class and a relation contain a relation class ("RC") as the second argument:

$$[\![class(C_{id}, RC)]\!] = \{x | x \in C_{id} \wedge x \in [\![RC]\!]\}$$

A relation class can be a "simple" existentially quantified relational path (an "*exrelclass*" construct in the abstract syntax). The interpretation of the "*exrelclass*" is as follows:

$$[\![exrelclass(R_{id}, CE)]\!] = \{x | \exists y, y \in [\![CE]\!] \wedge (x,y) \in R_{id}\}$$

which can be read as *"For entities which there exists a relation R_{id} to an entity of class CE"*.

The class expression "CE" is either a class identifier or another relational path. This construct introduces the recursive definition of class expressions and is referred to as the "Peirce product" [Brink et al., 1994], which is an operator taking two arguments: a binary relation and a set. The Peirce product returns a set.

Relation expressions can take other forms than existentially quantified relational paths. Quantification can also be achieved by using the solely keyword (*"solelyrelclass"* in the abstract grammar). The meaning of this construct can be

explained by a variation of the Peirce Product:

$$[\![solelyrelclass(R_{id}, CE)]\!] = \{x | \forall y, y \in (x, y) \in R_{id} \rightarrow y \in [\![CE]\!]\}$$

The "*solelyrelclass*" is similar to the *"exrelclass"*, except that it uses a universal quantifier instead of an existential one. The set expression says: *"All entities which are related through the relation R_{id} to entities of solely class CE"*.

The `all` keyword ("*allrelclass*" in the abstract syntax) is defined in the same way but with the implication reversed:

$$[\![allrelclass(R_{id}, CE)]\!] = \{x | \forall y, y \in [\![CE]\!] \rightarrow (x, y) \in R_{id}\}$$

This can be read as *"all entities related to all entities of class CE through R_{id}"*.

The last types of quantifications allow relational paths to use numerical quantifications. The meanings of these are:

$$[\![numrelclass(eq, N, R_{id}, CE)]\!] = \\ \{x | \exists y, card(\{y | R_{id}(x, y) \wedge y \in [\![CE]\!]\}) = N)\}$$

$$[\![numrelclass(le, N, R_{id}, CE)]\!] = \\ \{x | \exists y, card(\{y | R_{id}(x, y) \wedge y \in [\![CE]\!]\}) < N)\}$$

$$[\![numrelclass(ge, N, R_{id}, CE)]\!] = \\ \{x | \exists y, card(\{y | R_{id}(x, y) \wedge y \in [\![CE]\!]\}) > N)\}$$

The three numerical quantifications are very similar, except that there is a difference in the comparison to the given integer 'N'. The top numerical quantified set expression can be read as *"the set of entities to which exactly N entities of class CE is related through R_{id}"*.

8.4.4 Operators

To be able to specify compound statements, operators between relational paths are allowed. The operators are *"and"*, *"or"*, *"andnot"*, and *"ornot"* in the abstract syntax. The semantics is also expressed in set theory by using inter-

sections and unions of sets:

$$[\![and(P,Q)]\!] = [\![P]\!] \cap [\![Q]\!]$$
$$[\![or(P,Q)]\!] = [\![P]\!] \cup [\![Q]\!]$$
$$[\![andnot(P,Q)]\!] = [\![P]\!] \cap [\![Q]\!]^C$$
$$[\![ornot(P,Q)]\!] = [\![P]\!] \cup [\![Q]\!]^C$$

8.4.5 Attribute Values

Finally, there is the option that a relational path could in fact be an attribute, with the semantics seen below:

$$[\![attribute(A_{id}, eq, value)]\!] = \{x | A_{id}(x, value)\}$$
$$[\![attribute(A_{id}, le, value)]\!] = \{x | \exists y, A_{id}(x,y) \wedge y < value)\}$$
$$[\![attribute(A_{id}, ge, value)]\!] = \{x | \exists y, A_{id}(x,y) \wedge y > value)\}$$

$$[\![attribute(A_{id}, oneof, valueset)]\!] = \{x | \exists y, A_{id}(x,y) \wedge y \in valueset)\}$$

8.5 Translating GeoSML Constraints to SQL

This section gives an overview of the process of translating GeoSML constraints into SQL. Constraints are formulated in the context of a conceptual model. Therefore, the compiler needs access to descriptions of the conceptual model, a logical model of the database schema, and a mapping between the two. By introducing a map between the conceptual model and the logical model, changes in the low-level database schema can be accommodated in the map, thus leaving the conceptual model and ultimately the constraints unchanged. The implementation of the constraints can be updated by recompiling the constraints in the context of the new mapping and database schema. The syntax for conceptual models is developed in Section 4.4 and the syntax for constraints is given in Section 8.3. The following sections illustrate the logical schema, and the mapping between the conceptual model and the logical model is specified. The syntax for these two specification parts is straightforward and will not be explained in

detail. The compiler that translates constraints into SQL is developed by Mads Johnsen in Prolog using the SWI-Prolog toolkit [Wielemak, 2007]. Detailed specifications of the translation process and the syntax of the logical model and mappings between the conceptual model and the logical model are found in [Johnsen, 2005].

