Developing error handling software for object-oriented geographical information

Matt Duckham
BSc Hons (Edinburgh)
MSc (Leicester)

A thesis submitted for the degree of Doctor of Philosophy

Department of Geography and Topographic Science
University of Glasgow

December 1999

I declare that this thesis is entirely the product of my own work, except where indicated, and has not been submitted by myself or any other person for any degree at this or any other university or college.

Matt Duckham

The time which we have at our disposal every day is elastic; the passions that we feel expand it, those that we inspire contract it; and habit fills up what remains.

Marcel Proust

Abstract

The inclusion of error handling capabilities within geographical information systems (GIS) is seen by many as crucial to the future commercial and legal stability of the technology. This thesis describes the analysis, design, implementation and use of a GIS able to handle both geographical information (GI) and the error associated with that GI. The first stage of this process is the development of an *error-sensitive* GIS, able to provide core error handling functionality in a form flexible enough to be widely applicable to error-prone GI. Object-oriented (OO) analysis, design and programming techniques, supported by recent developments in formal OO theory, are used to implement an error-sensitive GIS within Laser-Scan Gothic OOGIS software. The combination of formal theory and GIS software implementation suggests that error-sensitive GIS are a practical possibility using OO technology.

While the error-sensitive GIS is an important step toward full error handling systems, it is expected that most GIS users would require additional high level functionality before use of error-sensitive GIS could become commonplace. There is a clear need to provide error handling systems that actively assist non-expert users in assessing, using and understanding error in GI. To address this need, an *error-aware* GIS offering intelligent domain specific error handling software tools was developed, based on the core error-sensitive functionality. In order to provide a stable software bridge between the flexible error-sensitive GIS and specialised error-aware software tools, the error-aware GIS makes use of a distributed systems component architecture. The component architecture allows error-aware software tools that extend core error-sensitive functionality to be developed with minimal time and cost overheads.

Based on a telecommunications application in Kingston-upon-Hull, UK, three error-aware tools were developed to address particular needs identified within the application. First, an intelligent hypertext system in combination with a conventional expert system was used to assist GIS users with error-sensitive database design. Second, an inductive learning algorithm was used to automatically populate the error-sensitive database with information about error, based on a small pilot error assessment. Finally, a visualisation and data integration tool was developed to allow access to the error-sensitive database and error propagation routines to users across the Internet. While a number of important avenues of further work are implied by this research, the results of this research provide a blueprint for the development of practical error handling capabilities within GIS. The architecture used is both robust and flexible, and arguably represents a framework both for future research and for the development of commercial error handling GIS.

Preface and acknowledgements

This thesis represents a combination of both original and derivative material. Chapters 1–4 are derivative, chapter 5 is an original analysis of derivative material and chapters 6–10 are original.

This research was supported by NERC Award No. GT19/96/9/EO under a CASE agreement with Survey and Development Services (SDS).

This work owes much to the brilliant supervision of Dr Jane Drummond and Dr David Forrest for providing steering and guidance when needed and the space to get on with it otherwise. Likewise, John McCreadie and SDS were hugely supportive of the research and on two separate occasions generously donated funds for international conference expenses.

Many people in a number of companies gave up their valuable time to help out during the research. In particular, the help, interest and openness of Elspeth Rodger (SDS), Justin Hassal (Informed Solutions) and Paul Rickatson (Kingston Communications) is much appreciated. At Laser-Scan, Mike Jackson, Adrian Cuthbert, Daniel Ormsby and Tony Ibbs were all extremely helpful. Gothic software itself was generously supplied under a development license from Laser-Scan, while Laser-Scan also assisted with conference expenses for two Laser-Scan Users' Group Conference and one international conference. East of Scotland Water, and in particular Raymond Grieve and Gordon Burns, were very helpful during the start-up period of research.

In the Department of Geography and Topographic Science thanks to Dr Jo Sharp for many influential discussions and to Brian Black and Stephen McGinley for computer support, in addition to many others who have helped over the past 3 years: Prof John Briggs, Helen Boyd, Eilidh Douglas, Anne Dunlop, Dr Stella Lowder, Sally Gemmell, Amy McNeill, Dr Alistair Morrison, Prof Ronan Paddison, Prof Chris Philo, Dr Paul Routledge, John Shearer and Dr Mike Stewart to name but a few. Also, in sincere thanks and remembrance of Doug Ball who during his brief time at Glasgow managed to touch so many of us lucky enough to meet him. The interest and support of Dr Tom Melham, Dr Muffy Thomas and Dr Ron Poet in the Department of Computing Science at Glasgow University is gratefully acknowledged. Finally, thanks to Prof Pete Fisher, Dr Lucy Bastin and Dr Jo Wood at Leicester University for support and useful discussions.

This research has made every effort to take advantage of the wealth of free and open software available on the Internet. This would not have been possible without the work of innumerable companies and individuals involved in producing the highest quality software available. In particular, the work was completed under RedHat Linux 5.x using a range of open source software including GNU Emacs and XEmacs 20.x, JDK 1.1.7, TeX3.x and LaTeX2e, Apache 1.3.1 and Apache JServ 1.0, GNU gcc 2.7.x, JESS 5.0.a3, Ghostview 3.5.x, Gimp 1.0, Bibview 1.5, XFig 3.x and free software including Netscape 4.x, Star Office 5.0, VE Ovation 3.0 (free license), KDE 1.0, and BEA weblogic Java classes (free demonstration license). No commercial software was used in the completion of this research, except those supplied on free licenses such as Laser-Scan Gothic, VE Ovation and BEA weblogic.

On a more personal note, respect to my five-a-side pals for guarding my sanity: Ken Barrett, Paul McGeoch, Jos Fothergill, Ian McWhirter, Ruriadh MacKenzie and Oli Shipp. Respect and thanks also due to all the members of the Ugly Groove Movement, in particular Captain Funk (Paul Gilbody), Up-the-country-thumpin' (Dr Paul Taylor) and Doctor Double-Wah (Dr Simon Kirby), whose influence extends far beyond solid grooves and kickin' bass lines.

Finally, the endless patience, tireless support and love of all my family, especially my Mum and Dad, my sister Helen, and my brothers Ben and Julian, and of my babe, Helen Lamb, has been the solid ground upon which everything else is anchored.

Contents

List of Figures

List of Tables

Acronyms

AI	Artificial intelligence
API	Application programming interface
CEM	Classification error matri(x/ces)
CEN	European Committee on Standardisation
CLIPS	C language integrated production system
CMAS	Circular map accuracy standard
CORBA	Common object request broker architecture
DCOM	Distributed component object model
DEM	Digital elevation model(s)
EER	Extended entity-relationship
ER	Entity-relationship
ESRI	Environmental Systems Research Institute
FOPC	First order predicate calculus
GI	Geographic information
GIF	Graphics interchange format
GIS	Geographic information system(s)
GUI	Graphical user interface(s)
HTML	Hyper-text mark-up language
ICA	International Cartographic Association
IO	Input/output
IS	Information system(s)
ISO	International Organisation for Standardisation
JDBC	Java database connectivity
JESS	Java expert system shell
JFC	Java foundation classes
JVM	Java virtual machine(s)
KC	Kingston Communications
KR	Knowledge representation
NCDCDS	National Committee for Digital Cartographic Data Standards
NCGIA	National Center for Geographic Information and Analysis
NF	Normal form
NF1	First normal form
NLIS	National Land Information Service (UK)
NMAS	National map accuracy standard
OGC	Open GIS Consortium
OGM	Open geodata model
OMG	Object Management Group
OO	Object-oriented/object-orientation
OOA	Object-oriented analysis
OOAD	Object-oriented analysis and design
OOD	Object-oriented design
OODBMS	Object-oriented database management system(s)
OOP	Object-oriented programming
OOPL	Object-oriented programming language(s)
PC	Personal computer(s)

PCC	Percentage correctly classified
PDF	Probability density function(s)
RDBMS	Relational database management system(s)
RMI	Remote method invocation
RMSE	Root mean squared error
RPC	Remote procedure call
SAIF	Spatial archive and interchange format
ScotLIS	Scottish Land Information Service
SDS	Survey and Development Services
SDTS	Spatial data transfer standard
SPDFDM	Standard procedure and data format for digital mapping
TCP	Transmission control protocol
URL	Uniform resource locator(s)
UML	Unified modelling language
VM	Virtual machine(s)

Chapter 1

Introduction

Error is an inescapable feature of geographic information (GI) for three key reasons. First, the physical world is inherently indistinct and indeterminate. Situations where physical phenomena can be indisputably categorised, delineated or identified are exceptions rather than the rule (Russell 1912; Lakoff and Johnson 1980). Second, all information is constructed within the context of a theory (Raper and Livingston 1995). GI science aims to favour those theories most faithful to the physical world. However, the effects of the personal, cultural and institutional context of the production of information should never be discounted (Nyerges 1991; Campari and Frank 1993). Third, even the most careful observer using the most advanced equipment must introduce imprecision and inaccuracy into measurements of the physical world (Maffini et al. 1989). The endemic nature of error in GI is widely recognised (Blakemore 1984; Abler 1987; Fisher 1989; Brunsdon and Openshaw 1993) leading to the description of error as a "function of information" (Goodchild 1995) and a "fundamental dimension of data" (Chrisman 1991, p167).

In contrast to the acknowledgement of a fundamental relationship between error and GI, a number of authors have noted that commercial geographic information systems (GIS) have yielded little or no error handling capabilities to date (Thapa and Bossler 1992; Brunsdon and Openshaw 1993; Aspinall 1996). This omission raises serious questions about the commercial validity of GIS (Openshaw et al. 1991; Morrison 1995) and about the legal liability for erroneous information and decisions based on GIS technology (Epstien and Roitman 1990; Cho 1998; Epstien et al. 1998). Given that all GI is error-prone, there is a clear need to address the uncertainty surrounding information stored in and produced by GIS. For example, a number of authors have highlighted the relationship between detail and accuracy in GIS as problematic (Chrisman 1983; Goodchild and Gopal 1989; Mark and Csillag 1989). In conventional map production cartographers are able to adapt the feature detail to the accuracy of the map. Unfortunately the traditional approach has not survived the translation to GIS, where digital data is routinely stored to an arbitrary number of decimal places. As a result, GIS can bypass the cartographers' awareness of error, potentially leading to highly inappropriate use of GI. Remedial action to address the lack of support for error in GIS has long been accorded a high priority by influential bodies such as National Center for Geographic Information and Analysis (NCGIA, Abler 1987), the National Committee for Digital Cartographic Data Standards (NCDCDS 1988), the International Cartographic Association (ICA, Guptill and Morrison 1995), the Open GIS Consortium (OGC, 1999) and the International Standards Organisation (ISO, Godwin 1999), and by a variety of researchers (Burrough 1986; Goodchild and Gopal 1989; Merchant 1994; Chrisman 1997; Aalders 1999). The development of a GIS able to handle uncertain GI is likely, therefore, to have a profound effect upon the future research into and commercial application of GI technology.

1.1 Error and truth: definitions and concepts

Error is often defined as the discrepancy between reality and some observation or representation of reality. The definition of error implies the existence of a 'true' reality and knowledge about that 'truth' in the form of information. Information that is subject to error is said to be uncertain. However, it has already been established that error is an endemic, unavoidable feature of information, and it follows that the 'truth' is always, to some extent, unknown or unknowable. Nyerges (1991) and Lanter and Veregin (1992) point out that information is compiled to fulfill the requirements of a useful abstract representation of some aspect of reality, focusing on certain relevant characteristics whilst suppressing others . For example, most people would accept that the storage of a point in a GIS to represent the location of, say, a utilities inspection cover or telegraph pole, does not imply that, like the point, these features have zero dimensions. More generally, the widespread use of two-dimensional Cartesian grid coordinates to map features over small spatial extents is not taken to imply that the world, like the Cartesian plane, is flat. It is simply that these representations of reality are adequate as far as they adhere to an abstracted idea of reality.

From a more pragmatic standpoint, the realisation that the objective truth is often undesirable and always unobtainable can be problematic. For example, accuracy is an important component of error that reports how close an observation is to reality. A number of different methods have been proposed for the assessment of accuracy in spatial data. These range from deductive estimates, through internal evidence (comparison to some theoretical values or model), to comparison with an independent source of higher accuracy (NCDCDS, 1988). However, none of these methods would claim to compare an observation with the 'truth', merely to compare the observation with an *ideal* observation or some observation *taken to be true*. This mismatch between the definition and the practical assessment of accuracy, between truth and ideal, causes the definition of accuracy, and of error more generally, to take on an "awkward aspect of circularity" (Goodchild and Jeansoulin 1998, p211).

The aim of error handling in GI science is to resolve these contradictions through the provision of a context for GI that reflects the factors likely to contribute to error. The term *meta-data* is used to refer to the concept of contextual information or 'information about information', of which information about error and uncertainty is just a sub-set. Before error handling can be incorporated into GIS, it is necessary to formulate a framework for defining the meta-data of interest and mechanisms for manipulating and using meta-data. Thankfully, there already exists a well established paradigm for the handling of meta-data in GI science, which is the subject of the following section (§1.2). It is perhaps worth highlighting that throughout this work the terms *data* and *information* are used synonymously, in accordance with the conventional modern usage (Collins 1997; Merriam-Webster 1999).

1.2 Fitness for use

Within GI science the concept of *fitness for use* is central to the treatment of uncertainty. Information about the error associated with a particular data set or data model is termed *quality*. The assessment of fitness for use demands data producers supply enough information about the quality of a data set to allow a data user to come to a reasoned decision about the data's applicability to a particular situation or problem (Chrisman 1991). This approach apportions and emphasises the responsibilities between data provider and data user. The data provider has a responsibility to provide explicit, appropriate information on uncertainty along with data. The data user's responsibility is to ensure the data is only applied to problems where such use is apposite.

Both the question of what constitutes 'appropriate' information provision and the question of whether a particular use is 'apposite' necessarily involve an element of subjectivity. To some extent the rôle of standards organisations, examined later in §2.1.4, has been to provide guidance for data providers and users in answering these questions. Increasingly, legal considerations may play a part in determining the relative responsibilities of data provider and user (Epstien et al. 1998). However, it is certainly unrealistic to expect, as some authors have suggested (eg Agumya and Hunter 1997), that the fitness of a data set for a particular use can ever be assessed entirely objectively. Rather than a simple 'yes' or 'no', 'fit' or 'unfit', the question of fitness for use will almost always yield an answer qualified by a degree of subjectivity.

An important implication of fitness for use is that the reduction or elimination of error is a peripheral consideration. The desire to reduce error is predicated upon the view that error is in some way bad. However, an understanding of uncertainty is often correlated with an understanding of spatial processes, sampling and analysis (Burrough and McDonnell 1998). For example, Heuvelink (1998) describes a number of techniques which can be used to determine the contribution of various data sources to the uncertainty of an analysis that uses those sources. Heuvelink suggests this information can then be used to target where additional sampling accuracy is required and where extra sampling effort would have no significant effect upon the uncertainty of the analysis. Consequently, error is best conceptualised as a description of the characteristics and limitations of data, rather than as faults to be eradicated from the data (Chrisman 1982).

Fitness for use has been successful because it comprehends the diverse nature of GI science.

The various academic disciplines that comprise GI science have traditionally adopted different approaches to the management of error in information. Photogrammetrists, geodesists and surveyors have highly developed techniques for increasing precision and minimising error (Mikhail 1978; Bannister et al. 1992) and for mathematically modelling error (Chrisman 1982). Cartographers aim to control, represent and communicate to the user levels of uncertainty (Chrisman 1991). Computer science emphasises the enforcement of consistency within an information system (IS) using formal methods (Hunter 1996). Fitness for use accommodates all of these different approaches, providing a basis for GI technology which does not prejudice one approach over another.

1.3 Development of error handling in GIS

Having identified the lack of support for uncertain information as a problem within GI science and adopted fitness for use as a paradigm to redress this problem, it is possible to outline a GIS able to tackle uncertain GI.

1.3.1 Error-sensitive GIS

The term *error-sensitive* GIS was coined by Unwin (1995) to describe a GIS able to store and manage not simply GI, but the uncertainty associated with that information. A fundamental understanding of the features and behaviour of uncertainty and GI is a prerequisite to the development of error-sensitive GIS. Without such an understanding, the development process will tend to produce systems that are restricted in their usefulness by the discipline-dependent assumptions implicit in the design. Fitness for use is important in combating this tendency in two ways. First, an acceptance of the diverse, integrating nature of GI science should promote the incorporation of fewer, and more clearly stated assumptions within the development process. Second, a recognition of the rôle of data provider and data user in the management and use of uncertainty is beneficial for the system developer. Fitness for use encourages a clear delineation between those issues which are the responsibility of the system developer and those which are the responsibility of data provider and data user.

The goal of error-sensitive GIS development, then, is to produce an IS able to provide core storage and management functionality for uncertain GI in the same way as conventional GIS provide core spatial functionality for (certain) GI. It follows from the discussion above that flexibility is of paramount importance to error-sensitive GIS if it is to provide this core functionality, as is honouring the responsibilities data providers and users have to each other.

1.3.2 Error-aware GIS

The development of an error-sensitive GIS aims to provide the core functionality needed to store and manage uncertain GI. Paradoxically, this functionality is in itself of limited value. Fitness for use emphasises the importance of understanding and using information about uncertainty. Information on uncertainty is only useful if it can be incorporated into the geographic decision-making process. While there has been much research into uncertainty, relatively little goes as far as helping users understand and apply uncertain information in a practical way. The root cause of this paradox can be traced to the tension between the relative simplicity and determinism of IS when compared with the rich, complex nature of uncertainty in GI (Hunter 1996).

The development of an *error-aware* GIS aims to bridge the gap between an error-sensitive GIS and the understanding and practical use of uncertain GI. By extending the functionality of an error-sensitive GIS, an error-aware GIS should be able to offer high-level domain specific error handling functionality. The concept of an error-aware GIS entails the use of a variety of software development techniques. In particular, the use of artificial intelligence (AI) can assist in the representation of knowledge about uncertainty and the application of that knowledge in an intelligent manner. It is worth underlining the clear distinction that the use of domain specific and intelligent systems introduces between error-sensitive and error-aware GIS. An error-sensitive GIS is a highly flexible and highly generic system, but one that is unable to usefully be applied to practical decision making. An error-aware GIS is a domain specific extension of an error-sensitive GIS. The specificity of an error-aware GIS necessarily reduces its flexibility, but allows it to address the ultimate goal of understanding and using uncertainty in GI.

1.3.3 Object-orientation

Object-orientation (OO) is increasingly evident in the research and development of GIS. Many commercial GIS are now making significant use of OO, and OO is now widely used in research (eg Milne et al. 1993; Becker et al. 1996; Tang et al. 1996). While there is still a wide variety of relational databases and relational database development techniques used in commercial and research GIS applications, increasingly these can be viewed somewhat as legacy systems. The reason for these relatively recent changes is that OO represents an enormous efficiency of concept over other approaches to IS development. Using exclusively OO, the entire development process from problem definition, through system design, to system implementation can be tackled. Recent developments in OO theory are beginning to match the successes of OO techniques, technology and programming. The difficulties presented by the development of error-sensitive and error-aware GIS, therefore, are an interesting challenge for an OO approach. Object-oriented development techniques offer a practical mechanism for the implementation of error-sensitive and error-aware GIS. Object-oriented theory offers a powerful, rigorous base for understanding and exploring the concepts surrounding the technology.

1.4 Research aims

The aim of this research is to develop a generic model of uncertain GI consistent with the concept of fitness for use and to implement this model in the form of an object-oriented error-sensitive GIS. The research further aims to implement error-aware extensions to this error-sensitive GIS. The following four development aims for error handling in GIS can be identified (adapted from Heuvelink, 1998). The four aims were originally proposed as a blueprint for error propagation software (Heuvelink 1998), but have proved equally useful as a more general guide for the development of error handling in GIS.

- **Error handling should not replace existing GIS**

  Error handling within GIS must add to and augment, rather than replace, existing GIS functionality. Maximal reuse of existing models and software is seen as crucial to the uptake, integration and efficiency of error handling within GIS.

- **Error handling must be flexible**

  Error handling should not be tied to any particular set of operations or uses, data structure, or software platform. Minimising the assumptions made about the target application domain will result in flexible software that is suitable for a wide range of applications.

- **Error handling should be efficient**

  Computational and conceptual efficiency are prerequisites of effective IS. In particular, error handling is often computationally demanding in terms of both data processing and data volumes. Continual advances in computer hardware and software mean computational considerations are rarely of overriding concern, but in combination with the need for simplicity and conceptual efficiency, can never be ignored.

- **Error handling in GIS should allow exploratory user interfacing**

  As already noted in §1.3.2, the provision of a user interface able to assist non-specialist users with assessing and understanding fitness for use is a fundamental and often overlooked component of error handling functionality.

1.4.1 Utilities application of error-aware GIS

It is important to reiterate that in large part the reasons for developing an error-sensitive GIS are highly pragmatic. To be successful, an error-sensitive GIS must be able to provide some practical advantage to users, over conventional GIS. While the aim of the research is the development rather than the application of error-sensitive GIS, the research would not be complete without using the systems developed in an example application. The application chosen here is that of a telecommunications network in Kingston-upon-Hull, UK. Utility applications generally are recognised as one of the most important commercial uses of GIS (Russomanno 1998), and telecommunications is one of the most important utility applications. The final stages of this thesis aim to show how the combination of error-sensitive and error-aware GIS can address the error handling needs of this rapidly expanding application area.

1.5 Thesis structure

The thesis structure broadly mirrors the progress of this research. Chapter 2 provides a background to research into error in GI and to current use of data quality as a mechanism for representing and reporting uncertainty. Chapter 3 explores the use of object-oriented analysis (OOA), object-oriented design (OOD) and OO generally. In particular it introduces a number of recent advances in object theory which are used in subsequent chapters. These two research strands of data quality and OO are combined in chapter 4, where the theory used for the development of an error-sensitive GIS is set out. Chapter 5 details the actual implementation of an error-sensitive GIS based on this theory. Chapters 6 looks at the architecture of an error-aware GIS and introduces the technology needed to link error-aware and error-sensitive GIS. Chapters 7, 8 and 9 describe the implementation of three error-aware software tools and explore their use within a telecommunications application. Finally, chapter 10 concludes this thesis and provides an overview of future research based on the outcome of this research.

Five brief appendices are also included at the end of this thesis. Appendix A contains a number of object systems referred to in chapters 3, 4 and 5. Appendix B contains the core of an expert system design referred to in chapter 7. Appendix C comprises a CD-ROM and a page listing the CD-ROM contents. The CD-ROM contains both code and documentation for the systems developed during this research. Appendix D contains a brief *curriculum vitae* current at the time of writing. Appendix E contains an additional paper, accepted for publication in the International Journal of Geographical Information Science, which does not form part of this thesis itself, but covers related research in support of this doctoral work.

Chapter 2

Data quality and models of error

In order to handle the uncertainty associated with GI, it is first necessary to develop a framework for discourse about error. The term *data quality* is used to refer generally to this structure imposed on the concept of uncertainty. The use of data quality is implied by the definition of fitness for use. Data providers can only document the uncertainty associated with their data and data users can only determine the suitability of data for a particular use if there is some common arena of discourse between the two. For example, intuitively the accuracy of data is an important element of uncertainty. Data users might be less confident in decisions based on highly inaccurate data than decisions based on highly accurate data and so expect data providers to document the accuracy of their data. However, even this relatively simple example quickly becomes fraught with difficulties: against what standard do data providers determine the accuracy of data? how can data users compare the accuracy of different data? and is it important for data providers and users to distinguish between the accuracy of *where* something is and *what* something is?

The study of data quality is central to the success of an error-sensitive GIS. At its core, any error-sensitive GIS is a formal model of error. An understanding of different approaches to data quality is the foundation of such a formal model. There is an enormously rich variety of structures which have already been proposed to describe the uncertainty associated with GI. The aim of this chapter is to explore these various approaches to data quality under three broad headings. The following section looks at the rôle of standards organisations and the use of data quality standards while §2.2 explores the representation of data quality within the research literature. Finally, §2.3 examines an approach to data modelling which can be used to examine the phenomena of data quality. The approach introduced in §2.3 is important in that it arguably encompasses the more specific data quality standards and research error models. Consequently it is a suitable candidate for use in the development of an error-sensitive GIS.

2.1 Data quality standards

A variety of organisations have worked on the production of data quality standards over recent years. As already mentioned, the ICA have been involved in attempts to standardise data quality (Guptill and Morrison 1995). In Europe, the European Committee on Standardisation's (CEN) draft standard for geographic information includes a data quality standard (CEN/TC287 1996). Following on from the CEN work, an ISO working group are currently developing an international standard for spatial data quality (Godwin 1999; ISO/TC211 1999). A recent authoritative study of the world's national spatial data transfer standards identifies 22 separate standards all of which mention the importance of data quality and almost all make significant attempts to standardise data quality (Moellering 1997). However, closer inspection reveals that, with a few notable exceptions, the majority of national data quality standards are closely related to the United States' Spatial Data Transfer Standard (SDTS). Originally developed by the National Committee for Digital Cartographic Data Standards (1988), SDTS defines five *elements* of data quality: lineage, positional accuracy, attribute accuracy, logical consistency and completeness. These five data quality elements have gained widespread acceptance amongst standard organisations, geoinformation researchers and professionals across the world (Morrison 1995) and have dominated the horizons of data quality for a decade, earning them the affectionate appellative 'the Famous Five'.

2.1.1 SDTS 'Famous Five'

The 1988 NCDCDS proposal which developed into the SDTS was designed with the concept of *truth in labelling* at its core. In common with fitness for use, truth in labelling rejects the formulation of prescriptive and arbitrary thresholds of quality in favour of the provision of detailed information on data quality. The five quality elements used for this purpose are as follows.

- **Lineage** is a complete description of the source material, processes, operations and transformations performed upon the data, including dates. SDTS leaves the detailed structure of lineage largely unspecified.

- **Positional accuracy** is a measure of the closeness of positional data to the 'true' (ideal) value.

- **Attribute accuracy** is a measure of the closeness of thematic data to the 'true' (ideal) value. The SDTS quality element attribute accuracy is composed of two distinct components. Categorical attribute accuracy describes the accuracy of categorical data, while continuous attribute accuracy is associated with quantitative data and has more in common with positional accuracy than categorical attribute accuracy.

- **Logical consistency** is the "fidelity of relationships encoded in the data structure" (US Geological Survey 1999b). Generally, this involves reporting the results of logical tests of validity upon the data set.

- **Completeness** is a description of the relationship between the features represented in the data and the set of all possible features. Usually, completeness comprises a description of the exhaustiveness of the data set.

It is difficult to understate the importance of the SDTS five data quality elements when discussing data quality standards. The influence of the 'Famous Five' is still clearly discernible even in transfer standards, such as the CEN draft standard (CEN/TC287 1996) and the Canadian Spatial Archive and Interchange Format (SAIF), that have developed approaches to spatial data transfer distinctively different to SDTS (see Geographic Data BC 1996). Despite the success of the 'Famous Five', it is arguable that neither SDTS nor any other standard provides a suitable basis for the implementation of error handling capabilities within GIS. The following sections (§2.1.2–§2.1.4) look at some of the arguments against using data quality standards in the design of GIS.

2.1.2 Exhaustiveness and expressiveness

Even a cursory examination of the literature suggests that the five SDTS elements of spatial data quality are not exhaustive. For example, the ICA Commission on Spatial Data Quality accepts the SDTS data quality elements, yet feels the need to augment these with semantic and temporal accuracy. In 'Elements of spatial data quality', Morrison (1995) points out that it was "more important to place solid definitions of the *seven* components in the literature than to attempt to be totally complete at this time" [emphasis added]. A variety of national data transfer standards, while usually supporting at least some of the core SDTS quality elements, propose a smattering of additional quality elements. The Japanese Standard Procedure and Data Format for Digital Mapping (SPDFDM) regards "map information level" as an important data quality element in its own right. "Reliability" and "cartographic identifiability" are included within the Netherlands' standard (NEN1878) whilst SAIF suggests the use of a raft of additional meta-data and quality elements (Moellering 1997). Similarly, a variety of researchers, whilst accepting the core SDTS quality elements, have suggested the use of additional data quality elements, such as detail (Goodchild and Proctor 1997), textual fidelity (CEN/TC287 1996), source and usage (Aalders 1996).

Further, Drummond (1996) notes that there is no firm agreement on the actual definitions of many of the elements of data quality which occur in standards and more generally in the literature. Standards are by their very nature both prescriptive and proscriptive. This leads to an inflexibility in standards that limits their expressive range. Therefore, it seems reasonable to suggest that no single data quality standard is likely to be exhaustive or expressive enough to suit every data user's needs. It follows that no standard classification of uncertainty will ever be universally accepted (Hunter 1996). The assertion that it is "widely accepted that data quality is described by five elements" (Agumya and Hunter 1997, p112) is, in fact, hotly disputed.