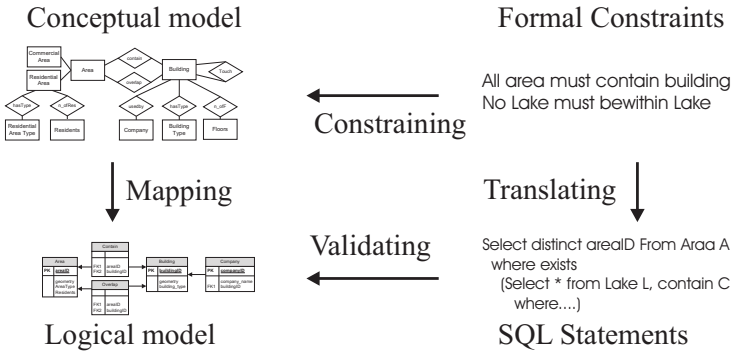

Figure 8.1: Basic principle for the translation process

The principle for translating constraints is seen in Figure 8.1.

1. Develop a conceptual model
2. Specify formal constraints
3. Develop a logical model
4. Mapping between the conceptual and the logical model
5. Translating GeoSML constraints into SQL

8.6 Examples - Translation of Constraints

In the following section, example from Chapter 4 is continued by adding some formal constraints to the conceptual model, developing a logical model for a database that can persistently store information specified by the conceptual model, and supplying a mapping between the conceptual model and the logical model. Finally, the SQL statements produced by the parser are shown.

8.6.1 Constraints

The five constraints included in the specification are formalized and added to the conceptual model which results in the following specification:

```
Conceptual model My Small Topographic Map
    Constraint concerning building:
    Natural language: Buildings must not overlap
    Formal: No building must overlap building
    Constraint concerning building, area:
    Natural language: Buildings must not overlap areas
    Formal: No building must overlap area
    Constraint concerning road_segment:
    Natural language: Roads can either be high ways, major roads or
    minor roads.
    Formal: All roadsegment must type highway or major industrial or
    minor
    Constraint concerning Building:
    Natural language: If two buildings of the same type is
        neighbors then the z-difference between the two
        buildings must be larger than 5 meters.
    Formal: all Building A type T neighbor Building B
    type T must havezdifferencelargerthan5(A,B)
    Constraint concerning building, road_segment, area
    Natural language: Road segments  must not intersect areas,
        unless there is a building within the area which is also
        intersected by the road segment.
    Formal all area intersect road_segment R must \\
        contain building intersect road_segment R
```

8.6.2 Logical Data Model

To be able to translate GeoSML constraints into SQL, a map between the conceptual and the logical model must be accommodated. A logical model defines the layout of database tables and views and the relations among these, specified by primary and foreign keys.

The parser expect spatial relations to be represented as tables or views on tables, in proceeding versions foreign keys and functional relationships, like spatial

relations, must be supported directly. Below, the GeoSML statements that describe the logical model are listed.

```
table(Area,[areaID,geometry,areatype,residents])
table(Building,[buildingID,numoffloors,geometry,buildingtype])
table(Roadsegment,[roadsegmentID,roadtype,geometry,roadname])
table(baIntersect,[buildingID,areaID])
table(brIntersect,[buildingID,roadsegmentID])
table(baOverlap,[buildingID,areaID])
table(bbOverlap,[buildingID,buildingID])
table(Touch,[buildingID,buildingID])
function(havezdifferencelargerthan5,[[A,buildingID],[B,buildingID]],zdiff([A
buildingID],[B,buildingID])
```

The following list defines a mapping between the logical model above and the conceptual model from the example on page 62.

```
classmap(area, Area,[],[areaID])
classmap(building, Building,[],[buildingID])
classmap(roadsegment, Roadsegment,[],[roadsegmentID])
valuemap(num_of_floors,numoffloors)
valuemap(building_type,buildingtype)
valuemap(areatype,areatype)
valuemap(num_of_residents,residents)
valuemap(roadname,roadname)
valuemap(roadtype,roadtype)
relmap br_intersect brIntersect, [buildingID],[roadsegmentID]
relmap ar_intersect arIntersect, [areaID],[roadsegmentID]
relmap ba_overlap baOverlap, [buildingID],[areaID]
relmap bb_overlap bbOverlap, [buildingID],[buildingID]
relmap touch Touch, [buildingID],[buildingID]
relmap contain Contain, [areaID],[buildingID]
```

It is straight-forward to specify the two type of models. The description of the logical model include two type of statements: `table` and `function`. The `table` keyword describes tables in the database by the name of the tables and a list of attributes for each table. The `function` keyword describes each function implemented by the database system by the name of the function, a map between variable in the formal constraint and the variables used in the database function, and the expression calling the database function.

Mappings between a conceptual model and logical model are described by four

three of statements: `Classmap`, `relmap`, `valuemap`, and `type`. A `classmap` relates an object type to a table in the data base. This is done by four parameters: The identifier of the object type, the name of the table, an optional condition identifying the rows in the table, and a list of the attributes defining the primary key of the table. Relations in the conceptual model are mapped to the logical model by the `relmap` keyword, taking 4 parameters: The identifier of the relation in the conceptual model, the name of the table, a list identifying the first primary key, and a list identifying the second primary key. The `valuemap` is used to bind attributes in an object type to a column in a table. This is done by two parameters: The identifier of the attribute type in the conceptual model and the name of the column which store the value of the attribute.

Further details and a formal specification of the syntax for specifying logical models and mappings among conceptual model can be found in [Johnsen, 2005]. It shall be noticed that minor changes are made compared to the definitions made by Johnsen. This is due to the introduction of unique identifiers of object types, attributes, and relationships in this thesis.