2.1.3 Responsibility

The importance of responsibility in fitness for use has already been emphasised (see §1.2). Under fitness for use, data providers have a responsibility to provide detailed, appropriate data quality information. It is arguable that in many cases, compliance with a data quality standard represents the most appropriate data quality information. Certainly, an error-sensitive GIS needs to be able to support the use of any data quality standard. However, it does not follow that an error-sensitive GIS should support *only* data quality standards. Given that no data quality standard can ever be exhaustive or expressive enough for every eventuality, effectively "hard-wiring" data quality standards into error-sensitive GIS design is a derogation of the responsibilities of the data provider to the data user. To illustrate, the development of conventional GIS might have been very different had spatial database design restricted GIS users to representing a standard set of geospatial objects. Similarly, restricting error-sensitive GIS users to only those quality elements based on data quality standards inhibits the utility of the system for those users.

This approach is not without hazard. Providing a GIS tool that is able to handle uncertainty in a highly flexible way, not tied to approved data quality standards, admits the possibility that producers and users will employ data quality in an entirely inappropriate manner. Nevertheless, data providers are much more likely to understand the peculiarities of their data than standards organisations. Enabling data providers to freely express this understanding is seen as a determinative factor in the uptake, growth and success of an error-sensitive GIS.

2.1.4 The rôle of data quality standards

The intention is not to downplay the importance of the five SDTS quality elements or data quality standards in general. The extensive use of SDTS in one form or another within many national and international standards organisations is an indication of its value. However, such quality standards should not form the basis of a GI science approach to data quality nor the development of an error-sensitive GIS. Though scarce, previous attempts to implement error handling within GIS have tended to be standards led (eg Guptill 1989; Ramlal and Drummond 1992; van der Wel et al. 1994). Data quality standards are designed to fulfill a specific need to disseminate expert advice, to provide guidance and a common lexicon for data quality. While an error-sensitive GIS must provide support for data quality standards, for reasons of exhaustiveness, expressiveness and responsibility they are not a suitable basis for research into error-sensitive GIS.

2.2 Research error models

There is a formidable volume of research concerned with error. A striking characteristic of this body of work is the diversity and lack of integration between the different research threads. Due to the complex nature of error in GI the current disaggregation is, perhaps, only to be expected and the majority of research work is, understandably, restricted to fairly specific application domains. Nevertheless, error handling within GIS depends upon the development of a coherent conceptual and computational framework for error. An understanding of the existing models of error is, therefore, an important component of this development.

Given that there is already a surfeit of literature that provides a variety of different classifications of existing error models (eg Veregin 1989; Openshaw et al. 1991; Lanter and Veregin 1992; Forier and Canters 1996), a slightly different approach is taken here. This section aims to review the range of error models used in research by charting the evolution of the different research strands, in particular highlighting the different representations of error used.

First, §2.2.1 looks at the development of error models that attempt to represent locational accuracy. These models make an (often somewhat arbitrary) distinction between the error in identifying where a phenomenon is and what it is. Locational-thematic error models, discussed in §2.2.2, make no such distinction and attempt to represent the accuracy of spatially distributed observations. A starkly different approach to error is to classify different types of error, and a variety of models that aim to produce error taxonomies are explored in §2.2.3. The central aim of this section is to indicate the range and relative merits of different approaches and to imply that no one model stands out above all the others. The idea that all the different models of error are all valid in their own right is explored in §2.3 alongside a conceptual model of IS that is, arguably, inclusive of all these different models.

2.2.1 Locational error models

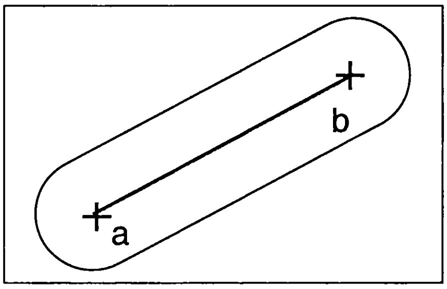


Figure 2.1: ε-band (after Chrisman 1983)

Locational error models attempt to represent the accuracy of locational information. One of the simplest locational error models is embodied by the United States' *national map accuracy standard* (NMAS), also termed the *circular map accuracy standard* (CMAS). CMAS states that for a given map or data set, a certain percentage of well defined points must lie within a set radius of their 'true' ground location (Keefer et al. 1988). While CMAS can be reported for a single point, the standard is more usually presented as a statistical composite that provides an uncertainty threshold for an entire map sheet or data set (Lundin et al. 1990). In many ways, the line-based analogue of CMAS is the ε (epsilon) band, proposed by Blakemore (1984). Originally developed from an automated cartographic generalisation technique (Perkal 1966 in Chrisman 1989), the ε-band is simply a buffer distance used to denote a zone of uncertainty around a line or polygon boundary. An ε-band for a simple line segment between two points, *a* and *b*, is shown in figure 2.1. Both CMAS and ε-bands are simple and efficient models of error that may be adequate for some applications. However, they fall well short of a comprehensive model of error for most application domains (Goodchild et al. 1992) and it is not surprising that a number of more sophisticated models have also been proposed.

Generally, most more sophisticated locational error models are derived from standard statistical models of variation, such as *central limit theorem* which dates back to Gauss' foundational work on statistics at the beginning of the 19th century. Under central limit theorem the random variation inherent in a set of observations is expected to be normally distributed over a large enough number of repeated observations (Hugill 1988). For scalar measurements, the mean of this distribution is an estimate of its 'true' value. Instead of reporting the percentage of points of a certain accuracy for a data set, the standard deviation and the root mean squared error (RMSE) can be used to communicate the precision and accuracy of a point location respectively (Drummond 1995). Such a model is highly robust and allows the production of derived measures of error, such as CMAS.

By associating RMSE values with the vertices of a vectorised line or polygon it is possible to build up a model of line and polygon error. The vertex-based model of line and polygon error is now well developed and widely used. The statistical properties of the vertices, in terms of the mean and standard deviations of each vertex in the x and y directions, can be used to construct complex models of positional uncertainty (Ehlers and Shi 1996; Shi 1998). Figure 2.2 illustrates the probability distribution for a simple line segment between two points, a and b, using Ehlers and Shi's 1996 model. The model superimposes a Gaussian probability distribution ($\phi(z)$) upon the vertex locations and derives the probability distribution of the line from the distributions of the two end points. The vertex-based error model can be used to track the propagation of locational error through a variety of spatial operations. Variance propagation can be used to calculate the error associated with derived spatial information (Drummond 1995), although this approach is only practical provided the derived information is a analytical function of the spatial information (Heuvelink 1998), such as length, angle and area. When the error associated with more complex, non-linear spatial operations is required, Monte-Carlo simulation can be used as a flexible non-deterministic error-propagation tool (eg Openshaw et al. 1991; Goodchild et al. 1999).

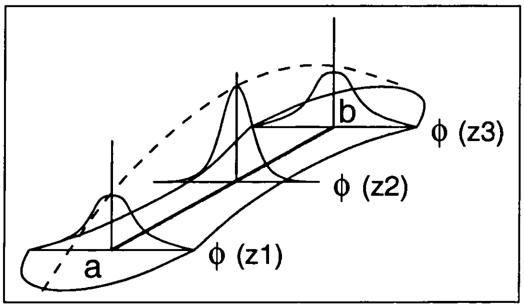


Figure 2.2: Vertex-based error band (Ehlers and Shi, 1996)

Despite the success of vertex-based locational error models, they are open to serious criticism. Since solely errors in vertex location and not line location are represented, the model leads to the counter-intuitive conclusion that accuracy is lowest at the sampled vertices and higher between vertex pairs (Chrisman 1989; Goodchild 1989). The effect is illustrated in figure 2.2, which emphasises the elevated accuracy away from the two vertices. In the same way that the ε-band is the line-based analogue of the CMAS, it is possible to construct a line-based model of locational error, analogous to the vertex-based model, which does not suffer from the problems of excessive accuracy. The use of probabilistic ε-bands, where ε distance determines the width of some cumulative probability function representing line uncertainty, has been championed by a number of authors (Chrisman 1982; Chrisman 1989; Mark and Csillag 1989). Until recently, the approach has not been particularly successful as it lacked a clear theoretical relation between the variability in point locations and the variability in line locations and was largely a conceptual, rather than empirical, model (Goodchild and Dubuc 1986; Leung and Yan 1998). However, Leung and Yan (1998) have proposed a point-based error model not biased towards vertices which seems to able to provide some of the conceptual advantages of the ε-band coupled with the stochastic power of vertex-based approaches. Figure 2.3 gives a schematic of the model, highlighting that while individual points on a line are normally distributed, the accuracy of the line is constant across its entire length. The approach, discussed in more detail in §9.3.1 allows the representation of uncertainty in point, line and polygonal features using a single index of accuracy and can be used to describe locational accuracy at any level of feature aggregation from individual geometries to entire databases. A clear drawback of the approach, however, is that it does not admit the possibility of varying accuracy over the length of the line.

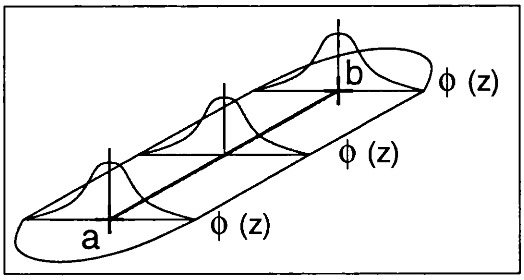


Figure 2.3: Point-based error band

While vertex-, point- and line-based locational error models have tended to dominate the literature, it would be incorrect to portray such approaches as the only locational error models. For example, Kiiveri (1997) has proposed a relatively complex model that uses rubber-sheet distortions to represent locational error. In the model, distortion is a function of two parameters, related to the maximum displacement in the x and y directions. Unlike vertex-, point- or line-based locational error models, the use of rubber-sheet distortions ensures topology is preserved and is compatible with raster as well as vector data models. This advantage is, however, attained at the cost of flexibility. The distortion-based location model is only practicable as a global error model for entire coverages and not for individual geometric features.

2.2.2 Locational-thematic error models

Locational error models make a distinction between the value and the location of observations. It is possible to distinguish between failures to assign the correct value to an observation and failures to correctly locate an observation, termed *identification errors* and *discrimination errors* respectively (Chrisman 1987). For many observations, this is a sensible course of action. When surveying a road, for example, the uncertainty associated with the observation of the road's location can be viewed as independent of the uncertainty associated with whether the road is in fact a track or a trunk road. However, in many cases location and identity and inextricably linked. Within certain bounds the effects of discrimination and identification errors in a digital elevation model (DEM), for example, may be identical. Locational-thematic error models represent the accuracy of spatially located thematic information where the distinction between locational and thematic begins to break down. The dichotomy between locational and locational-thematic error models can, of course, be traced back to the dichotomy between the underlying object and field data models respectively (Goodchild 1989). As a result, the remainder of this section relates to field-based data model errors as opposed to the predominately object-based data model errors of the previous section.

The simplest models of locational-thematic error are based on a matrix of classification errors, known as a mis-classification matrix or classification error matrix (CEM). A CEM cross tabulates the classification of cells in a raster data set with those in some ideal, verification data set (Rosenfield 1986). An example CEM for three land-use classifications, forest, water and urban areas, is shown in table 2.1. The CEM can be used to generate a variety of statistics on uncertainty. The percentage of correctly classified areas (PCC) has been used to represent the probability of finding a correctly classified qualitative value at a given location (eg Newcomer and Szajgin 1984). PCC is calculated by dividing the sum of the correctly classified cells by the total number of cells (in the case of the example in table 2.1, 63÷100=63%). It has been argued that the kappa statistic is a better representation of error than PCC since the latter does not account for values that are correct purely through random chance and hence tends to over-estimate accuracy (Veregin 1995). However, such simple statistics ignore the mis-classified cells and can give a misleading impression of accuracy (Rosenfield 1986; Story and Congalton 1986). The terms *producer's accuracy* or *errors of omission* are used to describe for a class C the number of correctly classified cells as a proportion of the total number of cells that should have been classified as C. In contrast, *user's accuracy* or *errors of commission* describe for each class C the number of correctly classified cells as a proportion of the total number of cells that were classified as C. Alongside the CEM in table 2.1 the producer's and user's accuracy have been calculated for each land-use class using these definitions. Work by Veregin (1995) is, perhaps, the logical conclusion of this evolution of increasingly detailed CEM-based location-thematic error statistics. Instead of creating statistical composites of the CEM, Veregin's method models the entire CEM in the form of sub-matrices of the original CEM for each class identified in the original CEM (Veregin 1995).

The foremost failing of CEM-based location-thematic error models is that they do not explicitly deal with spatial variation in error. While easy to understand and construct, for example as a by-product of the classification of remotely sensed images, the CEM implies that uncertainty is uniformly distributed across the area of interest. In an attempt to redress this failing, Goodchild

Classification	**Verification data**				**Producer's**	**User's**
data	Forest	Water	Urban		**accuracy**	**accuracy**
Forest	28	14	15	Forest	28/30 = 93%	28/57 = 49%
Water	1	15	5	Water	15/30 = 50%	15/21 = 71%
Urban	1	1	20	Urban	20/40 = 50%	20/22 = 91%

Table 2.1: Example CEM and derived accuracy indices (from Story and Congalton, 1986)

et al. (1992) and Veregin (1996) use an explicitly spatial model of locational-thematic categorical error for raster data sets, based on two components. First a vector of probabilities is required for each classified cell describing the likely misclassification for that cell. Second a spatial correlation coefficient is used to describe the degree of interdependence between adjacent cells. While the approach provides a much more detailed, spatial model of locational-thematic error, it is interesting to note that a GIS implementing the model might be expected to routinely store an order of magnitude more meta-data than actual geospatial data.

Related to the probability surface approach, a string of authors have developed the use of fuzzy sets to model locational-thematic error, in particular the error associated with locational-thematic boundary imprecision (Leung 1987; Altman 1994; Wang and Hull 1996; Davis and Keller 1997). Fuzzy locational-thematic error models require a vector of fuzzy membership values to be associated with each classified cell in a data set. A vector of values describes the degree to which each classified cell is a member of the different possible classifications. In practice, fuzzy locational-thematic error models are very similar to the probabilistic models of Goodchild et al. (1992) and Veregin (1996) although the two approaches are semantically and theoretically distinct. The former uses the probability that a cell belongs to a particular class, while the latter uses the more general possibility that a cell belongs to a particular class to represent error (Veregin 1996). In fact, there is a wide range of set theoretic approaches that might conceivably be used to model locational-thematic error, of which classical probability and fuzzy sets are just two. The use of non-monotonic logic, Dempster-Shafer theory of evidence and rough sets have all shown themselves capable of addressing particular aspects of locational-thematic error not fully covered by other approaches (Stoms 1987; Worboys 1998).

However, broadly speaking, such theories can only be used to address the uncertainty surrounding the categorisation of spatial data. For many locational-thematic data sets, the observations are not categorical but quantitative. Quantitative locational-thematic data arguably has more in common with locational than qualitative locational-thematic data. Consequently, in common with most approaches to locational error, quantitative locational-thematic error is usually represented using standard Gaussian models of natural variation. A decade of work by Gerard Heuvelink has yielded an extremely detailed model of quantitative locational-thematic error, based on stochastic models of error (Heuvelink et al. 1989; Heuvelink 1993; Heuvelink 1998). For the purposes of error handling in GIS, such models require the storage of a range of statistical indices based on the mean, standard deviation, correlation and autocorrelation of spatially located observations (Heuvelink 1998). Further, Heuvelink's work emphasises the importance of recording and retaining the original observations and uncertainties upon which a data set is based, throughout the life of the data set. Such requirements clearly have implications for the development of error-sensitive GIS.

2.2.3 Error taxonomy

In contrast to the more formal approaches to error above, a number of taxonomic approaches to error have been proposed. Burrough (1986) produced a classification of errors based on the source of the error. Group I errors are 'obvious sources' of error such as map scale and temporal factors. Group II errors result from the natural variation within the phenomena being measured, while Group III errors occur as a result of processing data. The approach has undoubtedly been influential in a number of later studies. Veregin (1989) produced a hierarchy of needs for treatment of error, the most basic three levels echoing Burrough's taxonomy. Elmes et al. (1994) produced a system capable of automatically estimating of the accuracy of spatial data based on a development of Burrough's taxonomy. Clearly related to Burrough (1986), Maffini et al. (1989) identified three different sources of error; the inherent properties of nature, the nature of measurement and the data models used.

A subtly different approach is also concerned with the source of errors, but only from the point of view of model and process errors. Goodchild (1989) distinguishes between 'source' and 'processing' errors, the former relating to the error resulting from differences between ideal and actual observations, the latter relating to errors introduced through processing of the data. Emmi and Horton (1995) use a similar distinction between measurement and processing error in an assessment of seismic risk. While the same themes often recur there is seemingly no limit to the number of equally plausible taxonomies of error. Burrough himself dropped the three group taxonomy from the recently revised "Principles of GIS" in favour of a more capacious seven point description of factors affecting spatial data quality, comprising currency, completeness, consistency, accessibility, accuracy and precision, sources of error in data and sources of error in derived data models and analysis (Burrough and McDonnell 1998). Goodchild (1995) champions the importance of differentiation between absolute and relative accuracy and identifies errors in measurement and definition, lack of documentation, interpretation, processing errors and physical distortions as the as the important sources of error. While many of the different error taxonomies may not even claim to be comprehensive, systematic, fundamental or generic they are all to a greater or lesser degree useful representations of error and uncertainty.

2.2.4 The rôle of research error models

This section has attempted to clarify the bewildering array of contrasting models of error used in GI science. The models described range from using single indices of error that apply to an entire data set, to vectors of values that apply to individual cells or observations in a data set. Some approaches use clearly defined statistical measures, such as RMSE, whilst others suggest relatively vague, unstructured meta-data, such as Burrough's (1986) three error groups. Certain data quality parameters, such as standard deviation, potentially form the basis of a wide range of models of error, whilst others, such as the parameters in Kiiveri's 1997 rubber-sheet error model, are highly specific to one particular model of error. However, no single model stands out as the most appropriate choice to represent the entire spectrum of errors. Error in GI is "inherently multi-dimensional" (Lanter and Veregin 1992, p825) and deciding which model of error is most appropriate for a given data set involves weighing up the relative merits of simple or complex, generic or specific, clearly defined or vague approaches.

Thus, the development of error-sensitive GIS is left with two alternatives. It would be possible to adopt one or more existing research models of error as fundamental to error-sensitive GIS, in the knowledge that such a system is likely to be incompatible with some existing error models and with possible future advancements. Clearly preferable, however, is to attempt to isolate GIS design from the volatile detail of error models. The second alternative, then, is to look for another framework for error-sensitive GIS development that is not dependent on a particular research model. The following section explores the only remaining candidate for this purpose.

2.3 A conceptual model of information systems

The previous sections of this chapter have identified a diverse range of representations of error, embedded in the data quality portion of spatial data transfer standards and in research error models. Paradoxically, both data quality standards and research models of error have been rejected as suitable frameworks for the development of error-sensitive GIS (§2.1.4 and §2.2.4 respectively) since neither can claim to be fundamental or generic. Consequently, a more general approach to error and uncertainty is needed: one able to support the entire range of error models without being tied to any particular data quality standard or research thread.

Such an approach does exist within the literature. The conventional conceptual model of IS is illustrated in figure 2.4 (see Maguire and Dangermond 1991; Worboys 1992; Kainz 1995; David et al. 1996). The conceptual model in figure 2.4 is implicitly realist in epistemology: it is based on the existence of a 'real world' containing observable phenomena[1]. The assumption made in figure 2.4, as in §1.1, is that the real world is infinitely complex and to some extent always unknowable. Consequently, reality is handled following the abstraction of the real world to an ideal, "practically adequate" (Nyerges 1991, p1485) data set, termed the *terrain nominal.* An actual data set or database constitutes a representation of this terrain nominal.

[1] The 'real world' may also contain unobservable phenomena, but the conceptual model adopted here only attempts to deal with observable phenomena.

From the point of view of physical, geographic features this information flow works well. For example, a utility company might need to record the locations of inspection covers within a GIS as part of a wider management strategy. The first step in the process is to form some practically adequate concept of the features of the real world pertinent to the application, namely that inspection covers are of interest and that their locations, perhaps represented as points in a Cartesian grid, and a limited set of their attributes will be recorded in the GIS. This abstraction forms the terrain nominal. Individual observations of inspection covers are representations of the terrain nominal, which is in turn an abstraction of reality.

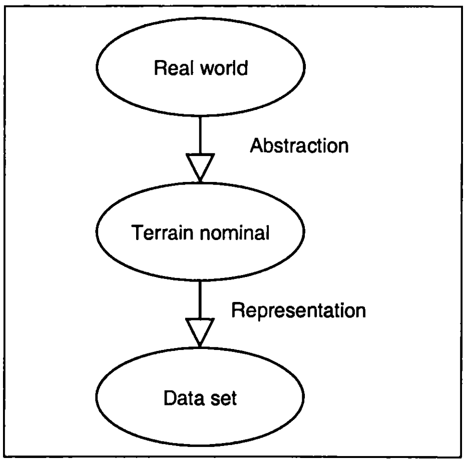


Figure 2.4: Conceptual model of IS

Crucially, in contrast to geospatial objects, there is no meaningful concept of data quality in the real world. A key concept is that data quality is not usefully modelled by the processes of abstraction and representation, rather it is only as a side effect of deficiencies in these processes that data quality arises at all (David et al. 1996). In the original definition of error (§1.1) the gap between reality and information was bridged by the realisation that information is compiled to fulfill the requirements of a useful abstraction of reality. This definition relates directly to the conceptual model of IS in figure 2.4. Error is the product of both deficiencies in the representation of our abstract view of the world and of deficiencies in the production of that abstract view. As a result of its parsimony with the definition of error, it is argued here that the conceptual model of IS represents a fundamental approach to error in information. The approach underlies the various different models of error proposed by standards organisations and researchers and consequently is preferable to any of these models as the basis for the development of an error-sensitive GIS.

It is worth noting that the use of the conceptual model in figure 2.4 makes the simplifying assumption that *all* geographic information refers directly to real world geographic objects. In fact, current data capture methods may often mean that many digital databases refer indirectly to real world objects via paper maps, themselves real world objects. A more complete conceptual model would additionally need to include such indirect information sources, where a database refers to a real world object or artifact such as a paper map which itself refers to geographic objects in the real world. However, for conceptual simplicity and because the current reliance on paper mapping for digital data capture should only be transitory the conceptual model in figure 2.4 seems reasonable.

2.3.1 Concluding remarks

The challenge to an error-sensitive GIS is clear. There is a rich diversity of existing models of error, embedded within both standards organisations and current research. There is every reason to believe that these models will continue to be amended, improved and extended in the future. In order to meet the aim of flexibility set out in §1.4, an error-sensitive GIS must be able to support the entire range of possible models of error. Further, while standards and research error models provide an important resource base for data providers and data users, there is evidence that an error-sensitive GIS should be flexible enough to allow users to adapt or define models of error for their own purposes.

The framework for error-sensitive GIS development is also clear. No single error model has enough expressive power or flexibility to represent all the others. It follows that any error-sensitive GIS tied to a particular data quality standard or research error model is unlikely to be able to respond to the full range of users' requirements. The conceptual model of IS presented in §2.3 is the only approach that can claim to be fundamental to error models generally. Therefore, it is the preferred basis for error-sensitive GIS design. Having identified the challenges and a framework to address those challenges, it only remains to identify a mechanism capable of translating the framework into a practical IS able to meet the requirements of an error-sensitive GIS. The next chapter deals with the identification of just such a mechanism.

Chapter 3

Object-oriented theory and technology

Object-orientation is increasingly the dominant IS development paradigm. However, the dominance of OO is not in itself sufficient reason to believe that OO is the correct tool for developing error-sensitive or error-aware GIS. This chapter aims to review the current state of OO technology and theory and justify the use of OO in tackling the development of error handling in GIS. Underlying the discussion of the core concepts behind OO in §3.1 is the idea that management of complexity is the key advantage of OO. The importance of complexity management is further highlighted in §3.2 as part of an exploration of OO software development and modelling techniques. The relevance of these advantages to GIS development is illustrated by the review of OO in GI science in §3.3. Following this discussion of concepts and technology, §3.4 introduces a formal theory of objects, used as a basis for the exploration of OO models of error in this research. The conclusions for the chapter are presented in §3.5, while §3.6 provides a selected bibliography for further background reading in some of the topics touched on in this chapter.

3.1 Object-oriented concepts

Object-orientation is a mechanism for structuring and managing the inherent complexity of the real world (Booch 1994). Unfortunately the term remains highly nebulous (Date 1990; Worboys 1994) and a variety of related concepts are often regarded as important to OO. This section surveys first the core OO concepts, likely to be mentioned in any discussion of OO, followed by some important, but more peripheral OO concepts, whilst attempting to highlight how these concepts are geared to manage and reduce complexity in system development.

3.1.1 Core object-orientation concepts

Object-orientation uses three fundamental tactics to address complexity: classification, encapsulation and inheritance. Classification deals with the complexity of a problem from the top down and focuses on the essential properties of an object in the problem domain that distinguish it from other objects. Complementary to classification, encapsulation approaches complexity from the opposite direction and aims to conceal the detailed mechanisms and features of an object. Finally, inheritance allows classified, encapsulated objects to be structured in a hierarchy. The hierarchy allows basic features to be described just once and inheriting classes can then incrementally specialise these basic features.

3.1.1.1 Classification and abstraction

Classification has already been identified as an essential component of OO. The terms abstraction and classification are often used synonymously in the literature. However, to avoid confusion with the (related) concept of abstraction in §2.3, the term classification is preferred here. The aim of classification is to produce idealised patterns or classes that describe the essential features of the system being studied from a particular standpoint. Crucially, a class models both the *structure* and the *behaviour* of different abstractions from the problem domain (Rumbaugh et al. 1991). An object is a particular *instance* of one or more classes, and consequently is composed of both a data structure and behaviours that operate upon that data structure: ie *object* = *state* + *behaviour* (Worboys 1995, p85).

The process of classification is closely related to classical theories of categories. In common with classical categories, classes can be though of as conceptual containers that hold objects with certain properties in common. The importance of categorisation, and so classification, is that it enables complex, detailed concepts to be packaged up into less challenging atomic concepts that can then be used as a basis for further categorisation and classification (Coad 1992). Classification is not unique to OO, and indeed its importance stems from the fact that categorisation and classification appear to be automatic and basic human thought processes (Lakoff 1987).

3.1.1.2 Encapsulation

Encapsulation can be though of as an abstraction over an object's behaviour (or "abstractions over expressions" Atkinson and Morrison 1985, p540). Encapsulation ensures that the detailed mechanisms of an object's behaviour are rigidly separated from the effects of that behaviour, essentially hiding data at one level of abstraction from other abstraction levels. The user of an object need only understand what the behaviour of an object is for a particular level of abstraction, and is actively prevented from accessing more detailed mechanisms and abstractions unnecessarily (Cohen 1984). Encapsulation is a valuable tool in complexity management in the same way as classification: by treating complex entities and systems as atomic building blocks allows the abstraction process to proceed to the next level.

3.1.1.3 Inheritance

Inheritance is the third core OO concept. Whilst classification and encapsulation are employed both separately and in combination in other IS paradigms, inheritance is a feature peculiar to OO and its AI correlates, most notably *frames* (Minsky 1975). Inheritance allows classes, and so objects, to share and develop common features. Inheritance is based on identifying generalisation-specialisation relationships between classes. A generalised *super-class* will exhibit properties common to all its *sub-classes,* which in turn will specialise, adapt and add to these core properties. Happily, generalisation-specialisation relationships are a common feature of complex systems. Through inheritance it is possible to take advantage of these relationships and reduce complexity by specifying the core properties of a class only once.