8.6.3 Generating SQL

Using the conceptual model, the logical model, and the mapping between the two, the Prolog parser can produce SQL statements which implement the GeoSML constraints from section 8.6. The translated constraints are built up as SQL statements so that if the constraints are fulfilled, the execution of the translated query will return an empty set of rows. The first example is the "No building must overlap building", which is parsed into the following SQL query:

```
SELECT * FROM Building a WHERE EXISTS(
    SELECT *
    FROM bboverlap b
    WHERE b.buildingID1 = a.buildingID
    AND EXISTS(
        SELECT *
        FROM Building c
        WHERE c.buildingID = b.buildingID2
```

The second example "No building must overlap area" is analogous to the first but constrain the relations among two tables. The constraint is parsed into the following SQL query:

```
SELECT * FROM Building a WHERE EXISTS(
    SELECT *
    FROM baOverlap b
    WHERE b.buildingID = a.buildingID
    AND EXISTS(
        SELECT *
        FROM Area c
        WHERE c.areaID = b.areaID
```

The general idea is that every class and relation in the conceptual model correspond to a table in the underlying database. The SQL expressions are built up as nested SELECT-FROM-WHERE clauses. As seen, classes and relations are no longer referred to by their names in the conceptual model; instead the actual table names in the database model are used. A comparison is made with the set of attributes that constitutes the primary key. Let us look at a more advanced query, namely the one corresponding to the constraint: "All buildings must type commercial or type highresidential or type lowresidential". The corresponding SQL query to the constraint is:

```
SELECT * FROM Roadsegment a WHERE NOT( a.roadtype =
'highway'
    OR
    a.roadtype = 'major'
    OR
    a.building\_type = 'minor')
```

As seen, the general structure is the same as in the first query. The translation of the operators in GeoSML is straightforward, since the used operators are available in SQL. The relational paths are not parsed into any "EXISTS" statements, since the relational paths are actually properties of the building class, resulting in much simpler SQL. A final example with user-defined variables and predicates, namely the constraint stating: "all building A type T touch building B type T must havezdifference(A,B,5)", will result in the following SQL code:

```
SELECT * FROM Building a
    WHERE EXISTS(
        SELECT * FROM Building b
            WHERE (b.buildingID != a.buildingID)
            AND EXISTS(
                SELECT * FROM Touch c
                    WHERE (c.buildingID1 = a.buildingID)
```

```
                AND (c.buildingID2 = b.buildingID)
                AND (b.type) = a.type
                AND NOT ('zdiff'(a.buildingID,b.buildingID))))
```

The SQL follows the same structure as the other examples, although there are some extra clauses in the innermost WHERE clause. The first of these expresses that the two buildings should be different, due to the different variables used in the constraints. The next clause expresses that the two buildings should be of the same type. This is the translation of the "T" variable. The third clause is the translation of the user-defined predicate, we can see how the user-defined predicate is translated straight into SQL.

A final example of translating constraints into SQL concerns the relation among road segments, buildings, and areas. The formal constraint states that `all area intersect road R must contain building intersect road R`, which has the following SQL translation:

```
SELECT *
FROM Roadsegment a
WHERE EXISTS
   SELECT *
   FROM Area b
   WHERE EXISTS(
      SELECT *
      FROM arIntersect c
      WHERE c.roadID = a.roadID
      AND c.areaID = b.areaID)
AND NOT EXISTS(
   SELECT *
   FROM Contain b
   WHERE b.areaID = a.areaID
   AND EXISTS
      SELECT *
      FROM Building c
      WHERE c.buildingID = b.buildingID
      AND EXISTS(
         SELECT *
         FROM brIntersect d
         WHERE d.buildingID = c.buildingID
         AND d.roadID = a.roadID))))
```

This SQL statement returns the road segments which intersect an area and not intersecting a building contained in that area.

8.7 Summary

The aim of the presented work is to enable product and production specialists to write formal constraints that can automatically be implemented in production software. We conclude that it is possible to define a language in which constraints can be formulated by using a syntax with resemblance to natural language. We believe that the constraint language in GeoSML has an expressiveness that can fulfill most requirements on writing constraints on geographic information. Even though the specification of formal constraints using GeoSML is quite easy to understand, it still requires some basic knowledge of the syntax and what can actually be achieved.

3d

Chapter 9

Conclusion

Abstract: This final chapter includes three sections. Section 9.1 gives a summary of the achieved results seen in the context of the design goals identified in Chapter 3. Section 9.2 draws the conclusions that can be made from the presented work and the elements of GeoSML are discussed and compared with the overall hypothesis stated in Section 1.2.2. Section 9.3 lists suggestions for future work.

9.1 Results

The main result obtained in this thesis is a framework for developing geographic information specifications. This framework includes a specification language which suggests a best-practice for developing and structuring specifications. The language, which is called the Geographic Information Specification Markup Language (GeoSML), provides a new principle for structuring and formalizing geographic information specifications. The aim has been to define a method which supports the development of geographic data collections – from the perception of the initial idea until the information is captured and stored in a database, and to provide a formal constraint language which is accessible for non-programmers.

In Chapter 3 the following design goals were identified. These goals are: GeoSML must provide the support for:

1. structuring of specifications written in natural language
2. cooperative design
3. separation of domain descriptions and design decisions
4. inclusion of requirements in the specification
5. incorporation of formal constraints in conceptual models
6. incorporation of quality reqirements in conceptual models

The following sections summarize the impact each of these design goals has had on the design of GeoSML.

9.1.1 Structuring of Natural Language Specifications

The production of high-quality geographic information requires clear and unambiguous specifications. A first step to achieve this is to provide a predefined structure of natural language specifications.