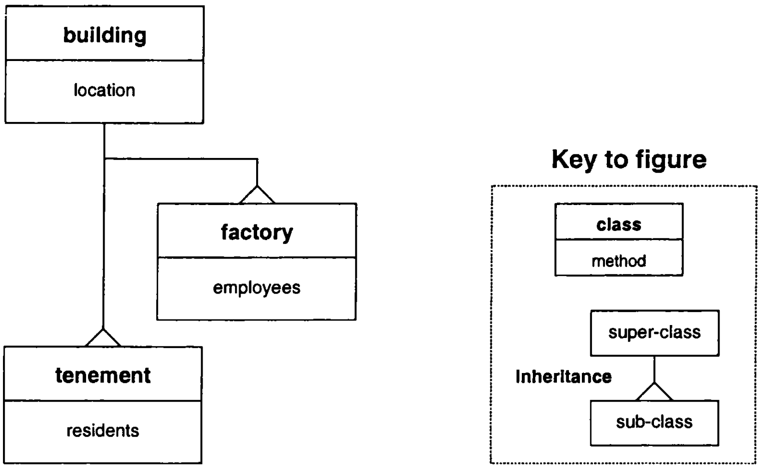


Figure 3.1: Classified inheriting object schema

For example, the classified, inheriting object schema in figure 3.1 has three classes: **building, tenement** and **factory**. An individual building object will be an instance of at least the class **building**. Since **tenement** and **factory** are sub-classes of **building**, an individual factory, say, will be an instance of both **building** and **factory**. The behaviours associated with each class are termed *methods.* The class **building** has one method, **location**. Since the classes **tenement** and **factory** inherit from **building** they automatically possess the **location** method, as well as their own respective **residents** and **employees** method.

3.1.2 Peripheral object-orientation concepts

A number of further concepts can play a peripheral rôle in a discussion of OO. In addition to classification, encapsulation and inheritance, the concepts typing, polymorphism, identity, persistence and composition are the most often cited. This list is not exhaustive, but more marginal

OO concepts such as exception handling, dynamic binding and overloading are not considered here, primarily because their impact is technical rather than conceptual.

3.1.2.1 Types and polymorphism

Type systems are in themselves a significant field of research within computing science. A type system restricts the possible interactions between different elements of an IS, and prevents "embarrassing situations" where types interact in a logically inconsistent way (Cardelli and Wegner 1985). Although not the same, types are related to classes, and informally a class can be thought of as the *type* of an object belonging to that class. For example, assume that in order to function properly a geographical marketing analysis of consumer habits requires objects belonging to class **tenement**. A type system could ensure that such an analysis only ever receives **tenement** objects and never **factory** objects or even plain **building** objects. Type systems, where the same value can take on more than one type are termed *polymorphic* (as opposed to *monomorphic*). Polymorphism is usually important to typed OO systems because an object is expected to take on not simply the type of the class to which it belongs, but also the type of super-classes of that class. For example, a local government taxation analysis might be less discriminating than the marketing analysis about what sort of building it needed. A type system for such a taxation analysis might allow any object of class **building** or any of the sub-classes of **building**, including **tenement** or **factory**, while still avoiding awkward interactions with objects belonging to non-building classes. Such a type system would be polymorphic. In fact, Cardelli and Wegner 1985 identify four different sorts of polymorphism. While all four are all widely used in OO programming languages, only the polymorphism in the above example, termed *inclusion polymorphism,* is wholly relevant to a conceptual discussion of OO.

3.1.2.2 Identity

Objects in the real world possess an identity independent of the object's state. Changes in the number of residents in a tenement building may occur, but the essential identity of the tenement remains unchanged. Similarly, an object within an IS maintains a label that uniquely identifies that object from its creation to its destruction (Worboys 1995). By analogy, the identity of a **factory** object from figure 3.1 is implicitly separate, perhaps contained in some hidden unique ID number, from its **employees** or any of its other attributes. The idea that a computer operating environment (often termed a *virtual machine* or VM) can support individual virtual objects, each object with a separate existence, heightens the correspondence between OO and our perception of reality (Cohen 1984). While this correspondence may be helpful in understanding OO, any such conceptual advantages are serendipitous since identity in OO is once again a primarily technical rather than conceptual invention.

3.1.2.3 Persistence

Persistence, also a technical object-oriented programming (OOP) concept, refers to the length of time for which an object exists. Traditionally, objects only existed during program execution. In order to maintain an object and its state, it used to be necessary to write some explicit code to convert an object in computer memory into a representation of that object on file or in a database. Such conversions are undesirable in part due to the technical effort required to write them. More importantly, however, the mapping effectively destroys the object, negating many of the advantages of OO so that, for example, the conceptual and technical protection afforded by typing will usually be lost in the conversion (Atkinson et al. 1983).

Increasingly, OO environments seamlessly enable objects to persist beyond the termination of the program used to create them. Such environments make little or no distinction between objects created in the current program execution, in a previous program execution or during the execution of a completely different program (Atkinson et al. 1983). Persistence is particularly important when attempting to share objects between programs running on different computers, for example over the Internet.

3.1.2.4 Composition

Many objects in the real world can be viewed as comprising a number of component objects. In a purely structural sense at least, an industrial estate can be considered as composed of a number of **factory** objects from figure 3.1. Worboys et al. (1990) outlines a number of more subtle ways in which objects can be composed of other objects, including aggregation, association and ordered association. Arguably, the different composition relationships are at root a product of the semantic interpretation of inter-object relationships rather than of OO itself. Composition follows implicitly from the concept of 'the ubiquitous object' (Goldsack 1996): the idea that in OO *everything is an object*. Whether attribute, aggregate or associate, each component of an object that makes up that object's state is itself an object[1]. From this viewpoint, it is unsurprising that OO provides fundamental-level support for inter-object relationships. It is possible to place any number of interpretations upon these relationships, which in turn should correspond to our perception of inter-object interactions in reality.

3.2 Object-oriented development tools

The previous section highlighted the technical and conceptual mechanisms for complexity reduction and management embedded in OO. The management of complexity is undoubtedly the

[1] In fact, largely for technical efficiency, object-oriented programming languages (OOPL) have traditionally supplied some primitive (ie not object-oriented) data types, such as `int` and `float`, as a set of core attributes upon which to build objects. In more recent OOPL, for example the Java language (see Flanagan 1996), it is entirely possible to discard primitive data types altogether and to program using 'pure' objects alone.

central advantage of OO over other IS development paradigms (Bhaskar 1983; Booch 1994). The previous section hinted that, in many cases, this advantage has arisen more through technical expediency and a need for efficient program code rather that as a result of deliberate design. However, the development of OO is part of a general trend away from data and implementation dependent software systems toward abstract software models (Abbott 1987). Object-orientation is currently at the leading edge of the evolution of semantic models of IS development that bridge the gap between the human perception of reality and the computerised representation of that perception (Peckham and Maryansk 1988). This section examines the use of the OO system development tools in continuing to close the gap.

3.2.1 Object-oriented analysis and design

Object-oriented analysis and design (OOAD) is a generic technique used to produce OO models, or *schema*, that address a problem for a particular domain. Object-oriented schema employ a variety of tools to represent the classes, objects and inter-relationships used to model the problem domain. These representations may be informal, such as the class diagram in figure 3.1, or more formal, such as the mathematical object systems discussed in §3.4. A clear distinction is usually drawn between the purpose of analysis, which aims to describe *what* a system is supposed to do, and design, which aims to describe *how* a system performs this function (Rumbaugh et al. 1991). Despite this theoretical distinction, most authors acknowledge the existence of a "continuum of representation" (Coad and Yourdon 1991a) where the practical distinction between object-oriented analysis (OOA) and object-oriented design (OOD) is blurred (de Champeaux and Faure 1992; Monarchi and Puhr 1992; Nerson 1992).

A range of OOAD methods have been proposed in a wealth of literature, for which references can be found in §3.6 at the end of this chapter. (eg Coad and Yourdon 1991a; Coad and Yourdon 1991b; Rumbaugh et al. 1991; Booch 1994). Generally, these methods are all, to some degree, iterative and inventive in that they involve the repeated application of a mix of intuition, experience and inspiration along with some less subjective mechanisms. The informal nature of OOAD is a reflection of the paradox at the core of any non-trivial IS, and especially of GIS: any attempt to model the complexity of the real world using the simple formality of an IS inevitably becomes, at some point, messy and subjective. The existence of these 'messy difficulties' encountered, during the transition through the different levels of abstraction from real world to IS, is usually termed *impedance mismatch* (Milne et al. 1993; Worboys 1995). Accepting a necessary degree of subjectivity in the analysis and design process, OOAD attempts to minimise impedance mismatch through the use of OO.

3.2.2 Object-oriented programming

Object-orientation in programming, and in particular the combination of inheritance, encapsulation, persistence and typing, results in code that is rapid to develop and easy to understand, build and test (Bhaskar 1983). Inheritance encourages extensive code reuse, allowing procedures to be written once and the reused again and again. Encapsulation enforces a highly modular programming style, allowing programmers to write and debug much larger code volumes. Persistence can dramatically reduce the volume of code needed for data translation and storage, which can account for as much as 30% of program code (Atkinson et al. 1983). Finally, the protection and structure afforded by typing is essential to large or complex programming tasks. There are now a large number of object-oriented programming languages (OOPL) available, such as Objective-C, C++, Eiffel, Smalltalk. One of the most modern OOPL, the Java language (Sun Microsystems 1999b), is used extensively throughout this research and embodies many of the most up-to-date ideas about OOP and OO generally. The key practical advantage common to all OOP, however, is that complexity management and code organisation are improved allowing more complex software to be developed with less effort.

3.2.3 Object oriented system development

Object-oriented analysis, design and programming have evolved to harness the conceptual power of OO for system development. As system development tools they represent significant advances over other development techniques. However, they are not the only development tools available. A range of analysis and design tools can be used, such as entity-relationship (ER) modelling, structured analysis and requirements engineering. Programming paradigms abound; procedural, functional, logical, rule-based programming each offer particular advantages. It is entirely possible to inter-mix elements of OO development with other development paradigms, for example, ER modelling as an analysis and design tool for OOP, or to program OO schema resulting from OOD using modular programming. The disadvantage of such hybrid approaches is an increase in impedance mismatch.

Object-oriented system development minimises impedance mismatch by smoothing the transition from analysis to design and design to programming. For the analyst, OOA promises an architecture neutral mechanism for resolving complexity in the problem domain into abstracted comprehensible OO schema which are arguably closer to our intuitive understanding of the world around us (Partridge 1994). For the designer, the advantage of OOD is in terms of efficiency of concept, enabling larger and more complex software projects to be implemented and maintained by smaller teams of designers. For the programmer, OOP offers highly efficient code organisation that is able to boost productivity and promote code reuse beyond anything achievable with even the best possible modular programming (Lewis et al. 1992). Crucially, when taken together it is the use of the OO paradigm as a single, consistent heuristic throughout the entire process of

OO analysis, design and programming, which is arguably a central cause of the revolution in OO use (Haythorn 1994). Object-orientation promises, for the first time, a one-stop solution for IS development: the same paradigm can be used from problem definition right through to a working software solution.

3.3 Object-orientation in GIS

Many of the themes highlighted previously in this section are reflected in the literature on object-oriented GIS (OOGIS). GIS are amongst the more complex IS types, by virtue of the spatial component of GI and its correspondence to physical reality (as opposed to, say, financial or economic information systems). Consequently, OO is important to GIS and to error handling in GIS because it offers a mechanism for structuring, managing and reducing the innate complexity of GI.

3.3.1 Technical advantages of OO in GIS

Despite a long legacy of relational, hierarchical and network database software and research, GI science has begun to recognise the technical advantages of OO. The inadequacies of the relational model in particular, which essentially requires all data to be arranged in tables, has plagued GI science for some time. For example, work by Guptill (1992) and Davis and Borges (1994) indicates that object-oriented database management systems (OODBMS) are more suitable than relational database management systems (RDBMS) for very large spatial databases. The constraints of *normal form* (NF) in the relational model can place a great strain upon spatial RDBMS. First normal form (NF1) allows each cell in a table to contain only single atomic values, while other normal forms dictate the manner in which these atomic values are distributed throughout related tables (Date 1990). Unfortunately, geometry in GIS is routinely modelled as sets of ordered coordinate pairs, which are of arbitrary and variable cardinality. To avoid violating various normal forms, spatial RDBMS must maintain a large number of fragmented tables and the keys necessary to relate different tables to each other. In fact, most relational GIS adopt a hybrid approach where well behaved attribute data is stored in an RDBMS and geometry is stored in a separate, proprietary file system (Batty 1992). While 20 years of relational database research has led to the invention of some ingenious mechanisms to circumvent these underlying problems, the constraints of NF can still affect the performance of large spatial RDBMS (Egenhofer and Frank 1992).

In addition to a performance penalty, the fragmented nature of data in spatial RDBMS has a detrimental effect upon database integrity (Davis and Borges 1994). Changes to one of the tables in a RDBMS may have implications for a range of related data distributed throughout the database. The task of tracking such changes may not be trivial and failures can result in a loss of database integrity, where information in the database conflicts with the database's own implicit data model. Batty (1992) notes that the problem is worsened where relational GIS adopt a separate geometry storage system, usually to combat the restrictions of NF. In contrast to the difficulties of maintaining spatial RDBMS integrity, OODBMS allows data to be located in logical, related packets (ie objects) that comprise both data and behaviour. Classes can be defined with data integrity checking built in allowing objects to individually guarantee their own data integrity. Significant reductions in IS complexity can result through tackling data integrity at the object rather than the database level, making it less likely that data integrity will be compromised.

3.3.2 Object-oriented semantic modelling in GIS

Whilst the technical advantages of OO in GIS may help acceptance and uptake of OOGIS, it is the semantic modelling capabilities of OO that have really driven research into OOGIS forward. Ralston (1994) suggests that the use of OOP can simplify the programming for spatial analysis problems, by diverting attention away from the technical details of coding and toward the behaviour of problem domain. While this is indicative of the semantic advantages of OO, Ralston's work does not go as far as to use the semantic modelling capabilities of OO. In one of his examples, rather than concentrate on the semantics of an international food aid distribution problem, such as modes and routes of transport, storage and distribution facilities and locations, shipping and ports, Ralston allows the matrix solution of the distribution system to take precedence, resulting in the classes *row* and *column* being central to the example solution (Ralston 1994). In general, OO allows the detailed algorithms and mathematical models to be encapsulated as object methods, so freeing the OO modeller to concentrate on the semantics of the system being studied.

In fact, it is geometry and topology, the cause of much of the dissatisfaction with the relational model, that have responded best to semantic modelling capabilities of OO. Worboys et al. (1990) highlight the conceptual inefficiency of the relational model with respect to the simple geometric features, point line and area. Worboys (1992) goes much further and offers a comprehensive, object-oriented model of geometric and topological features embedded in two-dimensional Euclidean space, based on combinatorial topology. A parallel research strand began with Guptill and Fegeas (1988) attempting to use the semantic modelling capability of OO to shift the emphasis away from layer-based and toward feature based GIS. Tang et al. (1996) present an integrated OO model of geometry, topology and spatial features founded in Guptill's earlier work with SDTS. Similarly, Milne et al. (1993) indicate how OO and OODBMS can be efficiently integrated with SDTS. The semantic power of OO is a direct result of the development of OO from cognitive science and AI, and a number of authors have taken advantage of the correspondence between OO and AI in the development of OOGIS software with intelligent capabilities (Zhan 1991; Mark and Zhan 1992; Kaindl 1994; Zhan and Buttenfield 1995). In fact, Worboys (1994) notes that to an extent OO modelling may have become a victim of its own success, the proliferation of OO models leading to a haze of bewilderment and misunderstanding surrounding OO. Increasingly, however, the haze is clearing, and OO is asserting itself as an understandable and practical approach to GIS that has already resulted in at least two major commercial GIS that claim to be fully OO (Smallworld and Laser-Scan Gothic) and the inclusion of OO concepts in some form into most commercial GIS.

The OO development process has also enjoyed some success within GIS. Egenhofer and Frank (1989) claim the relational model is simply not appropriate for use with the different levels of abstraction required for complex GI. Both Worboys (1992) and Becker et al. (1996) point to the failure of traditional entity-relationship (ER) modelling to adequately represent geographical features and situations as an advantage of OO modelling techniques and OO extensions to ER. Kösters et al. (1997) has demonstrated that a modified version of OOA, tailored to deal with GIS requirements, can significantly improve the process and results of GIS development. The combination of OO semantic modelling power and OO analysis techniques, then, present exciting opportunities for OOGIS development.

3.3.3 Relational and OOGIS

The relational model has undoubtedly been enormously successful as a data model for GIS. As a result, the move from relational to OO technology has not been without dissenters. It is possible to argue that the relational model is more appropriate for GIS than OO (see, for example the vigorous rebuttal of OO in the after-word of Date 1995). The core of such arguments against OO and in favour of the relational model generally fall into three groups.

- First, that the relational model is computationally more efficient than OO, and consequently RDBMS will always be more efficient than OODBMS.
- Second, the relational model represents a considerable investment in terms of data, finance, expertise, and research that should not be discarded lightly.
- Third, the relational model possesses a sound theoretical basis that contrasts strongly with relatively ill-defined nebulous OO concepts.

None of the allegations against OO are unfounded, but they are also certainly open to interpretation. Often efficiency issues are overstated; relational champions bemoan the lack of indexing in OODBMS, for example (Date 1990, chapter 25). Encapsulation can be a barrier to database indexing, as an object's state will often be hidden behind access methods and not directly accessible to the OODBMS. However, indexing is a largely technical concern, designed with RDBMS query efficiency in mind, so it is perhaps unsurprising that OO does not handle indexing as well as native OO concepts. Becker et al. (1996) show that by subverting the OO model it is entirely possible to allow just the OODBMS and query language access to an object's internal state, thus allowing indexing. Such extreme measures may be unnecessary; modern OOPL such as Java and C++ allow an object's attributes to be declared *public*, if desired, offering unhindered unencapsulated access. More importantly, any technical efficiency gains come at the expense of model semantics; arguably the relational model is computationally more efficient *because* it does not attempt to tackle the difficulties of modelling reality in the same way as OO.

The second issue, that of the relational legacy, is certainly significant. However, legacy systems are exactly what many RDBMS based GIS have become. While the investment in relational technology and expertise cannot be discounted, it is surely in itself not a reason not to develop and use advances in technology and expertise. It is worth emphasising that OO is still relatively new technology. In the 10 years since Chris Date described OO as just a "research direction" with little or no applications base (Date 1990, p24), OO has managed to catch up and arguably overtake 25 years of relational theory and technology. From a historical perspective we should perhaps embrace rather than fear revolution and upheaval in computer systems: a little more than two decades ago *relational* systems were considered just a research direction. The move toward OO systems is part of a wider move towards better semantic modelling, so we should expect OO to be usurped by better modelling techniques in the future. The third issue, that of the strength of relational theory, has for some time been an advantage of using relational technology. An important step in fulfilling the potential of OO with respect to GIS and GI science is the resolution of OO's paucity of theory. The following section is devoted to a discussion of the leading "theory of objects", which aims to close this theory gap.

3.4 A theory of objects

Despite the weight of evidence in favour of using OO to develop error handling in GIS, one crucial shortfall remains. The enormous success of OO is correlated with a proliferation of OO technology, but has not always been complemented by a growth in OO theory. The surfeit of object-oriented analysis, design and programming techniques which now exist are, as has already been noted in §3.2.1, necessarily highly subjective. Experience, personal preferences and choice of OOAD technique and programming language can all play an important rôle in the shape of the software engineered.

The existence of such theory-deficient software engineering techniques may not be of concern in many applications. However, even within the highly results led commercial sector, the use of more formal methods to guide and document OO software development is gathering acceptance (Bowen 1996). Rather than focus entirely on the production of software, this research aims to make some general statements about the problem domain being studied using OO. It should come as no surprise, then, that increasingly the focus of OO research is to provide formal methods to support OO technology and to formulate a comprehensive theory of objects.

The development of the ς (sigma) calculus by Martín Abadi and Luca Cardelli, resulting in the publication of *A theory of objects* in 1996, represents the most comprehensive attempt to date to provide a formal description of object systems. A number of other attempts have been made; λ (lambda) calculus, the foundation of functional programming, has been used with limited success by, for example, Cardelli (1984) and Fiadeiro and Maibaum (1991). However, since λ-calculus uses the function as a primitive construct, object calculi based on λ-calculus tend towards unnecessary complexity (Abadi and Cardelli 1996a). Similarly, approaches such as the use of predicate calculus to formalise object systems (Egenhofer and Frank 1989; Egenhofer and Frank 1990) can quickly become prohibitively complex. The ς-calculus of Abadi and Cardelli makes use of objects as a primitive construct and as a consequence is able to express more fully the features of object systems.

This section continues by introducing the ς-calculus alongside a re-evaluation of the core OO concepts; classification, encapsulation and inheritance of objects. The following discussion of ς-calculus and OO is a compromise between a rigorous discussion of the pertinent features of the ς-calculus and a suppression of some of the more involved features of the formalism. A fuller discussion can be found in the publications of Abadi and Cardelli (1995a, 1995b, 1996a, 1996b) on the subject.

3.4.1 Objects, methods and encapsulation

Objects within the ς-calculus are represented as collections of methods l_i each with bodies b_i. The symbol ς is used to bind the postfixed 'self' parameter (conventionally s or z) with occurrences of that parameter in the body of the method. Each object is enclosed in square brackets and associated with a label using the symbol $\triangleq$ (meaning 'equal by definition'), illustrated in equation 3.1.

$$o \triangleq [l_i = \varsigma(s)b_i \; ^{i\in 1..n}] \tag{3.1}$$

Informally, equation 3.1 defines a new object o which is a collection of n methods, each with distinct labels l_i and bodies b_i, in which references to the object itself may occur using the ς-bound variable s. A method l of an object o can be invoked using the dot operator (written $o.l$). Because a method may contain ς-bound references to the object upon which the method was invoked, a method can operate reflexively and recursively (a method can access other methods on the invoking object). Thus a precise formal definition of an object has been arrived at in equation 3.1: an object is a collection of named methods which may operate reflexively upon the object itself. The object already offers a form of encapsulation, in that it is not necessary to provide details about the method bodies b_i. More importantly, the method bodies are bound to the self and to any subsequent parameters, so the contents of an individual method are not directly accessible to other methods or objects.

3.4.1.1 One-dimensional point example

To clarify, the ς-calculus is illustrated using a simple geographical example. The point object p^1 in equation 3.2 is a ς-calculus representation of simple point in one-dimensional space, $\mathbb{R}^1$. In addition to the field x, that maintains the state of the point, the point possesses a get_x and a set_x method that access and update the current value of the x field, respectively.

$$\begin{aligned} p^1 \triangleq [x = 0,\ get_x &= \varsigma(s)s.x, \\ set_x &= \varsigma(s)\lambda(x')s.x := x'] \end{aligned} \tag{3.2}$$

There are a number of features to note about equation 3.2. First, the field x is in fact a method and strictly should read $x = \varsigma(s)0$. However, since the bound self parameter is unused in the body of the method (0) and no further parameters are defined, the notation $x = 0$ is used to highlight the fact that the method can be interpreted as a field. The example also assumes the existence of real number objects (since $p^1 \in \mathbb{R}^1$). However, this is entirely for illustrative purposes and ς-calculus system for real numbers is left undefined here. The get_x method returns the result of accessing the x field on the self parameter s. The set_x method accepts a new x' coordinate parameter in addition to the obligatory self parameter. The λ in equation 3.2 performs essentially the same function as the ς, binding the postfixed parameter to occurrences of that parameter in the body of the methods. Field update ($s.x := x'$) allows the λ-bound x' parameter to replace the body of the x field in the object s. In fact, since fields are simply a shorthand for methods in which the self parameter is unused, the field update notation is also shorthand for a more general method update notation.

An important feature of the ς-calculus is that it provides for the reduction of terms. For brevity, a formal treatment of reduction is omitted here, but for reference appendix A.1 includes an equational theory for untyped ς-calculus (from Abadi and Cardelli 1996a). Informally, the invocation of a method proceeds by replacing all occurrences of the ς-bound self and subsequent λ-bound parameters in the body of the method with those parameters used to invoke the method (see (Eval Select) in appendix A.1). Equation 3.3 illustrates how the point p^1 object might be used, first updating and then retrieving the value of the x field. The reduction steps are detailed, using the notation $o.l \rightarrowtail b\langle s \leftarrow o \rangle$, to denote the reduction of $o.l$ through the replacement of all occurrences of the self parameter s in the body b of a method l with the object o, where $o \triangleq [l = \varsigma(s)b]$. Reduction steps ($\rightarrowtail$) are distinguished from the simple reordering or rewriting of terms (called syntactic equivalence, $\equiv$).

$$\begin{aligned}
(p^1.set_x(2)).get_x &\equiv ([x = 0,\ get_x = \varsigma(s)s.x,\ set_x = \varsigma(s)\lambda(x')s.x := x'].set_x(2)).get_x \\
&\rightarrowtail ((s.x := x')\langle s \leftarrow p^1,\ x' \leftarrow 2\rangle).get_x \\
&\equiv (p^1.x := 2).get_x \\
&\equiv [x = 2,\ get_x = \varsigma(s)s.x,\ set_x = \varsigma(s)\lambda(x')s.x := x'].get_x \\
&\rightarrowtail (s.x)\langle s \leftarrow p^1\rangle \\
&\equiv p^1.x \\
&\rightarrowtail 2
\end{aligned} \tag{3.3}$$

3.4.2 Classes and inheritance

A class C can be represented in the ς-calculus as in equation 3.4. The equation goes some way to formalising the relationship between objects and classes: a class is simply an object with a *new* method that returns a new object containing the methods specified by that class. In short, a class can be though of as an 'object factory'.

$$\begin{aligned}
C \triangleq [&new = \varsigma(z)[l_i = \varsigma(s)z.l_i(s)\ ^{i\in 1..n}], \\
&l_j = \lambda(s)b_j\ ^{j\in 1..n}]
\end{aligned} \tag{3.4}$$

Invocation of the *new* method on C ($C.new$) produces a new object where each of the template or pre-methods, $l_i = \lambda(s)b_i\ ^{i\in 1..n}$, in the class C are bound to methods in the object being created. The term $l = \lambda(s)b$ is again shorthand for $l = \varsigma(z)\lambda(s)b$ where z is unused. The representation of inheritance can be achieved by creating sub-classes whose pre-methods depend at least in part upon the pre-methods of the super-class. A sub-class C' of C will inherit all of the pre-methods of C ($l_j\ ^{j\in 1..n}$) in addition to adding its own unique methods ($l_k\ ^{k\in n..m}$) as in equation (3.5).

$$\begin{aligned}
C' \triangleq [&new = \varsigma(z)[l_i = \varsigma(s)z.l_i(s)\ ^{i\in 1..n+m}], \\
&l_j = C.l_j\ ^{j\in 1..n}, \\
&l_k = \lambda(s)b_k\ ^{k\in n+1..m}]
\end{aligned} \tag{3.5}$$

3.4.2.1 Two-dimensional point example

Building on the example of a one-dimensional point p^1 in equation 3.2, discussed in 3.4.1.1, it is now possible to define an inheriting, two-dimensional point object schema. Equation 3.6 defines the class P^1, of which the object p^1 is an instance.

$$
\begin{aligned}
P^1 \triangleq [\mathit{new} = \varsigma(z)[x = \varsigma(s)z.x(s),\ \mathit{get}_x = \varsigma(s)z.\mathit{get}_x(s),\ \mathit{set}_x = \varsigma(s)z.\mathit{set}_x(s)],\\
x = \lambda(s')0,\ \mathit{get}_x = \lambda(s')s'.x,\ \mathit{set}_x = \lambda(s')\lambda(x')s'.x := x']
\end{aligned}
\tag{3.6}
$$

The reduction sequence in equation 3.7 indicates how the invocation of $P^1.\mathit{new}$ correctly returns a new one dimensional point object.

$$
\begin{aligned}
P^1.\mathit{new} &\equiv [\mathit{new} = \varsigma(z)[x = \varsigma(s)z.x(s),\ \mathit{get}_x = \varsigma(s)z.\mathit{get}_x(s),\ \mathit{set}_x = \varsigma(s)z.\mathit{set}_x(s)],\\
&\qquad x = \lambda(s')0,\ \mathit{get}_x = \lambda(s')s'.x,\ \mathit{set}_x = \lambda(s')\lambda(x')s'.x := x'].\mathit{new}\\
&\rightarrowtail [x = \varsigma(s)P^1.x(s),\ \mathit{get}_x = \varsigma(s)P^1.\mathit{get}_x(s),\ \mathit{set}_x = \varsigma(s)P^1.\mathit{set}_x(s)]\\
&\rightarrowtail [x = \varsigma(s)\lambda(s')0(s),\ \mathit{get}_x = \varsigma(s)\lambda(s')s'.x(s),\ \mathit{set}_x = \varsigma(s)(\lambda(s')\lambda(x')s'.x := x')(s)]\\
&\rightarrowtail [x = 0,\ \mathit{get}_x = \varsigma(s)s.x,\ \mathit{set}_x = \varsigma(s)\lambda(x')s.x := x']\\
&\equiv p^1
\end{aligned}
\tag{3.7}
$$

Finally, a two-dimensional point class P^2 can be now defined such that $P^2.\mathit{new} \in \mathbb{R}^2$ and P^2 inherits from P^1, as in equation 3.8. The new pre-methods y, get_y, set_y of the class P^2 appear as expected, but the pre-methods x, get_x, and set_x simply re-use the methods of the same name from P^1 in equation 3.6. Instances of the class P^2 will possess both x and y coordinates and the ability to access and update these coordinates.

$$
\begin{aligned}
P^2 \triangleq [\mathit{new} = \varsigma(z)[x = \varsigma(s)z.x(s),\ \mathit{get}_x = \varsigma(s)z.\mathit{get}_x(s),\ \mathit{set}_x = \varsigma(s)z.\mathit{set}_x(s),\\
y = \varsigma(s)z.y(s),\ \mathit{get}_y = \varsigma(s)z.\mathit{get}_y(s),\ \mathit{set}_y = \varsigma(s)z.\mathit{set}_y(s)],\\
x = P^1.x,\ \mathit{get}_x = P^1.\mathit{get}_x,\ \mathit{set}_x = P^1.\mathit{set}_x,\\
y = \lambda(s')0,\ \mathit{get}_y = \lambda(s')s'.y,\ \mathit{set}_y = \lambda(s')\lambda(y')s'.y := y']
\end{aligned}
\tag{3.8}
$$

3.4.3 Future development of ς-calculus

While a degree of subjectivity is inevitable in IS development (§3.2.1) the lack of a credible theory of objects has in the past been a hindrance to OO and promoted ambiguity and informality in OO system development. For the first time, the existence of a fundamental theory of objects presents the possibility of addressing some of the shortfalls of OO technology. Object theory has, in effect, caught up with object technology. Although undoubtedly complex, the introduction to ς-calculus presented here is necessarily somewhat superficial and leaves much unsaid. Type systems and reduction strategies, for example, are touched upon in the next chapter, but a full treatment is beyond the scope of this research. Extensions to the ς-calculus already underway are likely to yield clearer, simpler more concise formalisms that are better equipped for this type of discussion. One important product of the formal object-systems research should be a high-level yet fundamental object language. However, the ς-calculus is still relatively new and a degree of mathematical complexity is therefore unavoidable.