The approach to reaching this design goal has been to regard a specification as a set of statements. Each statement can be classified according to the roles it plays in the interpretation process. GeoSML provides a classification scheme that can be used to mark up statements according to these roles. The main

part of this scheme is illustrated by the representation system in Figure 4.1 and is concretized by the grammars for domain models, conceptual models, and the mappings between the two (see pages 48, 57, and 68 respectively).

9.1.2 Support of Cooperative Design

The second design goal of GeoSML is to support cooperative design processes. This design goal is motivated by the need for involving a variety of persons in the design of a data collection, each providing a unique knowledge. Examples of the required kind of persons are users of geographic information, cartographic experts, information architects, and computing engineers, who must be able to work together in the design process.

To achieve this goal, GeoSML includes a number of integrated views of the data collection being developed. Each view provides a unique understanding of the data collection, from user requirements, domain descriptions, over information design to implementation considerations. These views are concretized by the following modeling levels:

- Domain model
- Conceptual model
- Requirement model
- Logical model
- Mappings between these model types

9.1.3 Separation of Domain Descriptions and Design Decisions

The idea of separating descriptions of a domain and the design of a data collection is to force designers to distinct between descriptions of the reality, which the data collection seeks to represent, and descriptions of how the entities within the domain are represented in the data collection. This differentiation fails when it is unclear if a statement describes the reality or some properties of the data collection being developed.

The design goal is reached by introducing the notions of `domain model` and `conceptual model` into the same framework, and providing mechanisms for relating terms in two models by using the `is-represented-by` keyword.

9.1.4 Inclusion of Requirements in the Specification

The inclusion of requirements in the specification is motivated by the need for gathering and structuring potential user's needs, and to motivate the decisions which lead to the design of the data collection. Integrating requirements into the specification ensures that the design of a geographic data collection builds on identified and acknowledged needs of the potential users.

To meet this design goal, the GeoSML framework was extended with the `Requirement Model` keyword and accompanied by several symbols for structuring requirements in so-called refinement trees (see page 71). Furthermore, the "leaves" on the trees can be related to elements in a conceptual model and thus document the motivation of each element. The relations between corresponding requirements and conceptual models are defined by the `is implemented by` relation.

9.1.5 Formal Constraints

A key design goal for GeoSML is to support the formalization of statements included in a specification. It has been chosen to focus on the formalization of constraints which produced information must comply.

The result is a formal constraint language that enables designers to express formal constraints in the context of pre-existing conceptual model in a language with a syntax that resembles natural language. In Chapter 8 it is also shown how formal constraints can be translated into an ordinary query language like SQL.

9.1.6 Quality Requirements

The quality of the information is important when the fitness for use is evaluated against a given application. To enable such an evaluation the quality must be known and well described. To ease the access and availability of quality descriptions, a design goal of GeoSML is to include structured approaches when stating

requirements for the quality of the produced information, and to integrate these requirements directly into the specifications.

The solution has been to extend the conceptual modeling level in GeoSML with elements for acceptable quality level (AQL) and Quality Element Requirement (QER). AQL includes requirements for the completeness and accuracy of the stored information. QER is the requirements for the quality parameters describing the quality of a data collection (see Chapter 6).

9.2 Overall Conclusion

This section accounts for the extent to which the main hypothesis, as stated in Chapter 1, holds. The hypothesis will be presented and compared to the design goals stated in Chapter 3 and the results as presented in the previous section.

The main hypothesis as it is formulated in Section 1.2.2 on page 3:

> ***The knowledge embedded in natural language specification for geographic information can be represented in formal computational structures.***

To claim that the stated hypothesis holds, requires proofs that all the elements in a geographic information specification can be represented by formal computational structures.

Many examples of formal statements have been given for the various types of descriptions included in geographic information specifications. Although, it is too far-reaching to say that proof is given that the hypothesis holds. First of all, because GeoSML by nature is a semi-formal approach and natural language statements are allowed, but also because it is difficult to account for all the possible type of statements that may be included in a specification. The focus in the presented work has been on specifications for object-based geographic information and primarily topographic data collections. Other kinds of geographic information, such as raster images or terrain models, may require statements which are not treated in this thesis.

However, the achieved results illustrate that the identified specification elements within the domain of topographic mapping can be formalized. Throughout the thesis it has been shown how various type of statements can be formalized by first order predicate logic and traditional data modeling tools. Also, it has

been possible to develop a formal constraint language based on a subset of first order predicate logic, which has a syntax resembling natural language. A formal semantics for this language has been given, and a parser which can translate the formal constraints into SQL is available.

The inclusion of natural-language based statements in GeoSML is motivated by the design goal which requires GeoSML to support cooperative design processes, i.e. that the language must include and integrate a number of views on the data collection being developed. Views which are all required to include all the information necessary to design high-quality data collections. The goals requiring GeoSML to be formal and to support cooperative design are somehow conflicting. On the one hand specifications need to be formal to be precise, on the other hand persons with no particular training in writing formal statements are important to the specification and design of data collections.

Despite of these two contradicting goals, GeoSML attempts to reach both design goals. The aim has been to develop a specification language which primarily focuses on formal aspects of specifying and still being operational and understandable for most of the persons involved in the design process. The approach for closing the gap between natural language and formal specification has been to introduce tools for organizing and structuring of natural language statements, e.g the `concern` keyword which are used to explicitly point out the most important terms in a statement. The advantage of using the `concern` keyword is that statements can be retrieved in a more predictive manner than using free-text searches. Another example is the `is implemented by` keyword which provides a mechanism for relating natural language statements in domain and requirement models to formal constraints and other elements in a conceptual model.

Finally, it must be concluded, which is not that surprising, that the creation of geographic information still requires human interaction and intuition. Formal languages like GeoSML do not substitute skilful persons in the design and production processes, who have an in-depth understanding of the nature of representing geographic entities as map objects and the experience of interpreting the source material – like arial photos, when designing and producing geographic information.

We believe, that the principle on which GeoSML builds and the method for developing formal and structured specifications, which it constitute, are important tools for supporting the process of establishing high-quality geographic data collections, both at the design level and when the information is created.

9.3 Future Work

Even though the GeoSML framework includes a large number of facilities for specifying geographic information, it is still a first iteration toward a specification language designed to meet the requirements for structuring and formalizing specifications especially suitable for the production of geographic information. This section suggests projects that may be included in future iterations, in the task of improving the usability and functionality of GeoSML.

Elaborated Examples
An elaboration of the examples in this thesis would properly help designers to understand the intensions of GeoSML better. Developing such examples would also help the identification of possible weaknesses of the framework and the requirements to an improved version. Another result from developing more elaborated examples could be a step-by-step manual or a users guide for GeoSML, which in a concrete manner describes the best-practice for developing geographic information specification, suggested in this thesis.

Case Tool
Developing a case-tool will increase the useability of GeoSML and make GeoSML even more accessible to non-programmers. Without a case tool it will be difficult to handle the large amount of information required to describe e.g. topographic maps.

An approach for developing a case tool is to redefine the context-free syntax which has been used when defining GeoSML and bring it into the context of UML or by changing the markups into an XML based one.

Development of Additional Parsers
To improve the capabilities of implementing GeoSML based specifications in production software, the parser which has been developed to translate GeoSML constraints into SQL statement could be extended or modified to target other implementation languages. This could for example be ArcObject for the ArcGIS family or MapBasic for MapInfo. Also, the generated code could target one of the Java-based toolkits for checking the topology e.g. the Java Topology Suite [Davids and Aquino, 2003].

Versioning of Specifications
It would be an advantages if GeoSML included mechanisms for versioning. This subject is of great importance but has only briefly been touched upon in this thesis. Versioning a specification should allow the definition of named versions e.g. the 3.20 version of the TOP10DK specification. Such a versioning system would require that transaction and valid time is recorded for each element in the

specifications and that a reliable numbering system for specification elements is introduced. Introducing a versioning system would also give capabilities for retrieving the changes made to a specification within a given time period.

There has been developed a first suggestion for a version system for the GeoSML framework. A paper describing this system is under preparation.

The Constraint Language
It is believed that the expressiveness of the constraint language will cover most needs for specifying formal constraints on conceptual models. Nevertheless, a number of modifications and improvements can be made. One is to improve the syntax so it would be even more natural language feel and look alike. This could for example be done by allowing synonyms for relation names.

As it is, topologic relations must be specified at the conceptual level before they can be incorporated in a constraint. This gives an overhead when new topologic constraints are specified. First, a the relation must be included in the conceptual model, then the relation to the logic model must be described, including creating a view facilitating the topologic relation. Finally, the constraint can be specified and translated into SQL. Is work flow could possible be simplified by introducing predefined object types from which all other object types inherit from and then specify a set of topologic relations to these predefined object types.

Bibliography

[Ackoff, 1962] Ackoff, R. (1962). Scientific method. page 179. John Wiley & Sons.

[Allen, 1983] Allen, J. F. (1983). Maintaining knowledge about temporal intervals. *Commun. ACM*, 26(11):832–843.

[Arango, 1989] Arango, G. (1989). Domain analysis: From art form to engineering discipline. pages 152–159.

[Baader et al., 2003] Baader, F., Calvanese, D., McGuinness, D. L., Nardi, D., and Patel-Schneider, P. F., editors (2003). *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press.

[Bassiliades and Gray, 1995] Bassiliades, N. and Gray, P. M. D. (1995). Colan: a functional constraint language and its implementation. *Data Knowl. Eng.*, 14(3):203–249.

[Benthem, 1986] Benthem, J. V. (1986). *Assays in Logical Sementics*. Reidel Publications.

[Bjørner, 1995] Bjørner, D. (1995). Software systems engineering - from domain analysis to requirements capture - an air traffic control example. Technical Report 48, Macau.

[Bjørner, 1999] Bjørner, D. (1999). A triptych software development paradigm: Domain, requirements and software. towards a model development of a decision support system for sustainable development. In *Festschrift to Hans Langmaack: Correct Systems Design: Recent Insight and Advances*, volume 1710 of *Lecture Notes in Computer Science*, pages 29–60. Springer-Verlag.

[Bjørner, 2003] Bjørner, D. (2003). What is a method? an essay on some aspects of software engineering. In *Programming Methodology*, volume 9, pages 175–203. Springer-Verlag, N.Y., New York.

[Bjørner, 2004] Bjørner, D. (2004). Train: The railway domain - a "grand challenge" for computing science & transportation engineering. In *IFIP Congress Topical Sessions*, pages 607–612.

[Bjørner, 2006b] Bjørner, D. (2005-2006b). *Software Engineering vol 1, 2, and 3*. Springer.

[Bjørner, 2006a] Bjørner, D. (2006a). On methods and software development.