3.5 Conclusions

"Why another data model?" is the first of two important questions posed by (Brodie 1984, p22) with regard to the adoption of new modelling approaches. From the point of view of GI science the answer is simple: more than most information types, GI is inherently complex (§3.3). In order to capture the complex semantics of geographical reality, it is important that the chosen modelling technique be sophisticated enough to address this complexity. Conventional data models, such as the relational model, are simple, efficient and highly suitable for use with IS, but do not go as far as to adequately model geographical complexity. Therefore, there is a clear need for a model of reality that retains some of the simple efficiency of the relational model, but not at the expense of the ability to grapple with high level concepts rather than low-level details.

The second question is "What is the original contribution of the new data model?" (Brodie 1984, p22). Object-orientation, more than any other data model, offers both inbuilt complexity management and minimises impedance mismatch over the development cycle. Object-orientation enables system developers to manage complexity through the OO abstraction mechanisms of classification, encapsulation and inheritance. The consistent use of the OO heuristic throughout the analysis, design and programming of IS ensures impedance mismatch is minimised at the same time as maximising semantic modelling capability. Object-orientation, then, offers the best possible arsenal of tools to manage the inherent complexities of GI not addressed by other modelling techniques.

"The choice of an appropriate representation for the structure of a problem is perhaps the most important component of its solution" ((Worboys et al. 1990), p369). In attempting to develop error-sensitive and error-aware GIS, the modelling technique must simultaneously address the complexities of GI and of error and uncertainty. The degree to which the chosen technique is able to provide effective complexity management will therefore have a profound effect on the ultimate shape of the resultant error handling software. The choice of OO as the modelling technique used in this research follows directly from this need to manage complexity.

At the same time as allowing better modelling of the highly complex phenomena of spatial data quality, the use of OO as a development heuristic for an error-sensitive GIS should safeguard the aims of flexibility, efficiency and understandability already identified as important. The following chapter combines the OO concepts, theory and tools explored in this chapter with the conceptual model of spatial data quality identified in the previous chapter to produce a theoretical basis for an error-sensitive GIS.

3.6 Selected bibliography

The discussion of OO and object theory has covered a wide range of topics. Readers not already familiar with some elements of this discussion may find a number of core texts helpful when confronted with these topics for the first time. A number of important texts exist for those unfamiliar with OOAD. OOA and OOD are treated separately by Coad and Yourdon (1991a) and Coad and Yourdon (1991b) respectively. Rumbaugh et al. (1991) and Booch (1994) present two similar OOAD methods that have more recently been conflated, resulting in the formation of Rational Software and the Unified Modelling Language (UML, Rational Software 1999). The Rational OOAD process and UML are described by Jacobson et al. (1999) and Booch et al. (1999) respectively, although unfortunately these more recent books are closely tied to Rational Software's commercial products rather than OOAD more generally. UML, now the *de facto* industry standard for OOAD, is firmly situated in a practical software engineering tradition rather than in a theoretical tradition. This emphasis on practice and results rather than on theory and process means that despite being the industry standard, UML is not well suited to a research environment. Instead, as indicated in §3.4, theory-led techniques are preferred for research in OO systems. The existence of a sound theoretical basis is the key reason for using ς-calculus rather than UML in this research. The first five chapters of *A theory of objects* by Abadi and Cardelli (1996a) provide a thorough and informal background to OO concepts generally, followed by a comprehensive exploration of the ς-calculus. However, *A theory of objects* is the only book on ς-calculus currently available.

A general treatment of the subject of OOP is given in Budd (1991) and Voss (1991). However, OOP is best understood by learning to program using a good OOPL, such as Java. Arnold and Gosling (1996) provides an excellent beginner's level text for those starting to learn Java, although strangely inaccurate in places, while Flanagan (1996) provides a slightly more focussed discussion of Java for those familiar with procedural languages, such as C. This chapter was only able to touch upon λ-calculus. Hankin (1994) and Barendregt (1984) are core texts on λ-calculus from a computer science and mathematical perspective respectively, while Révész (1988) is recommended as a supplementary text due to its very practical approach.

Chapter 4

Error-sensitive GIS: theory

Chapter 2 closed by looking at a conceptual model of data quality that was flexible and expressive enough to offer a suitable basis for the development of an error-sensitive GIS. The use of OO was proposed in chapter 3 as the most appropriate method for capturing and managing the complexities inherent in GI and spatial data quality. This chapter describes the first step in the production of an OO error-sensitive GIS — the application of OOA to the conceptual model of data quality in §2.3. The result of this process should be an analysis that retains both the flexibility and expressiveness of the conceptual model of data quality, and the robust, architecture neutral, understandability of OOA. The chapter begins with a brief discussion of the analysis process, followed by a tour of the primary analysis results. The properties of the analysis are further scrutinised using the ς-calculus.

4.1 An object-oriented data quality model

This section details the initial results of the OOA of the conceptual model of data quality. The analysis indicates a clear OO structure for data quality. Following a brief discussion of the OOA process in §4.1.1, a discussion of this structure leads to the presentation of an OO model of data quality storage. This model exhibits the desired behaviour while at the same time being simple and potentially implementable in any OO database, including existing OOGIS.

4.1.1 Object-oriented analysis process

The different OOA techniques outlined in §3.2.1 all possess broad similarities. This research made use of elements from the three major analysis methods (ie from Coad and Yourdon 1991a; Rumbaugh et al. 1991; Booch 1994), but did not attempt to follow any one slavishly. Neither was any attempt made to incorporate the geographically biased extensions to OOA proposed by Kösters et al. (1997), since spatial data quality itself is, as discussed later, predominately aspatial. Increasingly, the different methods are in any case converging, as illustrated by the development of UML touched on in §3.6. The assumption made here is that the differences between analysis methods are unlikely to have a significant effect upon the results of this analysis, particularly since this research goes to considerable effort to formalise its results using object calculus.

The first steps of the analysis process involve looking at the problem domain in a highly abstracted manner, picking out the most general features and in effect 'taking a step back'. One of the first results is a *data dictionary* that contains the candidate classes, objects, relationships and behaviours. The analysis proceeds by progressively and iteratively refining and structuring the data dictionary into a high-level OO model of the problem. A variety of techniques can be employed in reporting the results of an OOA, but the most important is the *class diagram*, of which figure 3.1 is a simple example. The graphical notation used consistently throughout this research is based on that of Rumbaugh et al. (1991) since it is, arguably, the simplest, clearest and most focussed of the various possible notations.

The iterative nature of both the analysis and the design process means that the analysis results, as presented here, are not a faithful representation of the analysis process. Repeated alterations and improvements were made over a period of two years or more and while attempts are made to highlight where earlier analysis versions failed, the final analysis results are obviously a composite of the best elements from a large number of iterations of the analysis process.

4.1.2 Analysis results

The conceptual model of data quality presented in §2.3 allowed a fundamental distinction to be made between spatial data and spatial data quality: that data quality only comes into existence as a result of the process of abstracting and representing the real world. From this standpoint, an obvious first step in OOA is to identify two new object-oriented classes of data quality: abstractive quality and representative quality. Arguably, these two classes are fundamental to any discussion of data quality.

4.1.2.1 Representative quality

The process of representation inevitably entails the introduction of error. *Representative quality* is data quality that records error introduced through the representation of the terrain nominal. An example is the SDTS quality element positional accuracy. Positional accuracy is defined as the difference between an observed location of a geospatial feature and the 'true' or ideal location of that feature (NCDCDS, 1988). The 'true' location can be found in the ideal data set, ie the terrain nominal. Similarly, the SDTS quality element lineage, often seen as the starting point for any data quality standard or report (Aalders 1996), records the process history of data. Lineage therefore has implications for how an actual data set may differ from the terrain nominal so its definition is also consistent with the concept of representative quality.

Further investigation of this line of enquiry reveals two additional features of representative quality that are required to model the entire expected range of representative quality behaviour. First some representative data quality elements are only meaningful when the data to which they refer is of a particular type or metric. For example, the concept of positional accuracy is only sensible when discussing spatial as opposed to thematic data. The OO data quality model needs to be able to restrict the scope of some representative data quality elements to defined metrics.

Second, a few representative data quality elements can be thought of as compound, while most cannot. For example, a data object might be annotated with a constantly updated list of lineage objects detailing different operations and events through which the data object has passed. In contrast, the same data object is expected to have originated from a single data collection event, and so be annotated with at most one source object. Lineage is an example of a compound representative data quality element, whilst source can be regarded as a special, restricted case of a representative data quality element. The compound behaviour of representative data quality effectively sets the cardinality of a quality object's relationship with a data object and consequently needs to be reflected in the OOA. This cardinality will depend entirely on the definition of the representative quality element. In contrast to the unitary source quality element in the example above, it is entirely reasonable to require a source quality element, say, that reflects the multiple data sources of some compound data object. In such a case both the definition and the interpretation of the compound source quality element are fundamentally altered from that of the unitary source quality element in the prior example.

Crucially, from an OO perspective representative quality operates at an object level rather than a class level. The OOA of representation suggests that representative quality is expected to vary from object to object. There is no particular reason why one object should possess the same lineage, say, as a second object, even if they are of the same class.

4.1.2.2 Abstractive quality

The terrain nominal is by definition an incomplete description of the real world. However, for a particular abstraction of the real world it may be desirable to ensure that certain properties of the real world persist in the terrain nominal and so into the data set. *Abstractive quality* is defined here as data quality that supports or informs the linkage between the real world and the terrain nominal, links which might otherwise have been lost in the process of abstraction. The CEN quality element abstraction modifier is an example of the concept. An abstraction modifier is a textual record of the distortion resulting from the process of abstraction (CEN/TC287 1996) and is clearly an abstractive data quality element. The SDTS quality element logical consistency is another example, although it is not particularly useful in an OO environment. Logical consistency is intended as a check on the logical content and structure of data. For example, logical consistency in the form of topological consistency can be used in contour data to check that the height of an individual contour falls somewhere between its topological neighbours and that contour lines do not cross. This behaviour can be enforced with the concept of an abstractive data quality element called logical consistency. However, an OOA of contours would usually encode such behaviour within the contour class itself, negating the use of abstractive logical consistency in this example. Through careful use of OOA, logical consistency can effectively be removed from the data quality discussion.

In contrast to representative quality, OOA suggests that abstractive quality operates exclusively at the class level. Abstraction produces simplified classes of objects from complex real world objects. Any deficiencies in the process of abstraction will be felt equally by all objects of a particular class. For instance, the degree of abstraction given by an abstraction modifier is expected to be homogeneous across all objects of a particular class.

4.1.2.3 Quality storage model

Object-oriented analysis allows a sharp line to be drawn between class based abstractive quality and object based representative quality. Borrowing from OOP, the term *static* is used to describe class-based properties such as abstractive quality, as opposed to object-based properties such as representative quality, since static properties remain unchanged by the (dynamic) instances of that class. The class diagram in figure 4.1 illustrates many of the key results from the OOA. In particular, it forms the basis for the core structure of an error-sensitive GIS and details the data quality storage model.

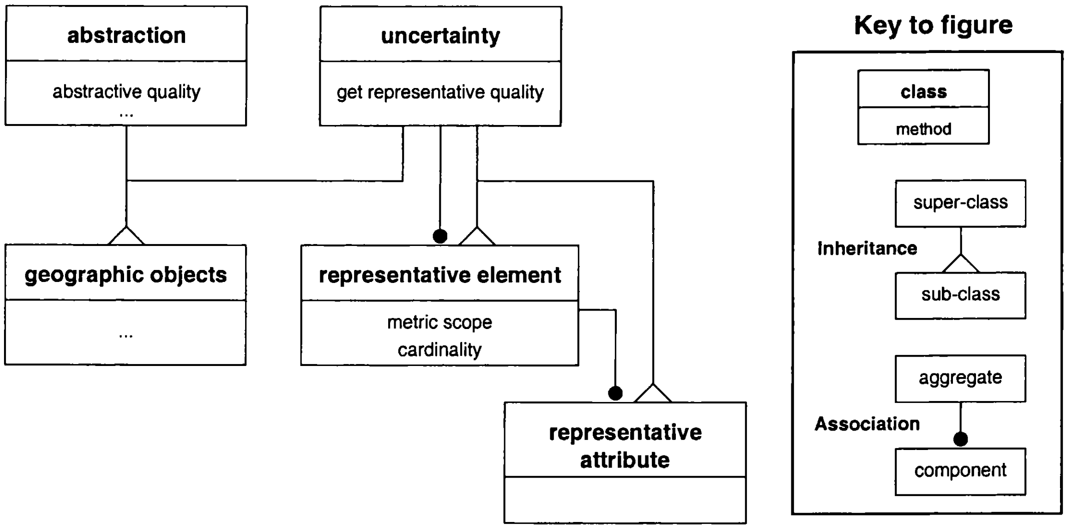


Figure 4.1: Class diagram of OOA results

With reference to figure 4.1, the class **representative element** is the super-class of all representative quality elements. A **representative element** is comprised of a collection of objects of class **representative attribute**, which in turn define the individual attributes of a **representative element**. For example, positional accuracy could be constructed by creating a new class **positional accuracy**, which inherits from **representative element**. This class would be an aggregate of a number of new classes each inheriting from **representative attribute**, for instance, **x-RMSE** and **y-RMSE**. Additionally, the **metric scope** of the **positional accuracy** class would be set to restrict objects of class **positional accuracy** to referring to spatial objects only. The **cardinality** behaviour would be set such that at most one **positional accuracy** object could refer to the same geospatial data object. The actual choice of structure is, however, entirely free and at the discretion of the database designer, as long as the quality classes conform to two requirements. First, new quality classes must inherit from **representative element** and so possess a **metric scope** and a **cardinality** method. Second, new quality classes must follow the basic pattern of being a named collection of **representative attributes**. In fact in the case of the second requirement, it would be possible to defer definition of even this relatively relaxed structure to the database designer. However, experience with actually using the results of this OOA indicates that **representative element** objects that do not conveniently conform to this simple structure are very infrequent. Consequently, the requirement can be imposed without loss of model flexibility or generality.

The class **geographic objects** represents the super-class of all geospatial data objects in the terrain nominal. The class **uncertainty** has one method, **get representative quality**. Since the class **geographic objects** inherits from the class **uncertainty**, *every* instance of **geographic objects** in the database will have its own **get representative quality** method with which to access its own representative data quality. All **geographic objects** also inherit from **abstraction**. The **abstraction** class contains any number of methods that outline the supported abstractive quality. In figure 4.1 the ellipsis below the method **abstractive quality** emphasise that other abstractive quality methods can also be used. Since abstractive quality methods are inherited, they will be identical for all **geographic objects** of a particular class. It does not necessarily follow that these methods are identical for all sub-classes of **geographic objects**. Each new sub-class of **geographic objects** is free to redefine or override the abstractive quality methods inherited from **abstraction**. The **abstraction** class simply guarantees that all **geographic objects** possess the 'hooks' on which to hang abstractive quality methods.

The requirement for all geospatial objects to inherit from **abstraction** and **uncertainty** is the only restriction placed upon the geospatial objects that can be represented in the database. Inheritance allows the transmission of error-sensitive behaviour to all sub-classes of **geographic objects**. The implication is that any OO database could be supported, including existing databases.

4.2 Formal analysis model

Traditional OOA might stop here and proceed onto the design stage. However, the analysis results presented so far, while plausible, are still relatively informal. Following the analysis of the problem domain, the progression to OO design can be problematic. Additionally, the analysis is still rather vague and potentially ambiguous. The aim of this section is to tighten up the analysis results using the ς-calculus and so minimise ambiguity and head off potential design problems before they occur.

4.2.1 Formal data quality storage model

The first step in formalising the data quality storage model is to obtain a simple mapping from the graphical class diagrams to ς-calculus terms. In fact it is a relatively straightforward and mechanical process to reformulate the class diagram in figure 4.1 as ς-calculus terms. Definitions 4.2.1–4.2.4 give the results of a direct mapping from figure 4.1 to ς-calculus terms for four of the five classes in figure 4.1: **abstraction, uncertainty, geographic objects** and **representative element** using the abbreviations *Abs, Unc, Geo* and *Rep* respectively. The class **representative attribute** is omitted from the discussion for brevity.

Definition 4.2.1

$$\begin{aligned} Abs \triangleq [&new = \varsigma(z)[abs\_qual = \varsigma(s)z.pre_{abs1}(s)], \\ &pre_{abs1} = \lambda(s)b_2] \end{aligned}$$

Definition 4.2.2

$$\begin{aligned} Unc \triangleq [&new = \varsigma(z)[get\_rep = \varsigma(s)z.pre_{unc1}(s)], \\ &pre_{unc1} = \lambda(s)b_1] \end{aligned}$$

Definition 4.2.3

$$\begin{aligned} Geo \triangleq [&new = \varsigma(z)[abs\_qual = \varsigma(s)z.pre_{geo1}(s), \\ &\quad get\_rep = \varsigma(s)z.pre_{geo2}(s)], \\ &pre_{geo1} = Abs.pre_{abs1}, \\ &pre_{geo2} = Unc.pre_{unc1}] \end{aligned}$$

Definition 4.2.4

$$\begin{aligned} Rep \triangleq [&new = \varsigma(z)[get\_rep = \varsigma(s)z.pre_{rep1}(s), \\ &\quad metric = \varsigma(s)z.pre_{rep2}(s), \\ &\quad card = \varsigma(s)z.pre_{rep3}(s)], \\ &pre_{rep1} = Unc.pre_{unc1}, \\ &pre_{rep2} = \lambda(s)b_3, \\ &pre_{rep3} = \lambda(s)b_4] \end{aligned}$$

In themselves, the ς-calculus terms in definitions 4.2.1–4.2.4 do not provide any significant information other than that already provided by the class diagram in figure 4.1. The terms are untyped and no attempt is made to supply any information on the working of any of the encapsulated method bodies $b_1 - b_4$ yet. However, they do form the basis of a range of further analysis that can be used to explore the secondary properties of the data quality storage model. Three important secondary properties of the quality storage model are investigated in the remainder of this section under the headings multiple inheritance, meta-quality and efficient data storage.

4.2.2 Multiple inheritance

The results of the OOA are expected to be architecture neutral, and consequently implementable in any OO system, even an existing OODBMS. However, even a very general class diagram, such as that in figure 4.1 can offer implementation problems. The class **geographic objects** inherits from both **uncertainty** and **abstraction**. The inheritance of one sub-class from more than one super-class is termed *multiple inheritance.* Multiple inheritance is often very useful during OOA and generally OOA does not proscribe its use (Coad and Yourdon 1991a; Booch 1994). Many OOPL such as C++ and Eiffel permit the use of multiple inheritance. For a variety of practical reasons many OOPL, such as Java and Smalltalk, only permit single inheritance where all sub-classes must inherit from at most one super-class. The conflict between multiply and singly inheriting OO environments became of particular relevance to this research, since the next chapter discusses the implementation of the results of this analysis using two separate OO environments: the Java OOPL which only supports single inheritance, and the Gothic OODBMS which supports multiple inheritance. There are of course an ever increasing number of OO environments available, and no particular reason to believe that either the semantically superior multiple inheritance model or the practically superior single inheritance model will eventually dominate. In order for the OOD to proceed for a programming environment that supports only single inheritance, it is necessary to reformulate the OOA results. Figure 4.2 illustrates the reformulation of figure 4.1 needed to allow the design to proceed for any singly inheriting OO environment such as Java.

Such conflicts are conventionally resolved during the design and programming of the software, albeit in an informal manner. This in itself is not necessarily a problem. Software development is an iterative process and the analysis results are by no means set in stone once the design is underway. However, it is in the interests of concept reuse not to make changes to the analysis on an *ad hoc* basis dependent on the current design strategy. Any such changes may adversely affect the properties of the analysis or its applicability to other OO environments in subsequent designs. By representing the object schema using the ς-calculus, it is possible to more closely control and manage the translation from single to multiple inheritance and ensure the results preserve the properties of the original multiply inheriting schema. The following simple example uses untyped ς-calculus to support the translation of the OOA results (figure 4.1) from multiple to single inheritance and shows that this translation does have a limited effect upon the working of the schema.

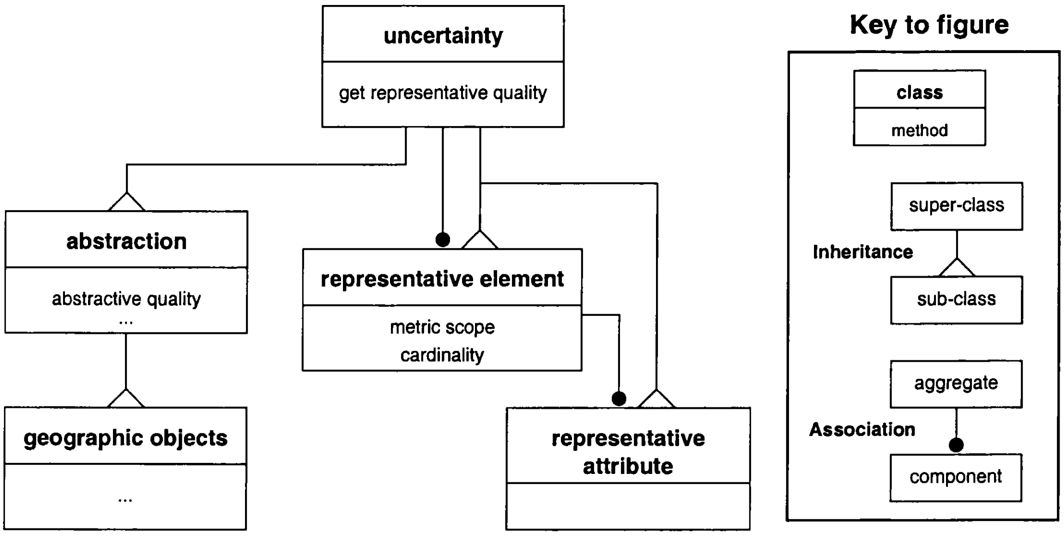


Figure 4.2: Class diagram of singly inheriting OOA results

4.2.2.1 Using untyped ς-calculus

It is possible to obtain a mapping for the singly inheriting classes shown in figure 4.2 and so further extend the object system. The terms for Abs' and Geo' in definitions 4.2.5 and 4.2.6 correspond to the redefined **uncertainty** and **abstraction** classes respectively in figure 4.2.

Definition 4.2.5

$$\begin{aligned} Abs' \triangleq [new = \varsigma(z)[&abs\_qual = \varsigma(s)z.pre_{abs1}(s),\\ &get\_rep = \varsigma(s)z.pre_{abs2}(s)],\\ pre_{abs1} &= \lambda(s)b_2,\\ pre_{abs2} &= \lambda(s)\,Unc.pre_{unc1}] \end{aligned}$$

Definition 4.2.6

$$\begin{aligned} Geo' \triangleq [new = \varsigma(z)[&abs\_qual = \varsigma(s)z.pre_{geo1}(s),\\ &get\_rep = \varsigma(s)z.pre_{geo2}(s)],\\ pre_{geo1} &= Abs'.pre_{abs1},\\ pre_{geo2} &= Abs'.pre_{abs2}] \end{aligned}$$

The untyped ς-calculus equational theory given in appendix A.1 has already been introduced (§3.4.1.1). Equality between objects (and so classes) is defined in the ς-calculus where two objects have exactly the same methods in any order (written $\leftrightarrow$ and shown in (Eq Object) in appendix A.1). By evaluating the terms for pre_{geo1} and pre_{geo2} in definition 4.2.3 and the terms for $pre_{geo1}{}'$ and $pre_{geo2}{}'$ in definition 4.2.6 it is possible to prove that the classes Geo and Geo' are equal.

The proof (theorem 4.2.1) requires the use of two judgments from the equational theory in appendix A.1. As in §3.4.1.1, the judgment (Eval Select) allows the method invocation reduction where the invoking object o replaces all occurrences of the self parameter s in the body b of the method l $(o.l \rightarrowtail b\langle s \leftarrow a\rangle)$. The judgment (Eq Object) states that two objects are equal if each of their respective methods are inter-convertible up to reordering. Again, recent publications by Abadi and Cardelli (1996a, 1996b) contain fuller explanations of the equational theory.

Theorem 4.2.1 *With reference to definitions 4.2.1 to 4.2.6 inclusive, the untyped ς-calculus terms Geo and Geo' are equal.*

Proof. The terms Geo and Geo' each have only three methods. We consider each case. *Case 1*: The body of the $Geo.pre_{geo1}$ method is $Abs.pre_{abs1}$ which reduces to $\lambda(s)b_2$ in one step by (Eval Select). Similarly, the body of the $Geo'.pre_{geo1}$ method is $Abs'.pre_{abs1}$ which also reduces to $\lambda(s)b_2$ in one step by (Eval Select). *Case 2*: The body of the $Geo.pre_{geo2}$ method is $Unc.pre_{unc1}$. The body of the $Geo'.pre_{geo2}$ method is $Abs'.pre_{abs2}$ which also reduces to $Unc.pre_{unc1}$ by (Eval Select). *Case 3* The bodies of both the $Geo.new$ method and the $Geo'.new$ method are $[abs\_qual = \varsigma(s)z.pre_{geo1}(s), get\_rep = \varsigma(s)z.pre_{geo2}(s)]$. Since the classes Geo and Geo' are composed of inter-convertible methods, by (Eq Object) we conclude $Geo \leftrightarrow Geo'$. □

This in turn implies that the reformulation of multiple to single inheritance of the schema in figure 4.1 required by implementations, such as Java, which do not support multiple inheritance, will not affect the working of the **geographic objects** class. In the same way it is possible to show that the ς-calculus terms for the classes Abs and Abs' class (definitions 4.2.1 and 4.2.5) are not equal. While the same results could have been achieved informally by studying the class diagrams or even via other formal methods perhaps using graph theory, the example introduces the concept of using formal proofs to reason with statements about object systems. It is worth noting that while adequate for this purpose, the equational theory of Abadi and Cardelli (1996a) is quite limited, and more flexible equational theories have been proposed (eg Gordon and Rees 1996).

4.2.3 Meta-quality

Many data quality standards, such as SDTS, restrict quality information to referring only to geographic information: the concept of quality as meta-data. The term *meta-quality* refers to quality information about quality (Aalders 1996). The CEN draft paper on geographic data quality is part of a more general movement to reporting not only meta-data but meta-quality information. The draft proposes that confidence, reliability and methodology should be reported for quality information (CEN/TC287 1996). Whilst the CEN proposed standard is more extensive than SDTS, it still restricts itself to a number of predefined meta-quality elements and to a single level of self-reference. Once the concept of meta-quality is acknowledged as important, however, there is no particular reason to impose these restrictions. Preferable to the arbitrary prescription of meta-quality elements and structure would be an error-sensitive GIS design free from such limitations.