[Booch et al., 1999] Booch, G., Rumbaugh, J., and Jacobson, I. (1999). *The Unified Modeling Language Reference Manual*. Object Technology Series. Addison-Wesley, USA.

[Borges et al., 2001] Borges, K. A. V., Jr., C. A. D., and alberto H. F. Laender (2001). Omt-g: An object-oriented data model for. *GeoInformatica*, 5(3):221–260.

[Brink et al., 1994] Brink, C., Britz, K., and Schmidt, R. A. (1994). Peirce algebras. *Formal Aspects of Computing*, 6(3):339–358.

[Brisaboa et al.,] Brisaboa, N., Mirbel, I., and Pernici, B. Constraints in spatio-temporal databases: A proposal of classification.

[Campos and Hornsby, 2004] Campos, J. and Hornsby, K. (2004). Temporal constraints between cyclic geographic events. In *Proceedings of GeoInfo 2004*.

[Casanova et al., 2002] Casanova, M., Straeten, R. V. D., and Wallet, T. (2002). Automatic Constraint Generation for Ensuring Quality of Geographic Data. In *3rd International Conference on Management Information Systems Incorporating GIS and Remote Sensing*, Halkidiki, Greece.

[Casanova et al., 2000] Casanova, M., Wallet, T., and D'Hondt, M. (2000). Ensuring Quality of Geographic Data with UML and OCL. In in *Proceedings of the 3rd International Conference on The Unified Modeling Language*, volume 1939 of *Lecture Notes in Computer Science*, pages 225–239. Springer.

[CE and W, 1949] CE, S. and W, W. (1949). The mathematical theory of communication.

[Chen, 1976] Chen, P. P. (1976). The entity-relationship model - toward a unified view of data. *TODS*, 1(1):9–36.

[Chrisman, 1984] Chrisman, N. R. (1984). The Role of Quality Information in the Long-Term Functioning of a Geographic Information System. *Cartographica*, 21(2&3):79–87.

[Christensen and Johnsen, 2005] Christensen, J. V. and Johnsen, M. (2005). Specifying formal constraints for geographic information.

[Clementini et al., 1993] Clementini, E., Felice, P. D., and van Oosterom, P. (1993). A small set of formal topological relationships for end-user interaction. In Abel, D. and Ooi, B. C., editors, *Advances in Spatial Databases - Third International Symposium SSD'93*, number LNCS 692 in Lecture Notes in Computer Science, pages 277–295. Springer-Verlag, Singapore.

[Cockburn, 2000] Cockburn, A. (2000). *Writing Effective Use Cases*. Addison Wesley.

[Cockcroft, 1997] Cockcroft, S. (1997). A taxonomy of spatial data integrity constraints. *GeoInformatica*, 1(4):327–343.

[D. Connolly et al., 2001] D. Connolly, F. v. H., I. Horrocks, D. L. M., Patel-Schneider, P. F., and Stein, L. A. (2001). Daml+oil (march 2001) reference description.

[Davids and Aquino, 2003] Davids, M. and Aquino, J. (2003). Java topology suite version 1.4. Technical report, Vivid Solutions.

[Davis et al., 1993] Davis, R., Shrobe, H., and Szolovits, P. (1993). What is a knowledge representation? *AI Magazine*, 14(1):17–33.

[Demuth et al., 2001] Demuth, B., Hussmann, H., and Loecher, S. (2001). OCL as a specification language for business rules in database applications. In Gogolla, M. and Kobryn, C., editors, *UML 2001 - The Unified Modeling Language. Modeling Languages, Concepts, and Tools. 4th International Conference, Toronto, Canada, October 2001, Proceedings*, volume 2185 of *LNCS*, pages 104–117. Springer.

[Drummond, 1995] Drummond, J. (1995). Positional Accuracy. In Guptill, S. C. and Morrsion, J. L., editors, *Elements of Spatial Data Quality*, chapter 3, pages 31–58. Elsevier, England. The International Cartographic Association.

[Egenhofer and Herring, 1990] Egenhofer, M. J. and Herring, J. R. (1990). Categorizing binary topological relations between regions, lines, and points in geographic databases. Technical report, University of Maine.

[Eir, 2004] Eir, A. (2004). *Construction Informatics - issues in the engineering, computer science, and ontology*. PhD thesis, DTU.

[Elmasri and Navathe, 1996] Elmasri and Navathe, editors (1996). *Fundamentals of database systems*. Addison Wesley.

[Elmasri et al., 1985] Elmasri, R., Weeldreyer, J., and Hevner, A. (1985). The category concept: An extension to the entityrelationship model. *Data & Knowledge Engineering*, 1:75–116.

[Engels et al., 1992] Engels, G., Gogolla, M., Hohenstein, U., Hulsmann, K., Lohr-Richter, P., Saake, G., and Ehrich, H.-D. (1992). Conceptual modelling of database applications using extended ER model. *Data Knowledge Engineering*, 9:157–204.

[ESRI, 2003] ESRI (2003). Arcgis: Working with geodatabase topology. White paper, ESRI.

[Fields, 1992] Fields, B. (1992). A guide to reading vdm specifications. Technical report, University of Manchester.

[Frank, 1998] Frank, A. (1998). Tiers of ontology and consistency constraints in geographic information systems.

[Friss-Christensen, 2003] Friss-Christensen, A. (2003). *Issues in the Conseptual Modeling of Geographic Data*. PhD thesis, Aalborg University, Department of Computer Science.

[Friss-Christensen and Christensen, 2004] Friss-Christensen, A. and Christensen, J. V. (2004). A framework for modeling spatial data quality. In *SDH-2004*, Liecester.

[Garshol, 2006] Garshol, L. M. (2006). Bnf and ebnf: What are they and how do they work?

[Garvin, 1988] Garvin, D. A. (1988). *Managing Quality: The Strategic and Competitive Edge*. Free Press.

[George et al., 1992] George, C., Haff, P., Havelund, K., Haxthausen, A. E., Milne, R., Nielsen, C. B., Prehn, S., and Wagner, K. R. (1992). *RAISE Specification Language*. Prentice Hall International.

[Gesbert, 2004] Gesbert, N. (2004). Formalisation of geographical database specifications. In *ADBIS (Local Proceedings)*.

[Goodchild, 1995] Goodchild, M. F. (1995). Attribute Accuracy. In Guptill, S. C. and Morrsion, J. L., editors, *Elements of Spatial Data Quality*, chapter 4, pages 59–80. Elsevier, England. The International Cartographic Association.

[Goodchild and Gopal, 1989] Goodchild, M. F. and Gopal, S., editors (1989). *The Accuracy of Spatial Databases*. Taylor & Francis.

[Goodwin, 2005] Goodwin, J. (2005). Experiences of using owl at the ordnance survey. *Proceedings of OWL: Expriences and Directions 2005*.

[Guptill and Morrison, 1995] Guptill, S. and Morrison, J., editors (1995). *Elements of Spatial Data Quality*. Elsevier, England. The International Cartographic Association.

[Herring, 1991] Herring, J. R. (1991). The matematical modeling of spatial and non-spatial information in geographic information systems. In Mark, D. M. and Franks, A. U., editors, *Congnitive and Linguistic Aspects of Geographic Space*, pages 313–350. Kluwer.

[Heuvelink, 1998] Heuvelink, G. B. M. (1998). *Error Propagation in Environmental Modelling in GIS*. Research Monographs in Geogrpahical Information Science. Taylor & Francis, UK.

[Hoel et al., 2003] Hoel, E., Menon, S., and Morehouse, S. (2003). Building a robust relational implementation of topology. In *Proceedings of the 8th International Symposium on Spatial and Temporal Databases*, Lecture Notes in Computer Science, Santorini. SSTD, Springer-Verlag.

[ISO, 1994] ISO (1994). Quality management and quality assurance - Vocabulary. Technical Report 8402, International Standardization Organization.

[ISO, 1996] ISO (1996). Information Technology - Syntactic Metalanguage - Extended BNF. Draft ISO/IEC 14977:1996, International Standardization Organization.

[ISO, 2001a] ISO (2001a). Geographic information - Quality evaluation procedures. ISO/TC 211 19114, International Standardization Organization.

[ISO, 2001b] ISO (2001b). Geographic information - Quality principles. ISO/TC 211 19113, International Standardization Organization.

[ISO, 2004] ISO (2004). Geographic information - rules for application schema. ISO/TC 211 19109, International Standardization Organization.

[Jackson, 1995] Jackson, M. (1995). *Software Requirementes & Specification - a lexicon og pratice, principles and prejudices*. Addison-Wesley.

[Jackson, 1997] Jackson, M. (1997). The meaning of requirements. *Ann. Software Eng.*, 3:5–21.

[Jackson, 2001] Jackson, M. (2001). *Problem Frames, Analysing and structuring software development problems*. Addison-Wesley.

[Johnsen, 2005] Johnsen, M. (2005). A high level database interfase with application to gis. Master's thesis, DTU.

[Kainz, 1995] Kainz, W. (1995). Logical Consistency. In Guptill, S. C. and Morrsion, J. L., editors, *Elements of Spatial Data Quality*, chapter 6, pages 109–138. Elsevier, England. The International Cartographic Association.

[KMS, 1999] KMS (1999). *Top10DK Specification*. National Survey and Cadastre, Copenhagen. Version 3.1.0. In Danish.

[Kolacny, 1969] Kolacny, A. (1969). *Cartographic Information - a Fundamental Concept in Term in Modern Cartography.*

[Lambrix, 2000] Lambrix, P. (2000). *Part-Whole Reasoning in an Object-Centered Framework.* Springer-Verlag New York, Inc., Secaucus, NJ, USA.

[Larman, 2001] Larman, C. (2001). *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and the Unified Process.* Prentice Hall.

[Molenaar, 1998] Molenaar, M. (1998). *An Introduction to the Theory of Spatial Object Modelling for GIS.* Taylor & Francis.

[Nilsson and Johnsen, 2007] Nilsson, J. F. and Johnsen, M. (2007). A high level logico-algebraic constraint checking language compiling into database queries. Forthcomming.

[Ohlbach, 2004] Ohlbach, H. J. (2004). Relations between fuzzy time intervals. In *Proceedings of 11th International Symposium on temporal representation and reasoning, Tatihoui, Normandie, France (1st–3rd July 2004)*. greyc.

[Parasuraman et al., 1985] Parasuraman, A., Zeithaml, V., and Berry, L. (1985). A conceptual model of service quality and its implications for future research. *Journal of Markedting*, 49:41–50.

[Parent, 2004] Parent, C. (2004). Mads formalization.

[Parent et al., 1999] Parent, C., Spaccapietra, S., and Zimányi, E. (1999). Spatio-temporal Conceptual Models: Data Structures + Space + Time. In in *Proceedings of the 7th International Symposium on Advances in Geographic Information Systems*, pages 26–33, Kansas City, USA.