Such considerations influenced the analysis results in figure 4.1. The class **representative element** itself inherits from **uncertainty**, consequently inheriting the association with **representative element** objects. It follows that meta-quality information can be assigned recursively if desired; any representative quality object can itself have representative quality information associated with it. Recursion within object schema has been recognised not only as an important tool in the development of OO systems, but also as a potential source of unexpected and undesirable object behaviour particularly when used in conjunction with multiple inheritance (Blaschek and Frölich 1998). By using the ς-calculus to represent the objects involved, it should be possible for developers to explore the properties of a system, communicate those properties in a formal way and avoid the pitfalls presented by the use of recursion.

4.2.3.1 Using typing rules within ς-calculus

The previous example made use of untyped ς-calculus. However, the introduction of types as a method for categorising object terms can increase the expressive power as well as the complexity of the ς-calculus. In this example, the first order ς-calculus with sub-types ($Ob_{1<:}$) of Abadi and Cardelli (1996a) is used. Type annotations are used to convey information about the type of methods and objects. For example, equation 4.1 states method l is of type B and equation 4.2 states type A is composed of methods l_i each of type B_i.

$$l : B \tag{4.1}$$

$$A ::= [l_i : B_i^{i \in 1..n}] \tag{4.2}$$

It is possible to give a type for an object of class *Unc* as in definition 4.2.7. This term defines an uncertainty type, *UncType*, with one method: a *get_rep* method which returns a representative quality object of type *RepType*. The representative quality type *RepType* (definition 4.2.8) is related to the type *UncType* in that it has all the methods of *UncType* in addition to some unspecified methods, indicated with ellipsis. Consequently, *RepType* is a sub-type of *UncType* (written *RepType* <: *UncType*).

Definition 4.2.7

$$UncType ::= [get\_rep : RepType]$$

Definition 4.2.8

$$RepType ::= [get\_rep : RepType, ...]$$

The term for a representative quality object *o* of type *Rep* can be written as in definition 4.2.9.

Definition 4.2.9

$$o : RepType \triangleq [get\_rep = \varsigma(s : UncType)b_3, ...]$$

In order to demonstrate that this property of meta-quality holds generally for the object system, it is necessary to be able to access the *get_rep* method of any object *o* of type *RepType* in the object system recursively, as in equation 4.3.

$$\begin{aligned}
&o : RepType \\
&o.get\_rep : RepType \\
&(o.get\_rep).get\_rep : RepType \\
&((o.get\_rep).get\_rep).get\_rep : RepType \\
&\ldots
\end{aligned} \tag{4.3}$$

Since we know $o : RepType$ is, by definition, well typed (definition 4.2.9) it is enough to show that the expression $o.get\_rep : RepType$ is also well typed to allow us to infer by induction that all the expressions in equation 4.3 are also typable. In fact, the expression $o.get\_rep : RepType$ is typable and a proof can be found in theorem 4.2.2. The example highlights one way in which the ς-calculus can be used to extend conventional OOA to allow the properties of an object schema to be verified and explored. In theorem 4.2.2, reference is made to a typing judgment, (Val Select), which can be found in appendix A.2. Typing judgments are very similar to the judgments in the untyped equational judgments in appendix A.1. The only difference, aside from the presence of type annotations, is that typing judgments are made within a particular typing environment, Γ, which essentially holds information about all the types available to the type system. Informally, the judgment (Val Select) can be used to determine the type for the result of method invocation upon an object of known type.

Theorem 4.2.2 *For the ς-calculus type* $RepType ::= [get\_rep : RepType, ...]$ *suppose* $o : RepType$. *If* $\mathbb{P}(x.l, n)$ *represents* n *successive application of the* l *method upon* x *such that* $\mathbb{P}(x.l, n) \triangleq (((x.l_1).l_2)...).l_n$ *then* $\mathbb{P}(o.get\_rep, n) : RepType$.

Proof. We use mathematical induction. *Base case:* $o : RepType$ by definition. *Induction step:* Let n be an arbitrary natural number and suppose $o' : RepType \triangleq \mathbb{P}(o.get\_rep, n) : RepType$. By (Val Select)

$o'.get\_rep : RepType$, which simplifies to $\mathbb{P}(o.get\_rep, n+1) : RepType$. Thus $\mathbb{P}(o.get\_rep, n) : RepType$ implies $\mathbb{P}(o.get\_rep, n+1) : RepType$. □

4.2.4 Efficient storage model

Object oriented analysis and design have already been defined as the *what* and *how* of OO system development respectively. Issues such as memory requirements and data volumes are usually considered entirely the preserve of the design stage of development (de Champeaux and Faure 1992). However, the storage of data quality elements associated with spatial information presents situations where even these relatively clear cut issues can become enmeshed in the continuum of representation.

Even within the NCDCDS data quality standard, a single item of information can be associated with up to four different items of quality information (lineage, consistency, completeness and either positional or thematic accuracy). Subsequent standards have tended to proliferate the number of quality elements used, many of which can meaningfully refer simultaneously to the same geographic information. The CEN draft standard, for example, can contain as many as 13 different types of data quality elements. If the existence of meta-quality information is also taken into account, it is clear that any spatial database, if saturated with data quality information, might increase in size by an order of magnitude or more. Such volumes of data place a strain not simply on the technology, but more importantly place a question mark over the validity of storing data quality elements at all. The *cost* of digital data storage is continually tumbling and so the storage of huge volumes of data quality information may be an option in some situations: in the future, terabyte storage media may supersede the megabyte or even gigabyte storage media commonly used today. However, intuitively the *value* of data quality information is in some way constrained by the *value* of the geospatial information to which it refers: data quality information 'adds value' to geospatial information. As a consequence, an organisation which stores gigabytes of geospatial data is unlikely ever to want to store terabytes of associated quality information, irrespective of how economical data storage media become. The effort involved in storing thousands of times more data quality information than geospatial information will always be difficult to justify as long as data quality information is not thousands of times more valuable than geospatial information. In terms of the discussion of OOAD in §3.2.1, the design of an error-sensitive GIS has the question of data volume as a *what* as well as a *how*.

4.2.4.1 Efficient quality storage

Given that data volume is an analysis issue in the context of developing an error-sensitive GIS, it is possible to develop strategies to minimise data volume. One such strategy takes advantage of the generally hierarchical nature of aggregated objects in an OO spatial database. A highly simplified example of the aggregation relationships which might exist in a vector OOGIS is shown in figure

4.3. Here the representation of a river is actually the aggregation of a number of lines, which in turn are aggregates of a number of points. Because each class of spatial objects in the database inherits from the **geographic objects** class in figure 4.1, the model used here allows individual representative quality objects to refer to any one of these spatial objects. A first step in minimising data quality volumes is to allow spatial objects in the database to infer quality from the objects of which they are a part. Assuming, say, Line A in figure 4.3 has a quality object associated with it, the point objects which make up Line A (Points A, B and C) should be able to infer their quality from the aggregate line.

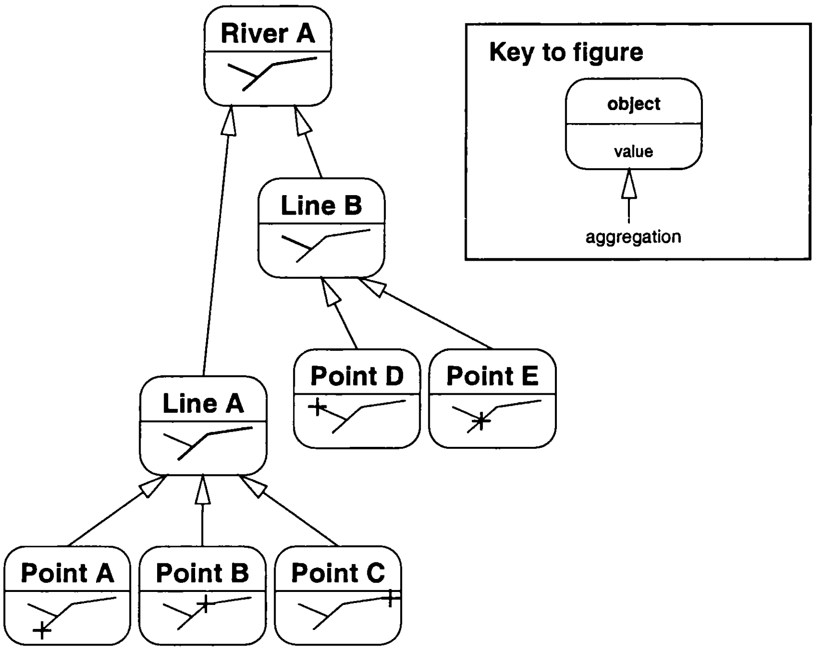


Figure 4.3: Geospatial object hierarchies

Unfortunately, the details of such a system very quickly become very difficult to explain using the traditional OO analysis tools of diagrams and text alone. The exact rules of precedence are important in avoiding conflicts, but can be complicated to communicate. What happens if we set the quality of a geospatial object that is currently inferring data quality from its parent? What happens if we set the quality on a parent where some or all of the child objects are already populated with data quality objects? OOAD methods do offer limited assistance: Booch (1994) for example uses a combination of specialised state transition diagrams, object diagrams and interaction diagrams to try to convey such information. However, such traditional OOAD techniques inevitably (and in part intentionally) restrict the analyst and the designer in their attempts communicate implementation and system details. Even within research, where analyst, designer and programmer may be the same person, the provision of a mechanism for exploring the details of a proposed system at the analysis stage is preferable to ignoring such details until the programming stage, when they are likely to be overwhelmed by other programming considerations. By using the ς-calculus it is possible to precisely define and communicate how precedence is to be handled in the efficient quality storage model. The following section introduces and explores just such a precise storage model. However, it is worth noting that the model developed here is just one possibility, and in fact a variety of different precedence rules could be used, as discussed later in §10.2.2.

The ς-calculus is able to capture the detailed working of an object system in an expressive yet implementation independent way. For this we use the untyped imperative calculus (Abadi and Cardelli 1996a) as it supports discussions about an object's state, as opposed to the stateless calculi used in the previous sections. The object *unc* in equation 4.4 forms the basis of an approach to efficient data quality storage. The object contains *true* and *false* objects and a simple *if-then-else* construct such as might be found in virtually every modern programming language, OO or otherwise. In fact, these conditional operators are not an addition to the calculus. Abadi and Cardelli (1996b) show that they can be constructed from pure ς-calculus. However, using the if-then-else tokens in place of their pure ς-calculus counterparts simplifies significantly the resultant terms.

$$
\begin{array}{l}
\mathit{unc} \triangleq [\mathit{has\_quality} = \mathit{false}, \mathit{quality} = [\,],\\
\quad \mathit{parent} = [\mathit{prop\_rep} = \varsigma(s)\lambda(c)[\,], \mathit{get\_rep} = [\,]],\\
\quad \mathit{child} = \varsigma(s)[\mathit{add\_sib} = \varsigma(z)\lambda(c)s.\mathit{child} := c, \mathit{set\_sib\_rep} = \varsigma(z)\lambda(s)[\,],\\
\qquad \mathit{unset\_child\_rep} = [\,]],\\
\quad \mathit{next\_child} = \varsigma(s)[\mathit{add\_sib} = \varsigma(z)\lambda(c)s.\mathit{next\_child} := c,\\
\qquad \mathit{set\_sib\_rep} = \varsigma(z)\lambda(s)[\,], \mathit{unset\_child\_rep} = [\,]],\\
\quad \mathit{get\_rep} = \varsigma(s)\underline{\text{if}}\ s.\mathit{has\_quality}\ \underline{\text{then}}\ s.\mathit{quality}\ \underline{\text{else}}\ s.\mathit{parent}.\mathit{get\_rep},\\
\quad \mathit{set\_rep} = \varsigma(s)\lambda(q)\underline{\text{if}}\ s.\mathit{has\_quality}\\
\qquad \underline{\text{then}}\ s.\mathit{quality} := q,\\
\qquad \underline{\text{else}}\ s.\mathit{child}.\mathit{unset\_child\_rep};\\
\qquad\quad s.\mathit{parent}.\mathit{prop\_rep}(s);\\
\qquad\quad s.\mathit{quality} := q; s.\mathit{has\_quality} := \mathit{true};\\
\quad \mathit{prop\_rep} = \varsigma(s)\lambda(c)\underline{\text{if}}\ s.\mathit{has\_quality}\\
\qquad \underline{\text{then}}\ s.\mathit{child}.\mathit{set\_sib\_rep}(s.\mathit{quality});\\
\qquad\quad s.\mathit{quality} := [\,]; s.\mathit{has\_quality} := \mathit{false};\\
\qquad\quad c.\mathit{quality} := [\,]; c.\mathit{has\_quality} := \mathit{false};\\
\qquad \underline{\text{else}}\ s.\mathit{parent}.\mathit{prop\_rep}(s),\\
\quad \mathit{add\_sib} = \varsigma(s)\lambda(c)s.\mathit{next\_child}.\mathit{add\_sib}(c),\\
\quad \mathit{add\_parent} = \varsigma(s)\lambda(p)s.\mathit{parent} := p; p.\mathit{child}.\mathit{add\_sib}(s);\\
\quad \mathit{set\_sib\_rep} = \varsigma(s)\lambda(q)s.\mathit{quality} := q; s.\mathit{has\_quality} := \mathit{true};\\
\qquad s.\mathit{next\_child}.\mathit{set\_sib\_rep}(q);\\
\quad \mathit{unset\_child\_rep} = \varsigma(s)s.\mathit{quality} := [\,]; s.\mathit{has\_quality} := \mathit{false};\\
\qquad s.\mathit{next\_child}.\mathit{unset\_child\_rep}; s.\mathit{child}.\mathit{unset\_child\_rep};]
\end{array}
\tag{4.4}
$$

It is necessarily not possible to provide a satisfactory explanation of the working of the ς-calculus terms in definition 4.4 using text and diagrams alone. However, it is possible to give a flavour of the working in an informal way. Figure 4.4 gives four objects in an aggregation relationship which echos figure 4.3. The ς-calculus terms for these four objects can be obtained by creating four clones of the original *unc* objects, o_1, o_2, o_3 and o_4 as in equation 4.5. The $\mathit{clone}(a)$ operation produces new object copy of the original object a with the same labels and methods. At the same time, the existence of a number of representative quality objects $q_1 ... q_n$ is assumed, although for simplicity these objects are not elaborated on here.

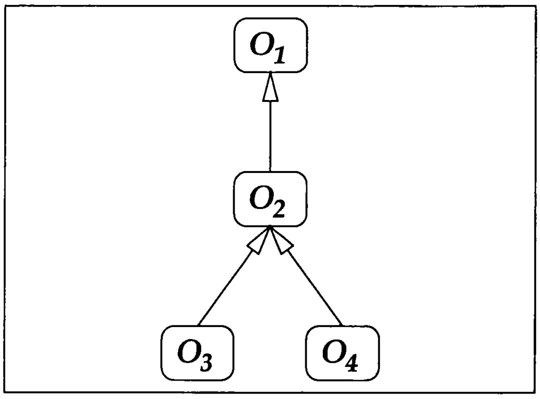


Figure 4.4: Aggregation in ς-calculus terms

$$o_i = clone(unc) \quad i \in 1..4 \tag{4.5}$$

$$\begin{gathered} o_2.add\_parent(o_1) \\ o_3.add\_parent(o_2) \\ o_4.add\_parent(o_2) \end{gathered} \tag{4.6}$$

The expression $o_2.add\_parent(o_1)$ updates the parent object o_1 with a reference to object o_2 and creates a reciprocal reference from the child object o_2 to o_1. A series of such method invocations (equation 4.6) can be used to complete the mapping from ς-calculus term to the aggregation relationships suggested by figure 4.5.

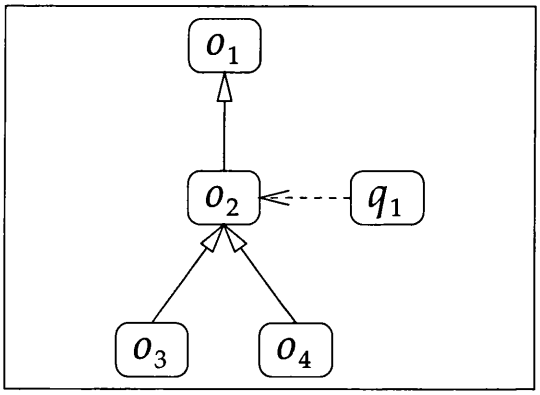


Figure 4.5: Setting quality objects

$$o_2.set\_rep(q_1) \tag{4.7}$$

$$o_2.get\_rep \twoheadrightarrow q_1 \tag{4.8}$$

$$o_1.get\_rep \twoheadrightarrow [\,] \tag{4.9}$$

$$\begin{gathered} o_3.get\_rep \twoheadrightarrow q_1 \\ o_4.get\_rep \twoheadrightarrow q_1 \end{gathered} \tag{4.10}$$

Having obtained this mapping, it is possible to set the quality of any of the uncertainty objects $o_{1..4}$ using the *set_rep* method. For example, the effect of term $o_2.set\_rep(q_1)$ is illustrated in figure 4.5. The ς-calculus can be used to show that having set the quality of an uncertainty object, the *get_rep* method can be used to retrieve that quality element, as in equation 4.7. The working of this reduction is omitted, but the reduction of the term $o_2.get\_rep$ in equation 4.8 proceeds much as in the examples in §3.4.1.1 and §3.4.2.1 and yields q_1. The symbol $\twoheadrightarrow$ is used to denote such a many-step reduction. Objects above o_2 in the aggregation (ie o_1) are unaffected by this change to the schema. Accessing the *get_rep* method on o_1 yields only the empty object ([ ]) rather than a quality object, as in equation 4.9. In contrast, objects below o_2 in the hierarchy (ie o_3 and o_4) will infer quality from their parent object due to the inference mechanism encoded in the *get_rep* method of the *Unc* class, as in equation 4.10.

The term in equation 4.4 would be excessively complex were that the limit of its behaviour. Rather than simply infer quality from parent objects, the model can actively maintain the minimum number of quality objects by purging the object hierarchy of redundant quality information as subsequent references to quality objects are added. For example, if a new quality object q_2 is added to o_3 (equation 4.11), q_2 will supersede the inference mechanism for that uncertain object. Consequently, the reference from uncertain object o_2 to quality object q_1 is removed (equation 4.12) and propagated to all the child objects of o_2 other than o_3 (ie o_4 in equation 4.12). These changes are reflected in figure 4.6.

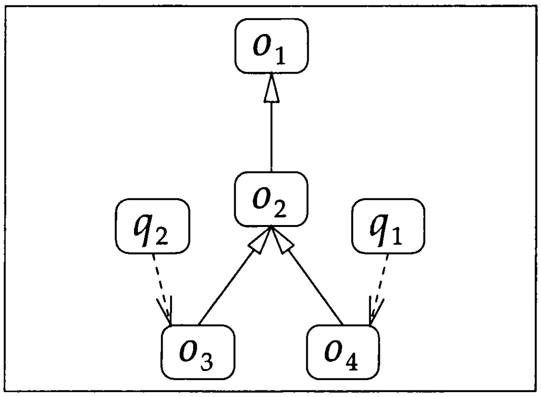


Figure 4.6: Propagation of quality objects

$$o_3.set\_rep(q_2) \tag{4.11}$$

$$\begin{aligned} o_3.get\_rep &\twoheadrightarrow q_2 \\ o_2.get\_rep &\twoheadrightarrow [\,] \\ o_4.get\_rep &\twoheadrightarrow q_1 \end{aligned} \tag{4.12}$$

Finally, if the quality of an object higher up in the aggregation hierarchy, such as o_1, is set (equation 4.13) all the references to quality objects below this point will be removed, to allow the inference mechanism to assert itself once again (equation 4.14). Again these changes are informally represented in figure 4.7.

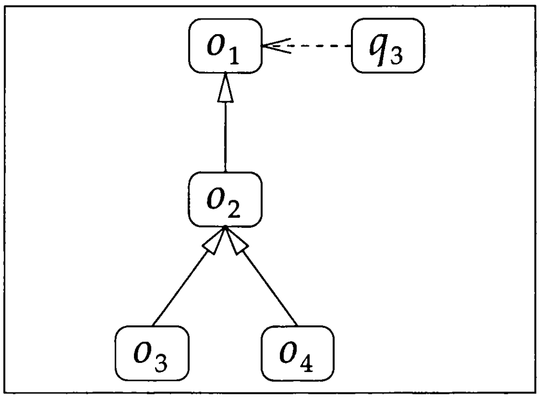


Figure 4.7: Overriding quality objects

$$o_1.set\_rep(q_3) \tag{4.13}$$

$$\begin{aligned} o_2.get\_rep &\twoheadrightarrow q_3 \\ o_3.get\_rep &\twoheadrightarrow q_3 \\ o_4.get\_rep &\twoheadrightarrow q_3 \end{aligned} \tag{4.14}$$

4.2.5 Use of formal object systems

This section has illustrated a cross-section of difficulties that may be encountered by the system developer during the transition from OO analysis to design. It may be beneficial to step out of the discussion of data quality in GIS briefly and look at the implications of this work for the use of formal object systems in GIS and IS development more generally. Whilst the literature acknowledges the indeterminacy of the boundary between OOA and OOD, underlying much of the OO development process is the assumption that the movement of information and distinction between OOA and OOD is reasonably well behaved.

This situation is illustrated in figure 4.8A, after Monarchi and Puhr (1992). Here, information about the problem domain is captured initially with OOA and subsequently further information is captured with OOD. Crucially, the OOD process encompasses all of the OOA results and the boundary between OOA and OOD is distinct. This model of the development process is, however, incomplete. Taking each example application in turn, a characterisation of the use of object calculus within the OOAD process is proposed based on the deficiencies in the model of the OOAD process shown in figure 4.8A.

The first of the three example uses of the ς-calculus deals with the representation of multiple inheritance in both OOA and OOD. Information captured about the problem domain in the form of a multiply inheriting object, may not be usable in the design without reformulating the multiple inheritance as single inheritance. The general situation where, despite information being captured by the OOA, a design cannot incorporate those features, is shown in figure 4.8B. Information can leak out of the development process at the boundary between analysis and design. The ς-calculus can be used to resolve any conflicts resulting from this information leakage, providing a route to reconcile the analysis with the design. Inheritance strategies are only one possible area where such information leakage can occur. As a consequence of the inherent subjectivity in OO development some mismatch between the concepts used in OOA and in OO programming

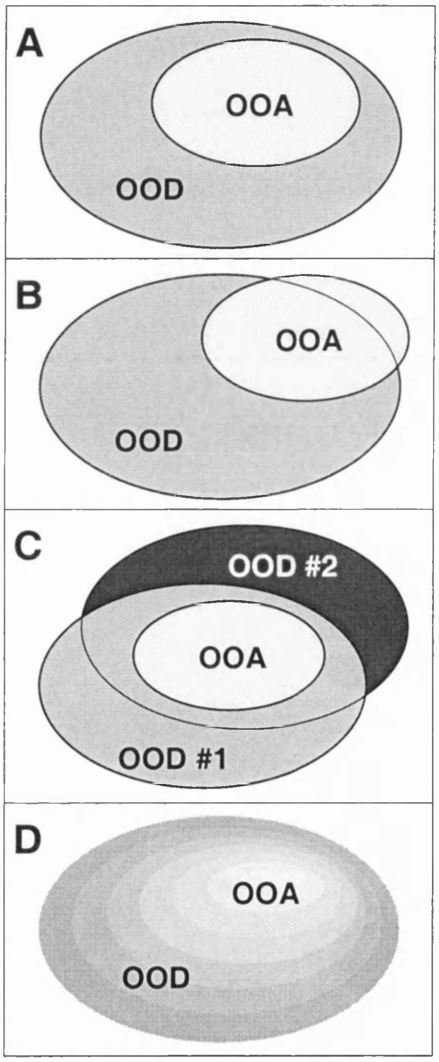


Figure 4.8: Summary of OOAD transition

environments is to be expected.

The second example looks at the need for a capacity to explore the properties of the results of an OOA. Within the commercial world, it may be enough to verify that the OO software produced fulfills all the client's requirements, to be satisfied that the development process used to produce that software has been a success. However, it may be unreasonable to infer the properties of an OOA from a software product produced from that analysis. In figure 4.8C, despite the OOA being captured in its entirety by more than one design it is not possible to make any assumptions about the general properties of the analysis from those designs. Different designs will extend the analysis results in different ways. Simply because one design and implementation possesses certain properties does not imply that another necessarily will. By using the ς-calculus to formalise and explore the analysis, it is possible to make statements about the core properties of the analysis which, in turn, should be fundamental to any design based on that analysis.

Finally, the ς-calculus was used to provide a highly expressive, implementation independent platform to communicate specific OOA results. The detailed working of an object system is usually excluded from the OOA process and is regarded as the *how* of system design rather than the *what* of problem definition. This does encourage the analyst to focus on general rather than implementation issues. However, equally it is the lack of an implementation independent language able to communicate the detailed working which has led to the proscription of detailed working within OOA. The example of the volume of data quality required to describe the uncertainty associated with spatial information shows that, in some cases, the detailed working can become a central part of the problem definition. The general situation is illustrated in figure 4.8D, where there is no distinct boundary between OOA and OOD. Consequently, no part of the development process is guaranteed to be exclusively the preserve of analysis or of design.

4.3 Conclusions

An OOA of the conceptual model of data quality is able to produce a simple, understandable, plausible OO data quality schema and from this research seems to offer two key benefits. First, the OO data quality schema seems to possess the desirable properties of the conceptual model of data quality. It does not depend on any particular data quality standard or error model and consequently should be flexible and expressive enough to represent any data quality element. Second, the schema should also possess the desirable properties of OO, namely that it should be semantically and conceptually close to the original model of data quality. Further, the implementation of the OO schema should be compatible with any OO environment, including existing OODBMS. The use of inheritance can allow the properties of the OO schema to be transmitted throughout an entire OO database allowing every geospatial object in the database, whether spatial or aspatial, to access its own data quality.

The use of ς-calculus is more than able to provide formal support for the spatial data quality analysis process. In particular, the ς-calculus highlighted several secondary properties of the OOA results. First, while the informal analysis results may not be entirely implementation independent, ς-calculus can be used to look for any implementation dependence and to guarantee architecture neutrality where problems are discovered. Second, the use of meta-quality is of increasing importance to data quality standards. The analysis results support this progression and allow an extremely flexible approach to meta-quality. Finally, the issue of data storage volumes is also of key practical importance to any error-sensitive GIS. Using the ς-calculus, analysis-level object systems can be provided to combat data storage volume issues in an architecture neutral way that would simply not be possible using conventional OOA. Building on these analysis results, the following chapter looks at the implementation of the results of this analysis and to what extent the promise of the OOA results can be realised in practice.

Chapter 5

Error-sensitive GIS: implementation

The previous chapter attempted to provide a solid theoretical basis for an error-sensitive GIS. Building on this theory, this chapter describes the implementation and operation of the error-sensitive GIS software. Central to the error-sensitive GIS theory summarised in §4.3 was the suggestion that the approach could support *any* reasonable quality element, associated within *any* database object within *any* OODBMS. The first three sections of this chapter (§5.1–5.3) examine the implementation process alongside these three implementation criteria: support for any OO database, any database object and any quality element. In order to illustrate the functionality of the error-sensitive database functionality, the operation of an error-sensitive ς-calculus object system is explored in §5.4. The chapter concludes with a final review of the error-sensitive GIS architecture (§5.5).

The implementation process necessarily entailed a considerable amount of programming. Occasionally, program code fragments are needed to illustrate a wider point. Wherever possible, discussions of program code are avoided here in order to prevent this chapter becoming excessively technical. Comprising the compact disc (CD-ROM) that accompanies this thesis and an index to that CD-ROM at the back of this volume, appendix C contains documentation, source code, and compiled code relating both to this chapter and to chapters 6–9.

5.1 Error-sensitive GIS implementation: Any database

The error-sensitive GIS analysis results were implemented in two entirely separate OO environments. First, a prototype was implemented using Java OOPL rather than an OODBMS. The key differences between OOPL and OODBMS are that the latter offers query handling, transaction processing, concurrency and most importantly persistence (Worboys 1999). In the case of Java, however, there are a variety of technologies that blur the distinction between programming language and database. For example, a persistent version of Java has been developed (Atkinson et al. 1996). The core Java release now offers lightweight persistence by allowing objects to be encoded as an input/output (IO) stream, termed *serialisation* (Harold 1997). Querying and connecting to existing (OO or non-OO) databases using Java Database Connectivity (JDBC) is now well established (Sun Microsystems 1999a).