[Parent et al., 1998] Parent, C., Spaccapietra, S., Zimányi, E., Donini, P., Plazanet, C., and Van-genot, C. (1998). Modeling Spatial Data in the MADS Conceptual Model. In in *Proceedings of the 8th International Symposium on Spatial Data Handling*, pages 138–150, Vancouver, Canada.

[Prieto-Diaz, 1990] Prieto-Diaz, R. (1990). Implementing faceted classification for software reuse (experience report). In *ICSE '90: Proceedings of the 12th international conference on Software engineering*, pages 300–304, Los Alamitos, CA, USA. IEEE Computer Society Press.

[Rolland and Achour, 1998] Rolland, C. and Achour, C. B. (1998). Guiding the construction of textual use case specifications. In *Data & Knowledge Engineering Journal*, volume 25. Elsevier Science Publishers.

[Ruschel et al., 2005] Ruschel, C., Iochpe, C., da Rocha, L. V., and Filho, J. L. (2005). Designing geographic analysis processes on the basis of the conceptual framework geoframe. In Chen, C.-S., Filipe, J., Seruca, I., and Cordeiro, J., editors, *ICEIS*, pages 91–97.

[S. et al., 2003] S., M., N., G., and D, S. (2003). A formal model for the specifications of geographic databases. In *Proceeding of GeoPro : semantic processing of Spatial data*.

[Shlaer and Mellor, 1992] Shlaer, S. and Mellor, S. (1992). *Object Lifecycles: Modeling the World in States*. Prentice Hall.

[Smith, 1996] Smith, B. (1996). Mereotopology: a theory of parts and boundaries. *Data Knowl. Eng.*, 20(3):287–303.

[Smith, 2004] Smith, B. (2004). The role of foundational relations in the alignment of biomedical ontologies. In *Proceedings of MEDINFO 2004*, pages 444–448.

[Smith et al., 2004] Smith, M. K., Welty, C., and McGuinness, D. L. (2004). The ontology web langauage - reference guide. Available online at: http://www.w3.org/TR/owl-guide.

[Sowa, 2000] Sowa, J. F. (2000). *Knowledge representation: logical, philosophical and computational foundations*. Brooks/Cole Publishing Co., Pacific Grove, CA, USA.

[Sutcliffe and Minocha, 1998] Sutcliffe, A. and Minocha, S. (1998). Scenario-based analysis of non-functional requirements. In *Fourth International Workshop on Requirements Engineering: Foundation for Software Quality (RESFQ)*.

[Tryfona and Jensen, 1999] Tryfona, N. and Jensen, C. S. (1999). Conceptual data modeling for spatiotemporal applications. *Geoinformatica*, 3(3):245–268.

[van Lamsweerde, 2001] van Lamsweerde, A. (2001). Goal-oriented requirements engineering: A guided tour. *RE'01 - 5th IEEE International Symposium on Requirements Engineering*, pages 249–263.

[van Lamsweerde et al., 1991] van Lamsweerde, A., Dardenne, A., Delcourt, B., and Dubisy, F. (1991). The kaos project: Knowledge acquisition in automated specification of software. *Proceedings AAAI Spring Symposium Series, Stanford University, American Association for Artificial Intelligence*, pages 59–62.

[Vangenot, 2004] Vangenot, C. (2004). Multi-represenation in spatial databases using the mads conceptual model. ICA Workshop on Generalisation and Multiple representation. Springer.

[Varzi, 1996] Varzi, A. C. (1996). Parts, wholes, and part-whole relations: the prospects of mereotopology. *Data Knowl. Eng.*, 20(3):259–286.

[Veregin, 1999] Veregin, H. (1999). Data Quality Parameters. In Longley, P. A., Goodchild, M. F., Maguire, D. J., and Rhind, D. W., editors, *Geographical Information Systems: Principles and Technical Issues*, volume 1, pages 177–189. John Wiley & Sons, Inc., USA, 2 edition.

[Veregin and Hargitai, 1995] Veregin, H. and Hargitai, P. (1995). An Evaluation Matrix for Geographical Data Quality. In Guptill, S. C. and Morrsion, J. L., editors, *Elements of Spatial Data Quality*, chapter 9, pages 167–188. Elsevier, England. The International Cartographic Association.

[von Neumann and Morgenstern, 1947] von Neumann, J. and Morgenstern, O. (1947). *Theory of games and economic behaviour*. Princeton University Press, Princeton. 2nd edition.

[Wagner, 2002] Wagner, G. (2002). How to design a general rule markup language? In Tolksdorf, R. and Eckstein, R., editors, *XSW*, volume 14 of *LNI*, pages 19–37. GI.

[Wang et al., 2001] Wang, R. Y., Ziad, M., and Lee, Y. W. (2001). *Data Quality*. The Kluwer International Series on Advances in Database Systems. Kluwer Academic Publishers, USA.

[Wielemak, 2007] Wielemak, J. (2007). Swi-prolog 5.6 reference manual. Technical report, University of Amsterdam.

[Wikipedia, 2005] Wikipedia (2005). Land-use. Internet. Last vistied June 2005.

[Wikipedia, 2006] Wikipedia (2006). Binary relation. In *Wikipedia*.

[Worboys, 1995] Worboys, M. (1995). *GIS: A Computing Perspective*. Taylor and Francis.

[Zeithaml et al., 1988] Zeithaml, V., Parasuraman, A., and Berry, L. (1988). *Delivering quality service: balancing customer perceptions and expectations*. Free Press, New York.