The full error-sensitive GIS was implemented using Laser-Scan Gothic OOGIS. A side effect of using two different OO environments is that together they support the contention that the error-sensitive OOA results can be implemented within very different OO environments. However, given the diminishing distinctions between Java OOPL and OODBMS, the two implementations additionally lend implicit support to the contention that the error-sensitive GIS can be supported by any OODBMS. The remainder of this section briefly introduces the contrasting OO environments, Java and Gothic, and outlines the process of implementation for each environment.

5.1.1 Java prototype implementation

The production of a working software prototype can be an important component of the software development cycle (Yourdon 1989). Producing a prototype implementation can significantly help in uncovering and resolving analysis and design flaws, and improve the efficiency of subsequent implementation processes (Friedman and Cornford 1989). Java proved an ideal prototyping language for this research; it is a highly modern OOPL that allows powerful prototyping with greater rapidly than is possible with older OOPL, such as C++. The prototype error-sensitive GIS differed from the full implementation in two key respects. First, the prototype is not persistent, so objects created during program execution are lost once that program execution is finished. As suggested above, it would be a relatively trivial programming task to equip the prototype with persistence, perhaps using serialisation. Second, since the prototype is based solely on the OOA results in the previous chapter it has none of the generic spatial functionality found in a GIS.

The discussion of multiple inheritance in §4.2.2 alluded to Java's lack of support for multiple inheritance. In fact, Java does provide a limited form of support for multiple inheritance, via special classes called *interfaces*. Programming languages often make a distinction between where a program construct, such as a method, function, field or variable, is *declared* and where it is *defined* (Kernighan and Ritchie 1988). Declaration involves giving types, arguments or identifiers (names) to program constructs. Definition involves actually stating the implementation or value for a particular declared program construct. Along with many other OOPL, Java allows methods to be *abstract*, meaning that a method is declared but not defined. An interface is a special Java class that possesses only abstract methods. A Java class can inherit from at most one other class, but from any number of Java interfaces (Arnold and Gosling 1996). Despite this concession to semantics of multiple inheritance, the lack of program code in interfaces still means in practice Java supports only single inheritance. Consequently, the prototype error-sensitive GIS design and implementation was based on the singly inheriting analysis in figure 4.2 rather than the multiply inheriting schema in figure 4.1. The prototype error-sensitive GIS source and working compiled code can be found in appendix C on CD-ROM. Java provides a documentation tool called javadoc that was used to produce hyper-text mark-up language (HTML) documentation for the prototype, also on the CD-ROM.

5.1.2 Laser-Scan Gothic implementation

Defining precisely what is meant by the term "OOGIS" can be fraught with difficulty. Different GIS that adopt the "OO" designation may actually have an OO graphic user interface (GUI), an OO database design interface, an OODBMS or any combination of these. For the purposes of this research, the important feature for the GIS is that it supports an OODBMS. Arguably the results might also translate to hybrid OO/RDBMS, often termed *extended relational* databases (Loftus et al. 1995). Extended relational databases, such as POSTGRES or Environmental Systems Research Institute (ESRI) MapObjects, offer an OO database design interface to what is essentially an RDBMS. Laser-Scan Gothic was chosen for use with this research because it is one of relatively few OOGIS that claims to support a fully OODBMS.

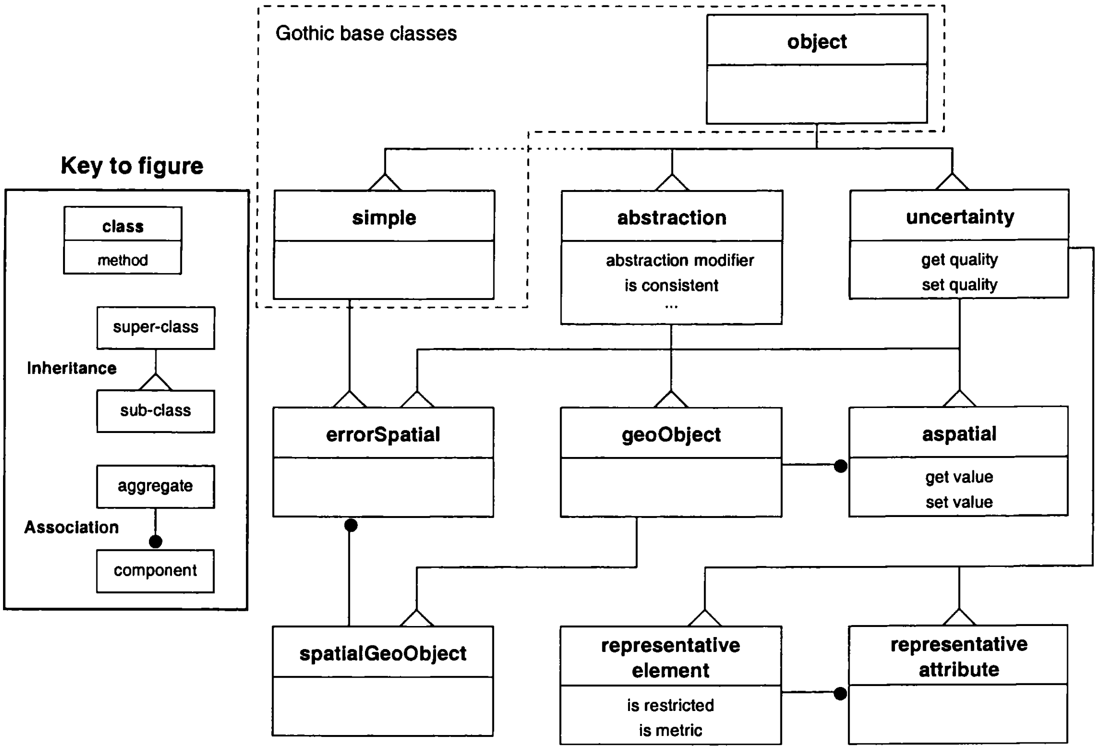


Figure 5.1: Gothic error-sensitive GIS design

The class diagram in figure 5.1 illustrates the results of the OOD process for the full Gothic implementation. This class diagram exhibits a number of developments from the OOA results in figure 4.1 upon which the design was based. The class **spatialGeoObject**, intended to be the super-class of any complex spatial object in the database, has been added along with its super-class **geoObject**. The class **geoObject** is composed of a number of **aspatial** attributes, while its sub-class **spatialGeoObject** is aggregated from a number of **errorSpatial** geometries. The class **errorSpatial** inherits from the Gothic base class **simple**, which provides a range of core geometry and topology object behaviours. All classes in Gothic inherit from the generic super-class **object**. This is the first point in the development process that has explicitly excluded field-based data — **simple** employs vector rather than raster geometry. Arguably, a limited number of modifications could allow the same approach to be used within a raster-based mapping context, although this has not been tested. The representative quality classes, **representative element** and **representative attribute** remain essentially unchanged from the analysis results in figure 5.1.

5.1.3 Implementation contrasts

The Gothic error-sensitive GIS is written using Laser-Scan's proprietorial C-like programming language, called *Lull*. The Lull source code for the Gothic error-sensitive GIS can be found in appendix C on CD-ROM. Unlike Java, Lull offers no documentation tool. Consequently, while the code is fully documented, the documentation is embedded within the Lull code. The Lull programming language is strikingly different to Java, in that it is *not* an OOPL. Where Java code directly defines a class, Lull code indirectly scripts a class' definition. When executed, this Lull script is used by the Gothic OODBMS to build classes and manipulate objects. The code in figure 5.2 gives an example Lull declaration of the class **uncertainty**. The code begins with a Lull function call to set up links to the Gothic database environment, and continues by scripting the **uncertainty** class: first declaring the class, then declaring the inheritance, then declaring the two methods **get quality** and **set quality**. The code in figure 5.3 is also a class declaration for **uncertainty**, but this time written in Java. Instead of scripting how to build an **uncertainty** class, the Java code in figure 5.3 declares the class and methods directly. Consequently, the Java code in figure 5.3 is much closer to the ς-calculus type declaration of **Uncertainty** in figure 5.4 than the Lull code in figure 5.2. However, both Java and Lull code fragments perform essentially the same function.

The most obvious conclusion to draw from the differences between figures 5.2 and 5.3 is that Lull is less intuitive and more verbose than Java. However, there are more subtle implications. Since Java classes are declared once, then compiled and used, changes to a Java class will need to be recompiled before they can take effect. In contrast, Lull classes are dynamically scripted at run time, they can be created, deleted or altered at any point while the Gothic OODBMS is running. In turn, this allows Java to be strongly typed, performing exhaustive type checks at compile time, whereas Gothic database objects are untyped. Loss of typing has practical consequences, primarily that Gothic is much harder to program and to debug than Java. However, loss of typing also has theoretical implications. Typing imposes restrictions upon the statements that can be made in an object system (§3.1.2.1). Typed statements will still hold in an untyped universe, it is simply that the typing restrictions ensure that unsound untyped statements are prevented from occurring in the typed universe. The discussion in §4.2.3 used typing rules to show that meta-quality is

```
# lull function create uncertainty class #
function integer create_uncertainty_class(FRAME f)
begin
  # Declare variables #
  integer st;  VAC vac_id;

  # Set up links to Gothic environment #
  st:=frame_fetch_value_resources(frame_id, "Vac", vac_id);

  # Define new uncertainty class #
  st:=meta_define_class(vac_id,"uncertainty","uncertainty_group");

  # Uncertainty inherits from gothic object class #
  st:=meta_inherit(vac_id,"uncertainty","object");

  # Declare get quality method on uncertainty #
  st:=meta_define_value(vac_id,"uncertainty","get_quality",
                        "uncertainty_group",
                        META_SCOPE_GLOBAL,DT_DESCRIPTOR,
                        DDT_COLLECTION,MVT_SET,MST_METHOD,
                        1,"quality_name",DT_STRING);

  # Declare set quality method on uncertainty #
  st:=meta_define_value(vac_id,"uncertainty","set_quality",
                        "uncertainty_group",
                        META_SCOPE_GLOBAL,DT_BOOLEAN,
                        DDT_INVALID,MVT_SINGLE,MST_METHOD,
                        1,"quality_object",DT_OBJECT_ID);
  return GOTH__NORMAL;
end;
```

Figure 5.2: Lull code uncertainty class declaration

a property of the typed analysis results. While the use of an untyped environment, such as Gothic, does not prevent this property holding, it does perhaps weaken the belief that the untyped Gothic implementation will behave in the same predictable way as the typed object-calculus system.

5.2 Aggregation relationships: Any object

The prototype Java implementation, outlined in §5.1.1, formed a useful proof of concept before embarking upon the full Gothic implementation. However, as already mentioned, the two OO environments Gothic and Java contrast starkly. Differences in inheritance strategies have already been noted, the prototype Java implementation being based on a singly inheriting OOA rather than the original multiply inheriting analysis. However, most significant amongst the further difficulties encountered were the lack of support for aggregation and the extensive use of non-object data types within Gothic.

```
// Declaration of class uncertainty
public abstract class uncertainty extends object{

  // get quality method declaration
  public abstract Collection get_quality(String quality_name);

  // set quality method declaration
  public abstract Boolean set_quality(Quality quality_object);
}
```

Figure 5.3: Java code uncertainty class declaration

$$\begin{aligned} Uncertainty \triangleq [&get\_quality : String \rightarrow Collection, \\ &set\_quality : Quality \rightarrow Boolean] \end{aligned}$$

Figure 5.4: ς-calculus uncertainty type declaration

The discussion in §3.1.2.4 drew attention to the importance of composition and aggregation in object systems. In particular, §3.1.2.4 attempted to show how aggregation is a consequence of the OO maxim "everything is an object". In Gothic, unlike Java, *not* everything is an object. In fact, within Gothic a surprisingly large amount of the database is not OO: neither attributes nor geometry are objects in Gothic. This proved an unexpected and serious drawback of using Gothic. The OOA depends on using OO inheritance to transmit the core error-sensitive GIS properties throughout the entire database. Even for true objects in the Gothic database, aggregation is severely limited. Gothic cannot directly handle objects that are attributes of another object. All inter-object relationships in Gothic must be encoded indirectly as complex and inflexible Gothic 'references'. These Gothic references are more reminiscent of the keys used in relational tables than an OO concept. Indeed, the capability for modelling complex, aggregated geographic objects was highlighted in §3.3.3 as a central reason for moving from RDBMS toward OOGIS, a capability Gothic largely lacks.

The problem highlighted the limitations of the error-sensitive GIS. The desired error-sensitive properties are *only* accessible as far as the implementing environment is OO. Gothic is, in effect, a hybrid database that mixes some OO concepts alongside some specialised spatial database concepts. The result is some way from a fully OODBMS and consequently error-sensitive GIS properties are, by default, absent from the non-OO portions of the database: only objects have methods so only objects can support error-sensitive behaviour. Despite these setbacks, the Gothic database still proved broadly able to support the error-sensitive GIS object model following a number of modifications. Rather than allow Gothic to manage geometry and attributes in a non-OO manner, the implementation uses references as surrogate attribute and geometry objects, by-passing those supplied by default in the Gothic database. The **aspatial** and **errorSpatial** classes in figure 5.1 perform just this function, replacing Gothic's non-OO attribute and geometry data types and allowing geometry and attributes to participate in error handling functionality in the usual way.

The main disadvantages of this approach are high complexity and loss of flexibility. A large proportion of Gothic's GIS application and database functionality is written to handle Gothic attribute and geometry data types and *not* objects. The use of non-Gothic attribute and geometry objects illustrated in figure 5.1 in effect works by 'breaking' Gothic's core data model and replacing it with an OO data model. As a result, much of Gothic's core attribute and geometry functionality is also by-passed. It would be relatively straightforward to reprogram this functionality to use objects rather than Gothic's attribute and geometry data types. Unfortunately, given that Gothic's application programming interface (API) runs to some seven volumes, most of which depends to some extent upon at least one of the attribute or geometry data types, such a task was beyond the resources of this research. A secondary disadvantage of this approach is that the Gothic database only indexes objects according to their geometry. Objects without geometry can easily be 'lost' in the database, since Gothic offers few mechanisms for retrieving stored objects that have no geometry.

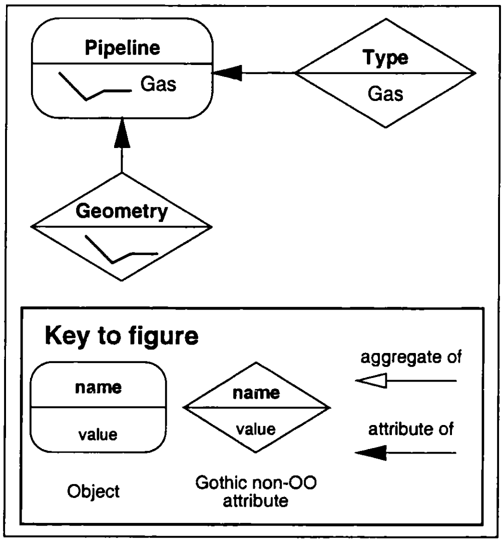


Figure 5.5: Gothic non-OO data model

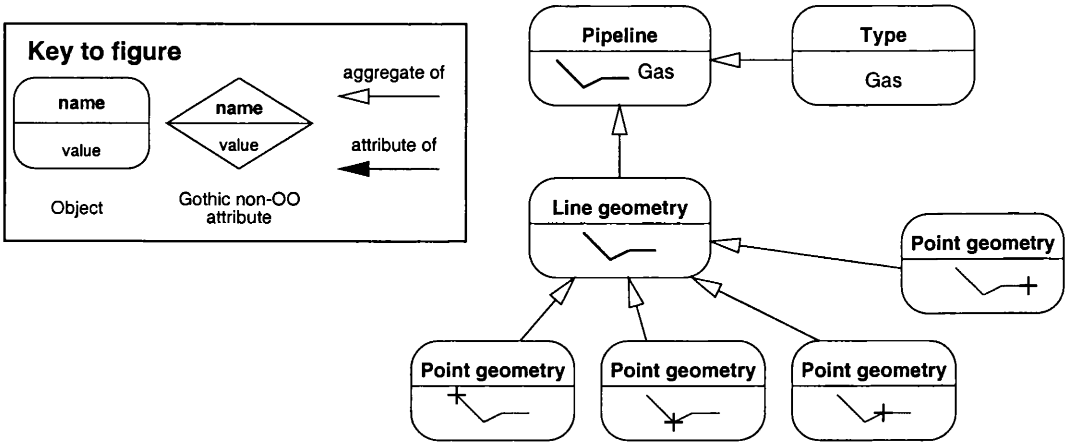


Figure 5.6: Object-oriented data model

Consequently, the error-sensitive GIS design shown in figure 5.1 does make one compromise to Gothic's limitations. Objects belonging to the **errorSpatial** class have non-OO geometry attributes (a consequence of inheriting from **simple**). This allows **errorSpatial** objects to be both error-sensitive and integrated with the full range of Gothic's spatial functionality. When required, an **errorSpatial** object can behave just like any ordinary error-sensitive database object, but in cases where Gothic's non-OO spatial functionality is needed the non-OO geometry attribute of an **errorSpatial** object can be used. The cost of this compromise is that while geometry objects offer error-sensitive behaviour, the spatial primitives from which these geometry are formed objects cannot participate in error-sensitive behaviour.

For example, the positional accuracy of line object, could be represented in the implemented error-sensitive Gothic database, but the positional accuracy of the individual vertices that make up that line could not. The situation is illustrated in figures 5.5–5.7. Figure 5.5 illustrates the normal Gothic data model, where a geospatial feature like a gas pipeline would be held in the database as an object, but the pipeline's geometry and attributes would be held as non-OO Gothic data types. As a result these non-OO geometry and attributes can have no quality information associated with them. In contrast, figure 5.6 illustrates an idealised OO data model, where everything is an object and quality information could be associated at any level of the aggregation hierarchy, including down to the level individual vertices. The compromise used to implement the Gothic error-sensitive GIS is illustrated in figure 5.7. Here geometry and attributes are objects, but geometry objects have non-OO geometry attributes. Any of the *objects* in figure 5.7 could have quality information associated with them, but non-OO geometry data types cannot support error-sensitive behaviour. The practical consequences of this compromise are explored in more detail in chapter 9.

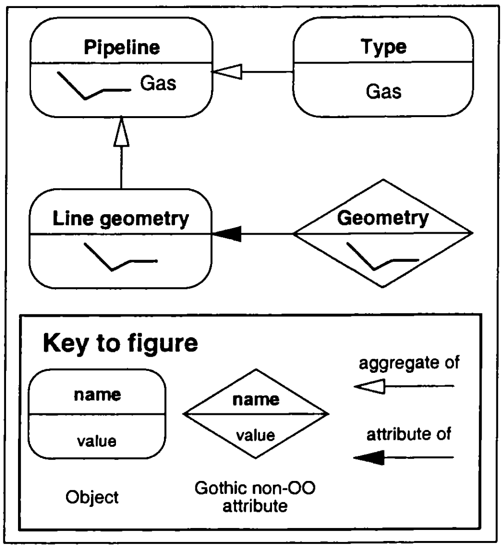


Figure 5.7: Compromise data model

5.3 Implementation performance: Any quality

Having addressed the inevitable implementation problems, the Gothic error-sensitive GIS did still perform much as hoped. In particular, the database was capable of supporting an extremely wide range of data quality elements. This section looks how the US, Canadian and European data quality standards were handled by the error-sensitive GIS. Finally, the section looks at a number

Quality element	Abstractive	Representative		Quality attributes
		Metric	Restricted	
Lineage	No	Not metric	Unrestricted	Process name Process description Process date Process date type
Positional accuracy	No	Spatial data	Yes	x-RMSE y-RMSE
Continuous attribute accuracy	No	Quantitative attributes	Yes	RMSE
Categorical attribute accuracy	No	Qualitative attributes	Yes	Probability correct
Accuracy test	No	Continuous and categorical accuracy	Unrestricted	Test type Test description Test date
Completeness test	No	Dependant on particular test	Unrestricted	Test description Test result Test date
Logical consistency	Yes	n/a	n/a	Is consistent
Completeness	Yes	n/a	n/a	Definition Selection criteria

Table 5.1: Example SDTS error-sensitive quality schema

of particular problem quality elements that the error-sensitive GIS might be called upon to handle, but which fall outside the usual standards-based discussion of data quality.

5.3.1 Spatial Data Transfer Standard

As already noted in §4.1.2.1, SDTS is undoubtedly one of the most influential and widely used standards in the world. The majority of national data transfer standards make significant use of SDTS or even, in the case of Australia, have adopted SDTS in its entirety (Moellering 1997). As a consequence, the five elements of spatial data quality highlighted by the NCDCDS, and enshrined in SDTS have found their way into many national and international data standards.

Support for the SDTS quality standard is, therefore, the starting point for testing any error-sensitive GIS implementation. Since SDTS is not an OO quality standard definition, it is necessary to convert the often relatively vague SDTS quality elements to a form suitable for use in the error-sensitive GIS. The process of conversion is essentially a mini-OOA of the standard, which aims to identify abstractive and representative quality elements, and the properties and attributes of those elements. This section outlines the key results of the process of implementing the SDTS data quality standard within the error-sensitive GIS, set out in table 5.1, based on the SDTS specifications set out in the US Geological Survey web site (US Geological Survey 1999c).

5.3.1.1 Lineage

Within SDTS, lineage records the entire process history of data including data source. It is important to be able to identify source for individual objects in the data, a sure sign that the SDTS lineage is a representative quality element. Consequently, lineage is modelled as a representative quality class with four quality attributes, shown in table 5.1. Two attributes are used to identify the name and description of a process that has operated upon an object or set of objects in the database. Two further attributes are used to describe when the process was operated. SDTS lineage objects can refer to any object in the database and so the class lineage is not metric. An individual object in the database may undergo many different processes, so the quality element lineage is not restricted, allowing many lineage objects to be associated with a single database object.

5.3.1.2 Positional accuracy

Like lineage, positional accuracy is a representative quality element since positional accuracy can apply to individual objects in the database. Unlike lineage, positional accuracy is only meaningful when associated with spatial objects in the database (rather than, say, attributes or other quality objects). Further, at most one positional accuracy object would usually be associated with an individual spatial object. Consequently, positional accuracy is a metric, restricted representative quality element. The key attributes of the positional accuracy class may depend upon the types of test performed to determine positional accuracy. Most commonly, the RMSE of position in the x and y direction are quoted as the parameters of positional accuracy, although other attributes are possible.

Additionally SDTS requires the type of the accuracy test performed to be documented (usually one of deductive estimate, internal evidence, comparison to source or independent source of higher accuracy). A description of the test should also be included along with a test date. Attributes describing test type, description and date could easily be included within the positional accuracy quality element definition. However, under the error-sensitive GIS architecture, a better approach is simply to define a separate 'accuracy test' meta-quality element that reports the details of a particular accuracy test, as in table 5.1. Such a representative quality element would be metric, as it could only meaningfully refer to accuracy quality elements such as positional accuracy, and unrestricted, since a single positional accuracy statistic might be the result of more than one actual accuracy test or assessment.

5.3.1.3 Attribute accuracy

Two separate types of attribute accuracy are defined in SDTS which in turn correspond to two separate attribute accuracy classes. Both will be only be meaningful when referring to objects of a particular metric and consequently will be metric representative quality elements. Continuous attribute accuracy in SDTS is very similar to positional accuracy. Consequently, in common with positional accuracy, an 'accuracy test' meta-quality object can be associated with a continuous attribute accuracy object. Categorical attribute accuracy, can be reported in SDTS in a number of ways, many of which are outlined in §2.2.2. CEM are the most sophisticated attribute accuracy statistic suggested by SDTS, but simpler measures, such as probability, or even subjective quality statements, such as "poor accuracy", are acceptable. As a result the detailed structure of categorical attribute accuracy objects may vary. In any event the 'accuracy test' class can again be used to report the details of any accuracy assessment leading to a categorical accuracy statistic.

5.3.1.4 Logical consistency

Logical consistency provides a check upon valid coding and topology in the data. As already mentioned, the concept of logical consistency operates at a class level, and so consistency is a abstractive quality element. By including a suitable quality attribute, such as the 'is consistent' method in table 5.1, in the **abstraction** class, a core logical consistency behaviour can be transmitted throughout the database. Most sub-classes will override the 'is consistent' behaviour with a method that reflects their own consistency needs, for example ensuring valid geocodes for attributes.

5.3.1.5 Completeness

The SDTS quality element completeness includes information on selection criteria, definitions, mapping rules and describes "the relationship between objects represented and the abstract universe of all such objects" (US Geological Survey 1999a). Under the analysis of data quality used here, completeness decomposes into a number of quality elements, some of which are abstractive and some of which are representative. Selection criteria, definitions and mapping rules are all abstractive quality elements since they are expected to apply equally to all members of a particular class. However, SDTS also encourages the use of taxonomic and exhaustive completeness, which reports the results of individual completeness tests upon groups of objects in the database. Such tests are best represented as representative quality objects as they refer to individual rather than classes of objects. Table 5.1 gives an example of how this situation might be implemented using two completeness elements: the abstractive quality element 'completeness' and the representative quality element 'completeness test'.

5.3.2 Spatial Archive and Interchange Format

Spatial Archive and Interchange Format (SAIF) was accepted as a Canadian national standard in 1993 and is arguably one of the more advanced standards in the world. SAIF is an object-oriented data format and consequently there is some commonality in approach between the SAIF format and the object model used here. It is worth noting that SAIF tackles meta-data in a much wider context than simply data quality. The SAIF meta-data class 'quality' is used to report accuracy and integrity issues, whilst lineage and source are treated as separate meta-data classes. However, SAIF also supports meta-data classes covering spatial and temporal reference systems, product description and general location (Geographic Data BC 1996). While the error-sensitive GIS design proposed here is geared towards managing data quality information, the dividing line between data quality and meta-data is blurred. Deciding what constitutes "traditional" data quality and what constitutes meta-data more generally within SAIF is inevitably somewhat arbitrary. Experience with using the error-sensitive GIS suggests that it could easily support the entire range of SAIF meta-data classes, including data quality.

Since both approaches are OO, the SAIF data quality classes can be implemented directly in the error-sensitive data model. Some changes may be desirable though: SAIF does not make the distinction between representative and abstractive quality element and treats all data quality in an analogous manner to representative quality. The standard SAIF schema provides no support for efficient data storage and generally does encourage quality to be accessed and used in a consistent way across the schema, providing limited support for defining restricted quality and no support for metric quality elements. Consequently, whilst implementing the SAIF 3.2 data quality schema (Geographic Data BC 1996) within the Gothic error-sensitive GIS, it made sense to use the additional error-sensitive functionality where possible. For example, relative and absolute positional and attribute accuracy are all elements of the SAIF data quality schema, but SAIF does not offer any mechanism for ensuring accuracy objects are associated with appropriate geospatial objects. Within the error-sensitive GIS it is a simple matter to declare each accuracy class as a metric representative quality class and prevent nonsensical use of quality, say, 'name' attributes being annotated with positional accuracy. In the same way, the SAIF quality class 'integrity' performs a similar function to SDTS logical consistency, and is consequently better represented as an abstractive quality element.

5.3.3 European Draft Standard CEN/TC 287

Though not a full European standard, the CEN/TC 287 draft quality standard is an important document as it looks set to be highly influential in the production of the data quality model in the forthcoming international standard, ISO 15046-13 (Godwin 1999). As mentioned previously, CEN/TC 287 is particularly interesting as it is the only existing standard which explicitly deals with meta-quality information. The discussion of SDTS in §5.3.1 revealed that implicit use of meta-quality already exists in popular data quality standards. However, CEN/TC 287 explicitly defines meta-quality elements that allow users to associate levels of confidence and reliability with alongside quality information (mentioned in §4.2.3).

The details of the standard can be found in the CEN/TC 287 draft quality standard documentation (CEN/TC287 1996). In addition to meta-quality information, CEN/TC 287 defines all the basic SDTS quality elements (lineage, positional and attribute accuracy, completeness and logical consistency) plus a number of new quality elements such as usage, temporal accuracy, and textual fidelity. CEN/TC 287 is organised along broadly OO lines, although it makes no explicit mention of OO. Consequently, in common with the SAIF standard discussed above, CEN/TC 287 was implemented directly, given a number of limited adaptation to take advantage of additional error-sensitive functionality.

5.3.4 Special and non-standardised quality elements

In addition to the three data quality standards, STDS, SAIF and CEN/TC287 discussed in the preceding sections, the error-sensitive GIS proved flexible enough to handle a very wide range of data quality elements. This section highlights a number of 'problem case' data quality elements that could be handled by the error-sensitive GIS, but which might prove more difficult to implement in a less flexible environment.

5.3.4.1 Complex representative quality objects

Since the only restriction placed upon the representative quality by the error-sensitive database is that it inherits from the class **representative element** and is composed of a number of **representative attribute** objects (see §4.1.2.3), the error-sensitive GIS offers a high degree of flexibility with regard to the quality elements that can be supported. All three data quality standards discussed above admit the use of the CEM as a measure of attribute accuracy. Matrix structures, such as a CEM, do not pose a difficulty to the error-sensitive GIS. However, many image formats are based on simple array, matrix or index structures. Consequently, the error-sensitive GIS should also be able to support data quality images. The use of image-based data quality elements is largely unexplored in the literature, but could allow scanned photographs or images of hard copy maps or plans to be included alongside other data quality elements as part of a data set's lineage or source. Image-based data quality elements were implemented during prototyping in Java, as Java provides considerable broad-based support for image handling, but not in the full Gothic implementation since Gothic is not flexible enough to support image objects without considerable additional reprogramming.

A logical progression from image-based data quality is to use geospatial-based data quality. The idea of 'quality maps' has been around for some time, and many traditional cartographic products contain small accuracy or reliability diagrams as part of their quality marginalia (Chrisman 1983) and is even mentioned within SDTS (US Geological Survey 1999c). Implementing geospatial-based data quality should be relatively trivial in any error-sensitive OOGIS, since hopefully any OOGIS will already contain sophisticated geospatial object support.

5.3.4.2 Method quality

In OO error-sensitive GIS dealing with the quality of method invocation results can be a tricky problem. A polygonal object, for example, may have a method that calculates its own area. The question arises that while positional accuracy may be known for the polygon, how should that accuracy information be propagated to the results of the polygon object's behaviours? There are in fact a number of ways to approach this problem and the most appropriate may differ depending on the method in question. One solution might be to have a shadow method that calculated, say, the accuracy of the area calculation. Anyone using the area calculation method could subsequently call the accuracy calculation method that could employ standard error propagation techniques upon the stored accuracy of each of the polygon vertices.

Another solution might be to associate quality with the results of the area method invocation directly. The area method will hopefully return an area object that may have a field detailing the actual magnitude of the area and perhaps the units of that area calculation. Assuming this area object is also part of the error-sensitive database, ie it inherits directly or indirectly from **uncertainty** and **abstraction**, it will be possible to associate quality with the resulting area object during the method invocation. The result of invoking the area method on the polygon object is then an area object that itself is annotated with an appropriate accuracy object.

In fact, the example application explored in chapter 9 uses neither of these approaches. The problem arises that in the case of error propagation there exists a wide range of methods that may be more or less suitable for particular situations. It is generally ill-advised to hardwire one method into an object. For example, both variance propagation and Monte-Carlo simulation may be appropriate methods for calculating the accuracy of the area of a polygon, dependent on a range of different factors. Consequently, error propagation in chapter 9 is dealt with by external, configurable applications rather than by the object itself.

5.3.4.3 Topology

Special mention needs to be made of topology. Topological consistency can be viewed as a sub-set of logical consistency and modelled as an abstractive quality element. Indeed, topological consistency is defined as a subset of logical consistency within SDTS (US Geological Survey 1999c). It would be entirely possible for the error-sensitive GIS to deal with topological consistency as an abstractive data quality element. In fact the results of this approach would probably not be very different from the way in which topology is handled in OOGIS anyway. Gothic defines topology as methods on each geospatial class (much as abstractive quality is defined as a method common to every object of a particular class) and uses these methods to manage and create topological relationships 'on-the-fly' (Laser-Scan 1996). However, topology was treated as outside the scope of this research. The reason for this, at first sight puzzling, omission is that topology is already very well established and supported by most GIS, far more so than any other area of data quality.

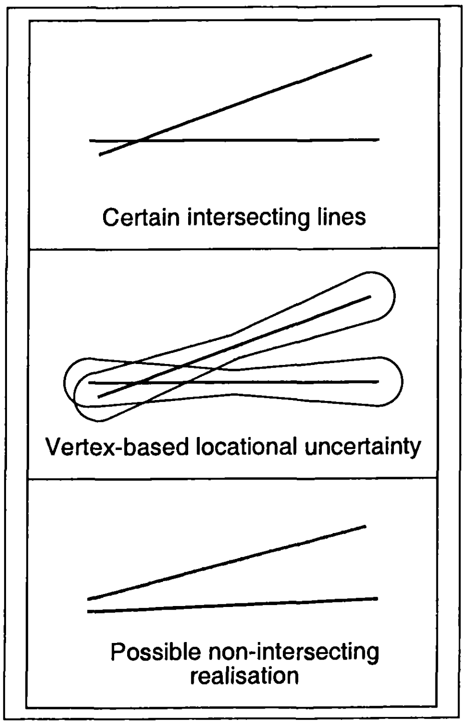

Certain intersecting lines
Vertex-based locational uncertainty
Possible non-intersecting realisation

Figure 5.8: Topological uncertainty

While topological consistency enjoys widespread use and understanding, the issue of the quality of topology is much less clear. Topology can be viewed either as a constraint on geometry or derivative of geometry. Topological information as a constraint on geometry is potentially very useful (see for example Flewelling et al. 1992). In practice qualitative spatial information is rarely collected and correspondingly there have been few attempts to model the uncertainty associated with such information (but see Cohn and Gotts 1996). In contemporary GIS, topology is usually derived from the stored geometry of geospatial objects. If that geometry is uncertain the derived topology will also be uncertain. The nature of that uncertainty will be dependent on the geometric and topological models used. For example, figure 5.8 shows two intersecting certain straight lines with unambiguous topology. However, by introducing uncertainty into the line location introduces concomitant uncertainty into the topology. In the example in figure 5.8 the use of a vertex-based locational error model can introduce uncertainty not simply into the location of the intersection, but place a question mark over the existence of an intersection at all. A few of models of topological uncertainty are now emerging, in particular based on the egg-yolk representation of uncertain spatial objects (Shi and Guo 1999). While this is a crucial area of future research with wide reaching implications for GIS, both topological consistency and topological uncertainty are considered outside the scope of this research which focuses exclusively on non-topological uncertainty.

5.4 Example error-sensitive object system

A discussion of the practical application of error-sensitive GIS software is deferred until after the introduction of the component error-aware architecture in chapter 6. However, this section attempts to solidify the concepts introduced over the past three chapters by exploring an example of the error-sensitive functionality in operation. The example uses what might be considered a third error-sensitive implementation: the ς-calculus error-sensitive object system in appendix A.3. The untyped ς-calculus object system in appendix A.3 uses the minimum of terms necessary to provide basic error-sensitive functionality. This object system represents the core formal specification of an error-sensitive GIS. In common with the simple object system in §4.2.4.1 we assume the existence of *true* and *false* objects in addition to an *if-then-else* construct. For simplicity, the object system also assumes the existence of integer and floating point number objects as well as string objects, which are ordered lists of alphanumeric characters. To highlight the fact that integers, floating point numbers and strings are still objects, they are written enclosed in square braces, eg [1], [2.0] and ["three"] respectively. Strings can be compared using the equivalence ($\equiv$) to yield a *true* or *false* object (eg ["abc"] $\equiv$ ["bac"] reduces to *false*) while integer and floating point objects can be compared using equivalence and greater-than or smaller-than operators (eg [1.0] $<$ [2.4] reduces to *true*). None of these assumptions are elaborated upon here, except to say that pointers to the calculus needed to build these simple constructs from first principles can be found in previous chapters and in Abadi and Cardelli (1996a). The use of Boolean, integer, floating point and string objects and there associated operators are the only deviations from pure untyped ς-calculus.

The object system presented below is used to provide a 'walk-through' of each of the core error-sensitive properties in turn (abstractive, representative, metric and restricted quality) using a few objects taken from the telecommunications database introduced in full in chapter 7. For simplicity, the object system presented here does not attempt to implement the efficient quality storage model already explored in detail in §4.2.4.1. Rather than try to explain the detailed working of each object, the simplest way to the introduce the object system is to show it in operation.

5.4.1 Database design

The object system in appendix A.3 uses the classes *Unc* and *Rep* to implement the classes **uncertainty** and **representative element** from figure 4.1 respectively. A new class *List* is also introduced and performs many of the important quality list management functions. The most important class missing from the error-sensitive object system is **abstraction**. The reason for this omission is that the definition of the class **abstraction** is dependent upon the quality schema adopted for a particular data set. The first job of the error-sensitive GIS database designer, then, is to decide both upon the geospatial object schema and the quality object schema that will be supported by the error-sensitive database. Assuming in this simplified example only the basic elements from SDTS outlined in §5.3.1 are to be used, the definition of a new **abstraction** class, *Abs*, with one logical consistency method, *is_cons*, is given in equation 5.1.

$$Abs \triangleq [new = \varsigma(z)[is\_cons = \varsigma(s)z.is\_cons(s)], is\_cons = \lambda(s)true] \tag{5.1}$$

The next step is to define the geospatial classes to be used. In this example the location and attributes of telegraph poles used to route overhead telecommunication cables are the only classes supported in the ς-calculus 'database'. Equations 5.3 and 5.4 give the ς-calculus terms for a telegraph pole class, *Pole*, with one attribute, *Height*, and point geometry given by the class *Point*. For brevity, these classes introduce a new syntax using the keyword *extends*. The formal definition of extends is given in equation 5.2. Informally, writing *Pole* extends *Abs*, *Unc* means that the class *Pole* inherits all the methods from *Abs* and *Unc* without the need to explicitly rewrite the numerous methods in *Abs* and *Unc* in full in the definition of *Pole*.

$$\begin{gathered}\frac{\vdash a \text{ extends } a'_1,..,a'_j \triangleq [new = \varsigma(z)[l_i = \varsigma(s)z.l_i(s)], l_i = \lambda(s)b_i]}{\vdash a \triangleq [new = \varsigma(z)[l_i = \varsigma(s)z.l_i(s), l_{jk} = \varsigma(s)z.l_{jk}(s)], l_i = \lambda(s)b_i, l_{jk} = \lambda(s)b_{jk}]}\\ \text{where } a'_j \triangleq [l_{jk} = \varsigma(s)b_{jk}] \text{ and } i \in 1..n,\ j \in 1..m,\ k \in 1..M_j\end{gathered} \tag{5.2}$$

$$\begin{gathered}\mathit{Pole} \text{ extends } \mathit{Abs}, \mathit{Unc} \triangleq [new = \varsigma(z)[point = \varsigma(s)z.point(s),\ height = \varsigma(s)z.height(s)],\\ point = \lambda(s)\mathit{Point}.new,\ height = \lambda(s)\mathit{Height}.new]\end{gathered} \tag{5.3}$$

$$\begin{gathered}\mathit{Point} \text{ extends } \mathit{Abs}, \mathit{Unc} \triangleq [new = \varsigma(z)[\varsigma(s)z.x(s),\ \varsigma(s)z.y(s)],\\ x = \lambda(s)[0.0],\ y = \lambda(s)[0.0]]\end{gathered} \tag{5.4}$$

$$\begin{gathered}\mathit{Height} \text{ extends } \mathit{Abs}, \mathit{Unc} \triangleq [new = \varsigma(z)[\varsigma(s)z.val(s),\ \varsigma(s)z.units(s)],\\ val = \lambda(s)[0.0],\ units = \lambda(s)[''\mathrm{m}'']]\end{gathered} \tag{5.5}$$

Finally, the representative quality classes positional accuracy, *Pos*, and lineage, *Lin*, along with the meta-quality class continuous accuracy test, *Test*, are defined in equations 5.6–5.10.

$$\begin{gathered}\mathit{Pos} \text{ extends } \mathit{Rep} \triangleq [new = \varsigma(z)[rmse = \varsigma(s)z.rmse(s),\ name = \varsigma(s)[''\mathrm{Pos}'']],\\ rmse = \lambda(s)\mathit{RMSE}.new]\end{gathered} \tag{5.6}$$

$$\begin{gathered}\mathit{RMSE} \text{ extends } \mathit{Unc} \triangleq [new = \varsigma(z)[\varsigma(s)z.val(s)],\\ val = \lambda(s)[0.0]]\end{gathered} \tag{5.7}$$

$$\begin{gathered}\mathit{Lin} \text{ extends } \mathit{Rep} \triangleq [new = \varsigma(z)[desc = \varsigma(s)z.desc(s), name = \varsigma(s)[''\mathrm{Lin}'']],\\ desc = \lambda(s)\,\mathit{Word}.new]\end{gathered} \tag{5.8}$$

$$\begin{gathered}\mathit{Test} \text{ extends } \mathit{Rep} \triangleq [new = \varsigma(z)[desc = \varsigma(s)z.desc(s),\ name = \varsigma(s)[''\mathrm{Test}'']]\\ desc = \lambda(s)\,\mathit{Word}.new]\end{gathered} \tag{5.9}$$

$$\begin{aligned} \textit{Word} \text{ extends } \textit{Unc} \triangleq [&\textit{new} = \varsigma(z)[\varsigma(s)z.\textit{val}(s)], \\ &\textit{val} = \lambda(s)[\text{""}]] \end{aligned} \tag{5.10}$$

The definition of these ς-calculus terms mirrors the database design process that precedes the use of any error-sensitive GIS. The error-sensitive GIS provides only enough class definitions to support the core error-sensitive functionality. The choice of both geospatial and quality schema needed for a particular application is a matter left to the database designer.

5.4.2 Abstractive quality

Abstractive quality operates solely at the class level. However, abstractive quality behaviours will usually be redefined for each new sub-class of the abstractive quality super-class **abstraction**. In equation 5.1, invocation of the is_cons method will always reduce to 'true' ($Abs.new.is\_cons \rightarrowtail true$) indicating that any sub-class of Abs will always by default be consistent. More sophisticated consistency behaviour can be obtained by overriding the body of the is_cons method in sub-classes of Abs. For example, the *Height* attribute of *Pole* is only valid as long as it lies somewhere between 8m and 12m inclusive. This logical consistency information can be added to the quality schema by updating the body of the is_cons method in *Height'* with a new method body that checks the for valid pole heights before reporting on consistency, as in equation 5.11.

$$\begin{aligned} \textit{Height}' \text{ extends } \textit{Abs}, \textit{Unc} \triangleq [&\textit{new} = \varsigma(z)[\varsigma(s)z.\textit{val}(s),\ \varsigma(s)z.\textit{units}(s)].\textit{is\_cons} \Leftarrow \\ &\varsigma(s)[\underline{\text{if}}\ s.\textit{val} > [12.0]\ \underline{\text{then}}\ \textit{false} \\ &\underline{\text{else}}\ \underline{\text{if}}\ s.\textit{val} < [8.0]\ \underline{\text{then}}\ \textit{false}\ \underline{\text{else}}\ \textit{true}], \\ &\textit{val} = \lambda(s)[0.0],\ \textit{units} = \lambda(s)[\text{"m"}]] \end{aligned} \tag{5.11}$$

It is now possible for a *Height'* object to report upon its own consistency. In equation 5.12, a new height object is created, which according to the definition of *Height'* in equation 5.11 has a default value of [0.0]. As a result, invocation of the is_cons method upon this new height method reduces to *false* and leads to the conclusion that the object is not logically consistent. In contrast, the new height object in equation 5.13 is first updated with a new value of 9m before the is_cons method is invoked, reducing to *true*.

$$(\textit{Height}'.\textit{new}).\textit{is\_cons} \rightarrowtail \textit{false} \tag{5.12}$$

$$((\mathit{Height}'.\mathit{new}).\mathit{val} \Leftarrow \varsigma(s)[9.0]).\mathit{is\_cons} \rightarrowtail \mathit{true} \tag{5.13}$$

A slightly more sophisticated use of this approach, illustrated in equation 5.13, redefines the *Pole* class in equation 5.3 to check whether the pole's height and geometry are consistent before reporting on its own consistency.

$$\begin{gathered}\mathit{Pole}' \text{ extends } \mathit{Abs}, \mathit{Unc} \triangleq [\mathit{new} = \varsigma(z)[\mathit{point} = \varsigma(s)z.\mathit{point}(s), \mathit{height} = \varsigma(s)z.\mathit{height}(s)].\mathit{is\_cons} \Leftarrow \\ \varsigma(s)\underline{\mathrm{if}}\ s.\mathit{height}.\mathit{is\_cons}\ \underline{\mathrm{then}}\ s.\mathit{point}.\mathit{is\_cons}\ \underline{\mathrm{else}}\ \mathit{false}], \\ \mathit{point} = \lambda(s)\mathit{Point}.\mathit{new},\ \mathit{height} = \lambda(s)\mathit{Height}.\mathit{new}]\end{gathered} \tag{5.14}$$

In turn, this allows the consistency of new *Pole′* objects to be checked as in equation 5.15 below.

$$((\mathit{Pole}'.\mathit{new}).\mathit{height}.\mathit{val} \Leftarrow \varsigma(s)[13.0]).\mathit{is\_cons} \rightarrowtail \mathit{false} \tag{5.15}$$

It is important to note that the *is_cons* method only *reports* upon logical consistency, it does not *enforce* logical consistency. Fitness for use only aims to supply enough information to allow a user to come to a reasoned decision about the fitness of a particular data set for a particular use. The discussion in §4.1.2.2 pointed out that in many cases it may be more effective to include logical consistency behaviour as an encapsulated method within the access methods of a class to enforce consistency. Increasing use and familiarity with OO should mean logical consistency becomes much a less important quality element in the future. However, the goal of error-sensitive GIS remains reporting rather than enforcing quality.

5.4.3 Representative quality

Individual geospatial objects can be annotated with individual representative quality objects using the *set_rep* method in *Unc* and all its sub-classes. For example, when adding a new *Pole* object to the database, it might be desirable to set not simply the geometry and height for that pole object, but additionally to annotate the new *Pole* with a lineage object. The ς-calculus terms in equation 5.16 below create a new pole object with a height of 9.0m at coordinate (10.0,10.0) and annotate this object with a new lineage object that contains information about the creation process, in this case very basic information about the creation date.

$$
\begin{aligned}
&p \triangleq Pole'.new \\
&p.height.val \Leftarrow [9.0] \\
&p.point.x \Leftarrow [10.0] \\
&p.point.y \Leftarrow [10.0] \\
&p.set\_rep((Lineage.new).desc \Leftarrow [\text{"create on 1999-07-03"}])
\end{aligned}
\tag{5.16}
$$

The *get_rep* method in *Unc* allows the interrogation of individual error-sensitive objects regarding their representative quality. The *get_rep* method requires the name of a quality class as an argument and returns a *List* object populated with all quality objects of that named class associated with the interrogated database object. For the pole p in equation 5.16, the *get_rep* method can be used to retrieve a list containing the original 'create' lineage object as the first and only element, as in equation 5.17.

$$(p.get\_rep([\text{"Lin"}])).get([1]) \rightarrowtail [\text{"create on 1999-07-03"}] \tag{5.17}$$

5.4.4 Metric quality

The definition of the *Rep* class in appendix A.3 is for quality that is not metric, ie can be associated with any geospatial or quality object. This type of behaviour, indicated by the *is_met* (is metric) method in *Rep* which always returns false, is appropriate for quality objects like the lineage object in the previous example, equation 5.17. In contrast, objects belonging to the positional accuracy class, *Pos*, should only be able to refer to spatial objects in the database, in this case *Point* objects. This behaviour is achieved in a two step process, first by modifying the definition of *Pos* class to ensure that it is metric, and subsequently notifying the *Point* class of this change, as in equations 5.18 and 5.19 respectively.

$$Pos' \triangleq Pos.is\_met \Leftarrow true \tag{5.18}$$

$$Point.mlist.add([\text{"}Pos\text{"}]) \tag{5.19}$$

The *Unc* class organises *Rep* objects according to their *name* field. Each quality class redefines *name* to be a unique string identifier for that class. When attempting to set the representative quality of an object, if the supplied quality object is metric the *set_rep* method checks that the quality object's class name appears in the list of allowable metric quality classes for that object.

Any attempt to set the positional accuracy of an object in the database will only be successful as long as the database object belongs to a class that allows annotation with positional accuracy classes, in this case *Point*. The term in equation 5.20 creates a new pole object and attempts to set the representative quality of the pole's geometry with a positional accuracy object of class *Pos'*. When attempting to retrieve the positional accuracy of the pole's point geometry using the *get_rep* method, we know the attempt was successful, since the *size* method of the resultant quality list object reduces to 1.

$$((Pole'.new).point.set\_rep(Pos'.new.rmse \Leftarrow [0.3])).get\_rep(["\mathrm{Pos}"]).size \rightarrowtail [1] \tag{5.20}$$

Conversely, equation 5.21 attempts to set the positional accuracy not of the pole's geometry, but of the pole itself. Since the class *Pole* has not been permitted to refer to *Pos* objects the attempt fails, indicated by the zero size of the list object returned by the appropriate *get_rep* invocation.

$$((Pole'.new).set\_rep(Pos'.new.rmse \Leftarrow [0.3])).get\_rep(["\mathrm{Pos}"]).size \rightarrowtail [0] \tag{5.21}$$

5.4.5 Restricted quality

In addition to being metric, positional accuracy objects will usually be restricted, in that a database object can be annotated with at most one positional accuracy object. Equation 5.22 defines a new class *Pos''* based on *Pos'* in equation 5.18, that is both metric and restricted.

$$Pos'' \triangleq Pos'.is\_res \Leftarrow true \tag{5.22}$$

The effect of this change can be seen when attempting to update a new point object with more than one positional accuracy object, as in equation 5.23. The first update is accepted, in exactly the same way as for equation 5.20, annotating the point object *p* with an RMSE of 0.3.

$$\begin{aligned} &p \triangleq ((Point.new).set\_rep(Pos''.new.rmse \Leftarrow [0.3])) \\ &p.get\_rep(["\mathrm{Pos}"]).size \rightarrowtail [1] \\ &p.get\_rep(["\mathrm{Pos}"]).get([1]).rmse.val \rightarrowtail [0.3] \end{aligned} \tag{5.23}$$

While subsequent updates are accepted they replace preceding updates. In equation 5.24, invoking the *set_rep* method on *p* with a new positional accuracy RMSE of 0.9 replaces the previous 0.3 RMSE accuracy object.

$$\begin{aligned}
&p.set\_rep(Pos''.new.rmse \Leftarrow [0.9]) \\
&p.get\_rep([\text{"Pos"}]).size \rightarrowtail [1] \\
&p.get\_rep([\text{"Pos"}]).get(1).rmse.val \rightarrowtail [0.9]
\end{aligned} \tag{5.24}$$

In contrast, the same process with objects from the unrestricted lineage class *Lin* repeatedly adds to the list of Lineage objects in equation 5.25 below.

$$\begin{aligned}
&p.set\_rep(Lin.new.desc \Leftarrow [\text{"create on 1999-07-03"}]).get\_rep([\text{"Lin"}]).size \rightarrowtail [1] \\
&p.set\_rep(Lin.new.desc \Leftarrow [\text{"update on 1999-07-04"}]).get\_rep([\text{"Lin"}]).size \rightarrowtail [2] \\
&p.set\_rep(Lin.new.desc \Leftarrow [\text{"update on 1999-07-05"}]).get\_rep([\text{"Lin"}]).size \rightarrowtail [3] \\
&\qquad\qquad\qquad \ldots
\end{aligned} \tag{5.25}$$

5.4.6 Meta-quality

Since both geospatial objects and representative quality objects inherit from the uncertainty class *Unc*, meta-quality behaviour for representative quality objects is achieved in exactly the same way as representative quality behaviour is achieved for geospatial objects. Equation 5.26 redefines the continuous attribute accuracy class, *Test*, to ensure it is a metric quality class. Following the modification of the positional accuracy class in equation 5.27 to reflect this change, meta-quality can be used using exactly the same mechanisms as operate for standard representative quality behaviours. Equation 5.28 gives a relatively complex ς-calculus term which adds a continuous accuracy test object to a new positional accuracy object, which in turn is added to the geometry of a new pole object.

$$Test' \triangleq Test.is\_met \Leftarrow true \tag{5.26}$$

$$Pos''.mlist \Leftarrow List.new.add([\text{"}Test\text{"}]) \tag{5.27}$$

$$
\begin{aligned}
&p \triangleq \mathit{Pole.new}\\
&\mathit{p.size.val} \Leftarrow [9.0]\\
&\mathit{p.point.x} \Leftarrow [10.0]\\
&\mathit{p.point.y} \Leftarrow [10.0]\\
&\mathit{p.set\_rep}((\mathit{Lin.new}).\mathit{desc} \Leftarrow [\text{"create"}])\\
&\mathit{p.set\_rep}((\mathit{Pos.new}).\mathit{rmse} \Leftarrow [0.9])\\
&\mathit{p.get\_rep}([\text{"}\mathit{Pos}\text{"}]).\mathit{get}([1]).\mathit{set\_rep}((\mathit{Test.new}).\mathit{desc} \Leftarrow [\text{"deductive estimate"}])
\end{aligned}
\tag{5.28}
$$

5.5 Conclusions

The goal of this chapter was to explore to what extent the error-sensitive GIS theory presented in chapter 4 was able to associate any quality with any object in any OODBMS. Broadly speaking, the implementation results seem to support each of these aims. Despite clear contrasts in the implementation process and architecture between the Java prototype and the full Gothic implementation, the existence of two separate OO implementations does seem to suggest that the analysis results are portable to differing OO environments or even OODBMS. Further, the two implementations do indeed appear to allow quality to be associated with practically any object in the database, including other quality objects. Finally, the range of quality elements that can be supported by the error-sensitive GIS is certainly wider than found in most data quality standards, and is not limited to simple, well-behaved quality elements.

5.5.1 Laser-Scan Gothic OOGIS

The results of the Laser-Scan Gothic implementation, in §5.2, highlighted the most important limitation of the approach taken by this research: the error-sensitive GIS can only operate properly within fully OO environments. Throughout the error-sensitive GIS analysis, design and programming process, the attempt has been made to strike a compromise between building a general system that will work with any GIS and a specific system that is powerful enough to make a significant contribution to GIS error handling capabilities. A key element of this compromise was the use of OO as a development paradigm. Chapter 4 showed that by restricting the discussion to OO environments, powerful error-sensitive functionality could be obtained. At the same time chapter 3 argued that while OOGIS are currently a minority technology in terms of commercial GIS, OO offers so many advantages to GIS over relational technology, that relational GIS can increasingly be viewed as legacy systems that will over time be largely superseded by OOGIS.

The failure of the Laser-Scan Gothic implementation to support the error-sensitive architecture without significant modification (§5.2) reveals a chink in this armour. While commercial GIS may claim to be OO, the term is often used very loosely to denote hybrid or even nominally OO systems. In the future, this may be less of a problem. Gothic was designed and written more than a decade ago at a time when OO was less clearly defined and understood than it is now. Both theoretical innovations, like ς-calculus, and practical innovations, like Java, are encouraging convergence in OO concepts. This convergence should lead to a concomitant technological convergence, necessary if commercial GIS technology is to incorporate OO concepts to the same degree as in other branches of IS.

The error-sensitive GIS development process highlighted where current OOGIS technology, such as Gothic, may fall short of being truly OO. The Gothic database depends in large part upon specialised non-OO data types and offers narrow database functionality geared exclusively to dealing with spatially referenced data. Hopefully the next generation of OOGIS will be able to address such problems and offer more flexible, fully OO databases. There is already some evidence of such a trend. A variety of ongoing initiatives have grown up recently which already seem to be yielding some practical commercial and research results, such as the use of Java as the basis for an OOGIS (see for example Professional Geo Systems 1999, OpenMap Java GIS from GTE Interworking 1999, and DESCARTES Java GIS visualisation, Andrienko and Andrienko 1999).

5.5.2 Example ς-calculus object system

The use of ς-calculus in the example implementation (§5.4) is important as it provides a blueprint of the minimum error-sensitive functionality in the face of uncertainty surrounding the credibility of current OOGIS technology. By using simple ς-calculus systems it is possible to construct formal software models that can help elucidate precisely how an OO software system should work. Using such formal models it is then possible to make general statements about the object systems being studied, and provide object system specifications independent of the peculiarities of particular software.

The ς-calculus does seem to be a potentially useful tool in OOGIS development generally. It allows the formulation of rigorous, object-based formalisms that both specify and allow exploration of the resulting object systems, features of formalisms championed by Frank and Kuhn (1995). Further, the example in this chapter does illustrate the value of untyped ς-calculus object systems, in the future the use of the far more powerful typed ς-calculus may prove even more valuable. However, the results are undoubtedly complex at times, and there is always the danger of 'overformalising' (Bowen and Hinchey 1995) — putting unnecessary emphasis and effort in the production formal systems. The production of a high-level formalism based on ς-calculus, touched upon in 3.4.3, would certainly lessen this danger. In the case of the development of an error-sensitive GIS, the existence of ς-calculus has certainly proved beneficial by focusing attention on the fundamental properties of error-sensitive object systems rather than the somewhat capricious details of OOGIS database programming.

Chapter 6

Error-aware GIS: component architecture

The error-sensitive GIS, presented in the preceding chapters, offers the core functionality necessary to store and manage the quality associated with geospatial information. It allows data quality to be accessed through a consistent interface and defines the minimum set of features needed to model data quality adequately. However, the error-sensitive GIS is relatively intricate and involved; it stops well short of assisting in the understanding and use of data quality information. For an error-sensitive GIS to be effective, there is a clear need to provide tools that can help error-sensitive GIS users to better understand and apply data quality information.

The concept of an error-aware GIS, introduced in §1.3.2, aims to bridge this gap between core error-sensitive functionality and the practical use and understanding of data quality information through the deployment of domain specific and intelligent technology. In attempting to marry these two extremes of the application spectrum, the flexibility of an error-sensitive database and the specificity of error-aware tools, there exist both hazards and opportunities. This chapter aims to navigate these hazards and opportunities and to chart the development of a component software architecture suitable for connecting error-sensitive and error-aware software.

6.1 Error-aware GIS: a challenge and an opportunity

The aim of an error-aware GIS, as stated above, is to extend error-sensitive functionality through the use of domain specific and intelligent software. The implication is that error-aware software developed for one application is unlikely to be suitable for other application areas. Natural resource management applications, for example, may demand software designed to aid users in visualising the uncertainty associated with the classification, indeterminate boundaries and subpixel mixing of land parcels (Bastin et al. 1999). In contrast the features in a telecommunications GIS are relatively unambiguous and the visualisation of classification accuracy and indeterminacy may be a low priority. Instead positional, geometric and topological accuracy may be far more important to telecommunications and utilities applications than to natural resource management (Russomanno 1998). While there may be some commonality across application areas, any attempt to develop an error-aware software panacea capable of dealing with the needs of every user in every application area is doomed to failure. Therefore, the error-aware GIS architecture must be capable of supporting any number of component software tools designed to address the specific needs of a given application area. The challenge is to provide an architecture able to allow rapid simple error-aware software development that can be closely integrated with error-sensitive functionality. At the same time if the hard-won flexibility of the error-sensitive GIS is to be retained, the architecture needs to offer a clear distinction between error-sensitive and error-aware GIS.

In meeting the challenge of developing a closely integrated yet flexible error-aware GIS architecture, a significant opportunity presents itself. The need to integrate flexible user interface software with core database functionality is not unique to error-aware GIS development. There is a general movement within GIS and IS development toward *open* architectures where spatial data can be accessed and analysed by different computers over a network using "vendor-neutral" computing standards (Sondheim et al. 1999, p347). By using an open GIS architecture, the error-aware GIS should be able to enjoy not only improvements in system design, but additionally take advantage of other opportunities offered by open GIS, namely the ability to access and share heterogeneous geographic information across a network, termed *interoperability*. Interoperability is particularly important to geospatial information which is by its very nature more complex and more expensive to collect and maintain than most other types of data (Frank and Kuhn 1995). As an indication of the need for greater sharing of geographic information, the OGC reports that the US government alone spends $4bn yearly on spatial data conversion (Open GIS Consortium Technical Committee 1999a). Without closer integration of geospatial datasets, use of geospatial data cannot take advantage of economies of scale (Frank and Kuhn 1995) and arguably the full potential of GIS will never be realised (Flowerdew 1991). In order to allow interoperability, an open GIS must offer two key elements. At a technical level, there is a need to allow component software and systems to communicate across a network. At a conceptual level there is a need to develop shared, standard data models. Both these elements are discussed in the following section.

6.2 Distributed component architecture

The two distinct problems facing any interoperable database alluded to above are usually termed *syntactic* and *semantic* heterogeneity (Vckovski 1998). Syntactic heterogeneity concerns the largely technical problems of interoperable computer systems and software, while semantic heterogeneity is concerned with the largely conceptual problems of interoperable data models. These two problems demand very different solutions, reflected by the OGC twin track approach to open GIS which defines both the Open Geodata Model (OGM) to combat semantic heterogeneity and the Services Architecture to deal with syntactic heterogeneity (Open GIS Consortium Technical Committee 1999b).

Semantic heterogeneity occurs when the definitions, interpretations, classifications or measurements used in related data sets do not agree, and is a significant and challenging research subject in its own right (Goodchild and Longley 1999). In the case of integrating error-sensitive databases, however, the problem may be greatly reduced. As long as the interoperating error-sensitive GIS are all based on a the same core model of data quality explored in chapters 4 and 5, there will already exist some semantic commonality between the quality information in any error-sensitive GIS. There is some circularity in this argument: as long as everyone uses the same data model semantic heterogeneity is *never* a problem. Nevertheless, the error-sensitive data model set out in this thesis is highly flexible: as discussed in §4.1.2.3 there are very few restrictions placed on the types of geographic information and data quality that can be supported while §5.3 indicated that the most commonly used data quality standards can be supported by the error-sensitive GIS. Therefore, while other error-sensitive data models are possible, it seems reasonable to suggest that the inbuilt flexibility of the error-sensitive data model presented here makes it likely that other error-sensitive data models and standards will be semantically congruent if not homogeneous. Semantic heterogeneity issues may still arise, even between databases that use the core error-sensitive data model presented in this thesis. For example, different error-sensitive databases may contain *homonymous* quality classes (semantically different classes with the same name, Vckovski 1998). However, the basic error-sensitive definitions of abstractive, representative, metric and restricted quality will be common to all error-sensitive GIS that follow the basic error-sensitive data model set out previously.

From the point of view of implementing an open error-aware GIS architecture, then, it is the management of syntactic rather than semantic heterogeneity that will need to occupy the remainder of this section. Syntactic heterogeneity concerns the practical difficulties facing interoperable systems: the need to provide a powerful platform neutral interface between applications and databases without compromising the flexibility of either.

6.2.1 Client/server systems

The discussion in §6.1 highlighted some of the reasons for wanting to keep error-aware applications separate from the error-sensitive database. This desire to separate applications from data is part of a general movement in GIS, and IS more generally, away from monolithic IS toward *distributed computing* (Sondheim et al. 1999). Distributed computing is a general term that is used to denote computing systems where processing tasks and data that are distributed across a network can be accessed in a relatively transparent way (Coleman 1999).

The most common distributed computing architectures for more than a decade now have been based on the client/server model, illustrated in figure 6.1. The idea behind the client/server model is to provide a clear delineation between the responsibilities of different computer systems

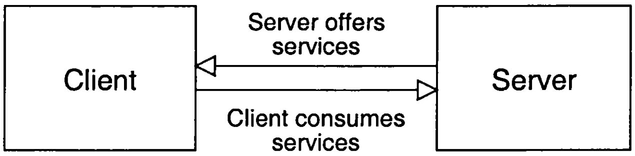


Figure 6.1: Basic client/server architecture

on a network. Computer systems that can offer services to other computers on the network are termed *servers* while computer systems that consume these services are termed *clients*. A client will request a service from a server, which will then process the request and respond to the client with the result of the requested process. The client/server model, when applied to GIS, allows spatial databases to deal exclusively with the problems of basic spatial data storage, management and processing. If the spatial database can offer these services across the network, client GIS applications can be developed to meet specific GIS user and application needs built upon this basic GIS functionality.

Servers make the services they can offer known to clients through the metaphor of a *contract* (Meyer 1992). Crucially, this contract, termed a *client interface* (Adler 1995), defines *what* services are provided by a server but not *how* they are provided. Clients can be built to take advantage of particular services offered by a server, since the contract metaphor acts as a guarantee that a server will always offer those particular services. However, since the details of how those services are supplied is hidden from the client, the server can be modified, upgraded or even replaced as long as the services defined in the client interface remain unchanged.

6.2.2 Multi-tier distributed systems

The client/server model has proved a major step forward in IS architecture. Not only does it allow networked access to services but it encourages much improved system architecture where client applications are both integrated with and isolated from the server's underlying data infrastructure. There are any number of different configurations based on the basic client/server concept, many of which are enumerated by Evans et al. (1995). However, an important drawback of any basic "two-tier" client/server architecture is the focus on request and response. It is often the case that clients actually need to request multiple services, perhaps across multiple servers or coordinating the various responses (Adler 1995). As a result, recent years have seen the increasing popularity of multi-tier client/server architectures also termed *component* architecture, where a client requesting a service from a server may itself act as a server offering services to other clients. Multi-tier client/server architectures blur the distinctions between the client and server rôles, since component software can both supply and consume services. The simple three-tier architecture in figure 6.2 illustrates the concept as the middle-tier performs both server and client functions. However, different tiers still retain rigid client interfaces defining exactly what services can be provided by a given server. As a result multi-tier client/server architectures can provide even greater software robustness and flexibility than two-tier client/server architectures, enabling distinct processing tasks to be performed by individual client/server components.

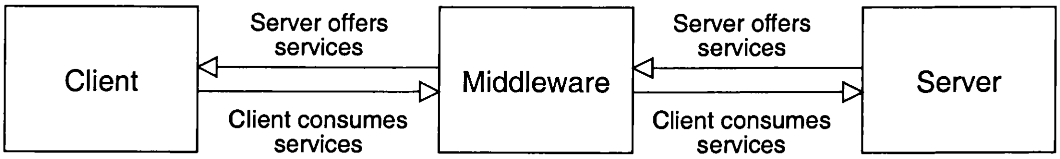


Figure 6.2: Multi-tier client/server architecture

One final innovation is required to allow distributed systems to address fully the problem of syntactic heterogeneity. By dedicating a middle-tier in a multi-tier system to the task of mediating standardised communication between the lower and upper tiers it is possible to construct an open system that allows open access for any number of client applications to any number of servers. This middle-tier, often termed *middleware* (Evans et al. 1995), is currently the focus of high levels of research and commercial interest. The object management group (OMG), a commercial consortium founded in 1989 to promote interoperable open systems, has established the leading middleware architecture: the *common object request broker architecture* (CORBA). Multi-tier client/server systems such as CORBA offer an ideal opportunity to implement an error-aware GIS in an environment that fulfills all the requirements for stability and platform independence, at the same time as opening the door to the many advantages of fully interoperable GIS.

6.3 Implementing a three-tier distributed system

Having decided upon multi-tier distributed systems as an appropriate vehicle for supporting the error-aware architecture, the next stage was to implement such a system to interface with the error-sensitive database. In fact, Java OOPL, used to implement the prototype error-sensitive GIS, proved ideal for this task. There is a clear analogy between OO and the client/server model, and indeed the strong similarity has led to considerable convergence between the two technologies (Loftus et al. 1995). In many ways the multi-tier client/server architecture represents the OO paradigm applied on a macro-scale to IS organisation. Objects both supply and consume the services of other objects; encapsulation encourages a focus on the *what* rather than the *how* of systems architecture; objects are conceptually distinct system components with well defined rôles. Most importantly, the analogy between the idea of an object interface, first introduced in §5.1.1, and a client interface is near perfect and is explored in more detail later in this section. In addition, the Java OOPL was developed with networking support at its core, and all of the packages needed to build Java middleware have been a standard inclusion in Java since the earliest releases.

6.3.1 Java and CORBA

Despite competition from rival middleware formats, such as Microsoft's distributed component object model (DCOM), CORBA is rapidly becoming the *de facto* industry standard as it is currently the only true cross-platform solution. The decision to build a Java rather than a CORBA distributed system therefore requires some explanation. The implementation used here is based on the Java *remote method invocation* package (RMI). RMI performs a very similar job to CORBA but with two distinct advantages. First, RMI is free software. Whilst CORBA is an open standard, the software needed to build a CORBA distributed system is usually proprietorial and commercial. Second, RMI is more powerful and less complex than CORBA. RMI is able to achieve these advantages over CORBA at the cost of being a Java-only solution (Harold 1997). In many cases this may not be a problem, since there exists a Java virtual machine (JVM) for most computer platforms. Further, while RMI cannot yet claim to be truly cross-platform in the same way as CORBA, there has in recent months been considerable convergence between RMI and CORBA. It looks certain that in near future CORBA will incorporate many of the features of RMI.

6.3.2 Request broker

Whether based on CORBA, RMI or DCOM, at the heart of any interoperable middleware is the *request broker*. The request broker presents a standardised interface for clients to access services. There are a variety of modes of operation between client and request broker, but most follow the same idealised pattern outlined in figure 6.3. A client wishing to access some service offered by the request broker needs some basic information about what services it requires. For this reason, request brokers usually include a *naming service*. The job of the naming service is to listen for client connections and requests for a particular named service that the client expects from the request broker. Assuming the requested service has been registered with the request broker, the request broker responds with the *client stub* named by the request. Here the analogy between OO and client/server models is completed, since the client stub is both an object interface and a client interface. The client stub defines the interface of a server object. With that server object, a client is then free to continue without any further explicit reference to the request broker. Client applications can be written to use the client stub transparently, as if it were an ordinary object. Behind the scenes the request broker mediates the services offered by the server object using a *server skeleton* that defines how the services offered in the client stub are implemented.

6.3.3 Implementation

The first task in developing any middleware is to determine what services it should provide. A range of services are likely to be needed by error-aware client applications both at the database and the object level. For example the ability to query the database and select database objects is likely to be necessary in addition to the ability to set and retrieve the quality of selected database

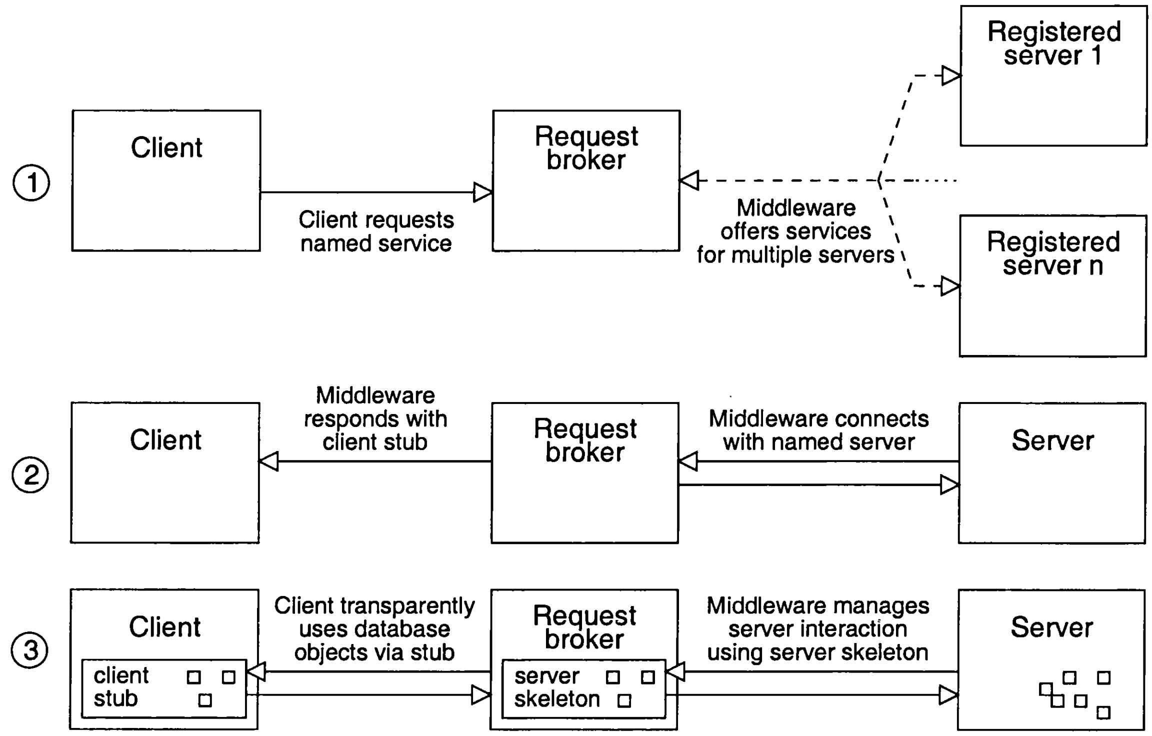

①
Client
Client requests named service
Request broker
Middleware offers services for multiple servers
Registered server 1
Registered server n
②
Client
Middleware responds with client stub
Request broker
Middleware connects with named server
Server
③
Client
client stub
Client transparently uses database objects via stub
Request broker
server skeleton
Middleware manages server interaction using server skeleton
Server

Figure 6.3: Three-tier interoperable architecture

objects. The final set of services provided by the error-aware component architecture is summarised in table 6.1. It is not suggested that these services are necessarily complete or comprehensive: they were developed to meet the needs of this research, but in an *ad hoc* manner. A more structured analysis of the services required by interoperable GIS clients would be needed before a commercial error-aware GIS could be implemented. In particular, a commercial error-aware GIS would need to address issues such as concurrency and transaction management which, while handled by the Gothic database, were considered beyond the scope of this research and are absent from the discussion of error-aware GIS presented here.

6.3.3.1 Gothic database services

Having decided upon the services needed for the error-aware GIS, the next task is to implement those services in the Gothic error-sensitive database. Whilst Gothic was not specifically designed to interface with an interoperable middleware client[1], Gothic does offer a relatively high degree of flexibility to program such a client interface. Probably the most desirable method would have been to use *remote procedure call* (RPC) libraries. RPC was a forerunner of interoperable architectures like RMI and CORBA and allows server functions to be called directly by client ap-

[1] Very recently, Laser-Scan have developed a middleware component for Gothic, called *Integrator*. However, even if it had been available in time for this project, Integrator offers a more limited range of services than were needed for this research, excluding for example schema definition services.

	Method name	**Function**
Schema definition	defineAssociation	Defines an association between two classes in the database schema (eg associating an attribute class with a geospatial object class)
	defineClass	Defines a new class in the database schema
	implementMetricQuality	Allows a specified class to be annotated with a particular metric quality class
	addBehaviourToClass	Defines a new method on a class
	addConstructorToClass	Defines a new constructor for a class
	useEfficientStorage	Turns on or off efficient data storage
Object selection	selectRegion	Selects all objects within a specified geographic region in the database
	selectGeoObjects	Retrieves the geospatial objects associated with a particular object
	selectGeometryObjects	Retrieves the geometry objects associated with a particular object
	deselectObject	Deselects a currently selected object
	deselectAllObjects	Deselects all currently selected objects
Object manipulation	createNewObject	Creates a new object of a particular class in the database
	getDBObject	Retrieves a single selected object from the database
	getDBVector	Retrieves a list of selected objects from the database
	getQuality	Retrieves the representative quality objects of a specified type for a particular object
	setQuality	Sets the representative quality of a particular object
	getValue	Gets the value of a specified object attribute or method
	setValue	Sets the value of a specified object attribute or method

Table 6.1: Java middleware services

plications (Rosenberger 1998). However, a simpler albeit less elegant solution was to use the *transmission control protocol* (TCP) socket libraries offered by Gothic. TCP is the network protocol upon which most higher client/server protocols, such as RPC, RMI, CORBA and the Internet, are based. The code for the Gothic socket server implementation can be found in appendix C in the `lull/server` directory. The code operates by creating a server object that listens for clients connecting to the database on a particular socket. Messages passed to the error-sensitive database are parsed and processed by the database and the results are then returned to the client.

The most significant disadvantage of using TCP based solutions is the high level of impedance mismatch. TCP offers data communication, but not object communication. The information contained in an object must be translated into a form that can be sent using TCP, in contrast to the component architectures used to implement the Java middleware which does allow direct communication between objects. The translation inevitably results in information loss, leakage or distortion. Using TCP the objects in the error-sensitive database will never fully correspond to those being manipulated by middleware or client applications. However, given limited resources the TCP socket programming solution proved a powerful and practical compromise.

6.3.3.2 Java middleware

Having decided upon the services needed by the error-aware GIS and upon a client interface for the Gothic error-sensitive database server, implementing the Java middleware is a relatively mechanical process. Network programming using Java is uncomplicated and the code for the Java middleware can be found in appendix C in the `java` directory. The key difference between the Gothic error-sensitive database client interface and the Java middleware client interface is that many of the Java middleware services are offered in a transparent object-oriented way. The Java middleware maintains a list of selected database objects. Error-aware client tools can access these Java objects and invoke methods on them, for example setting and retrieving quality information. The invoked methods actually then make a connection through the Java middleware to the Gothic database and return the results of the method invocation on the corresponding object in the Gothic database. However, this process is entirely hidden from error-aware clients, vastly simplifying the task of programming the error-aware GIS.

6.3.3.3 Starting the error-aware GIS

Before error-aware GIS clients can actually access any services, it is necessary to start both the Gothic error-sensitive database and Java middleware servers. This process entails a sequence of start-up operations, which must be performed just once.

- **Start error-sensitive database**: Starting the Gothic error-sensitive database is the first step in the start-up process, ensuring that both error-sensitive class definitions and the Gothic server class definition are loaded into Gothic.

- **Start error-sensitive database server**: A new server object is created in Gothic and the server starts listening for client connections.

- **Start the Java naming service**: Before the Java middleware is started the Java request broker naming service (registry) needs to start inside a new JVM. The Java release from Sun provides a simple command line interface that can start the naming service using the command 'rmiregistry'.

- **Start the Java middleware**: A second JVM is now needed to start the Java middleware. The most important step in this process is to ensure the Java middleware registers itself with the naming service. This can be achieved with a single line of Java code, as in the pseudocode in figure 6.4, which instructs the naming service to associate (termed *binding*) the new middleware object with a specified name (`gothic-server`).

6.3.3.4 Programming with the error-aware GIS

Having started the Java and Gothic servers, programming client error-aware Java tools is relatively straightforward. Figure 6.5 outlines an error-aware client class definition. Before an error-

```
// Set the package name for this class
package eaGIS.gothicSocket.gothicServer;

// Import Java RMI classes
import java.rmi.*;
import java.rmi.server.*;

// Gothic server object inherits from generic Java remote server class
// UnicastRemoteObject and implements gothicServerRMI interface
public class gothicServer extends UnicastRemoteObject
                          implements gothicServerRMI{

  // Constructor definition is called whenever a new object is created
  public gothicServer(){

    // Naming object supplied by rmiregistry. Rebind method associates
    // the name gothic-server with the gothicServer object being created
    Naming.rebind("gothic-server",this);
    ...
  }
  ...
}
```

Figure 6.4: Example Java middleware class

aware client can connect to the Java middleware, it first needs to retrieve the middleware object client stub from the naming service. In the pseudo-code in figure 6.5 the naming service queries a specific machine for the `gothic-server` object. This highlights the fact that error-aware client applications will usually be running on physically remote machines, and consequently the Java RMI naming service will need to be started on the local error-aware client machine as well as the remote host (in this case `m-duckham.geog.gla.ac.uk`). Having successfully retrieved the client stub, error-aware tools can query the database directly, for example selecting all objects in a specified region. Additionally, selected database objects can be used indirectly through Java objects held by the Java middleware. The method `getDBVector` returns a list of all the currently selected database objects. Individual database objects in this list can then be manipulated in the same way as any ordinary Java object.

6.4 Error-sensitive GUI

Two simple Java tools were developed to provide a front-end interface for the error-sensitive database, based on the component architecture described above. These tools provide a basic GUI for the error-sensitive database but stop short of being error-aware applications since they provide no specialised error handling functionality. However, they are still useful illustrations of the advantages of using distributed systems. By utilising a component architecture the GUI can

```
// Import Java RMI classes and gothicServer stub definition
import java.rmi.Naming;
import eaGIS.gothicSocket.gothicServerRMI;

// Import Java-Gothic database class definitions
import eaGIS.gothicBase.*;

public class exampleClient{

  public exampleClient(){

    // Lookup gothic server object using rmi naming service
    gothicServerRMI gs = (gothicServerRMI)Naming.lookup
         ("rmi://m-duckham.geog.gla.ac.uk/gothic-server");

    // Select all database objects in rectangular region
    gs.selectRegion();

    // Get the list of selected database object
    Vector db_list = gs.getDBVector();

    // Get the first database object in the list
    uncertainObject uo = (uncertainObject)db_list.elementAt(0);

    // Get any lineage objects associated with that object
    Vector quality_list = ou.getQuality("lineage");
    ...
  }
  ...
}
```

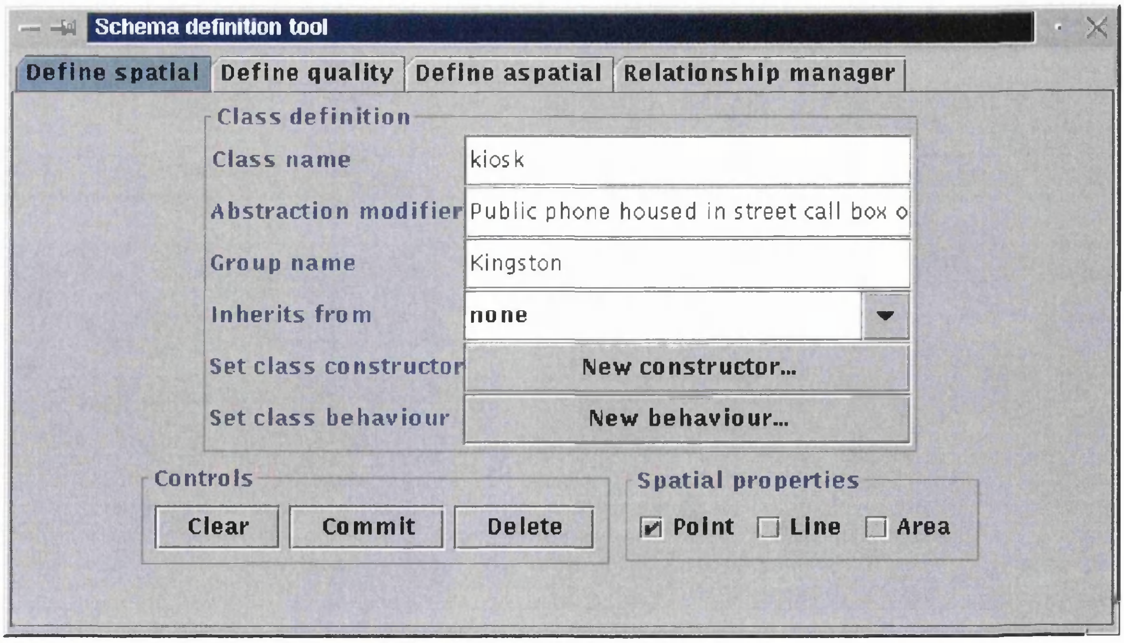
Figure 6.5: Example Java error-aware client class

be very closely integrated with the error-sensitive database while ensuring that the error-sensitive database is in no way dependent on elements in the GUI. The first error-sensitive GUI is a schema definition tool, allowing a user to define an integrated geospatial and quality schema. Second, a data browser allows very basic access to both geospatial and quality information in the database. Both tools were built using Java and Java's extended GUI library, called the *Java foundation classes* (JFC). This section gives a brief overview of each tool, whilst source code and documentation for both tools can be found in appendix C.

6.4.1 Schema definition tool

The schema definition tool provides a GUI for defining new geospatial and quality object schema based on the core error-sensitive object schema. The tool, shown in figures 6.6 and 6.7, is based on four tabbed frames, called *panes*, where each pane can be brought into the foreground by clicking on one of the tabs. The first three panes allow geospatial, attribute and quality classes to be defined. The geospatial class definition pane is illustrated in figure 6.6. The class name, description and inheritance can be defined for new classes using these three panes. All classes are obliged to inherit from uncertainty abstraction, or some sub-class of these two super-classes. Methods and constructors can also be defined for any new class, although there is no visual programming interface for the body of these methods which must be programmed in Lull. In addition to these core features, the geospatial class definition pane provides a mechanism for associating new geospatial classes with particular predefined error-sensitive geometry classes. The quality class definition pane provides spaces for defining the metric and restricted behaviour of new quality classes. The fourth pane (figure 6.7) offers an interface for associations between classes. Attribute classes can be associated with either quality or geospatial classes. New classes can also be associated with metric quality classes where appropriate.

Since the schema definition tool is written in Java the error-sensitive class definitions are themselves objects. Schema definition using the tool is essentially a process of constructing meta-data objects that describe new classes. A completed schema can be written to the error-sensitive database via the Java middleware using the schema definition functions outlined in table 6.1. Clearly, this meta-data information can be very important and is arguably a component of the data quality of any data set. Unfortunately, here again the Gothic database reveals its foremost weakness. In common with attributes and geometry, schema definitions are not treated as part of Gothic's object model and so it is very difficult to integrate the schema definition into the error-sensitive GIS once it has been written to Gothic. Like geometry, schemas are fundamental to the way Gothic operates and so rewriting Gothic to use an OO schema definitions would have been beyond the resources of this research.



Class name: kiosk

Abstraction modifier: Public phone housed in street call box o

Group name: Kingston

Inherits from: none

Set class constructor

Set class behaviour

☑ Point □ Line □ Area

Figure 6.6: Schema definition tool: Class definition

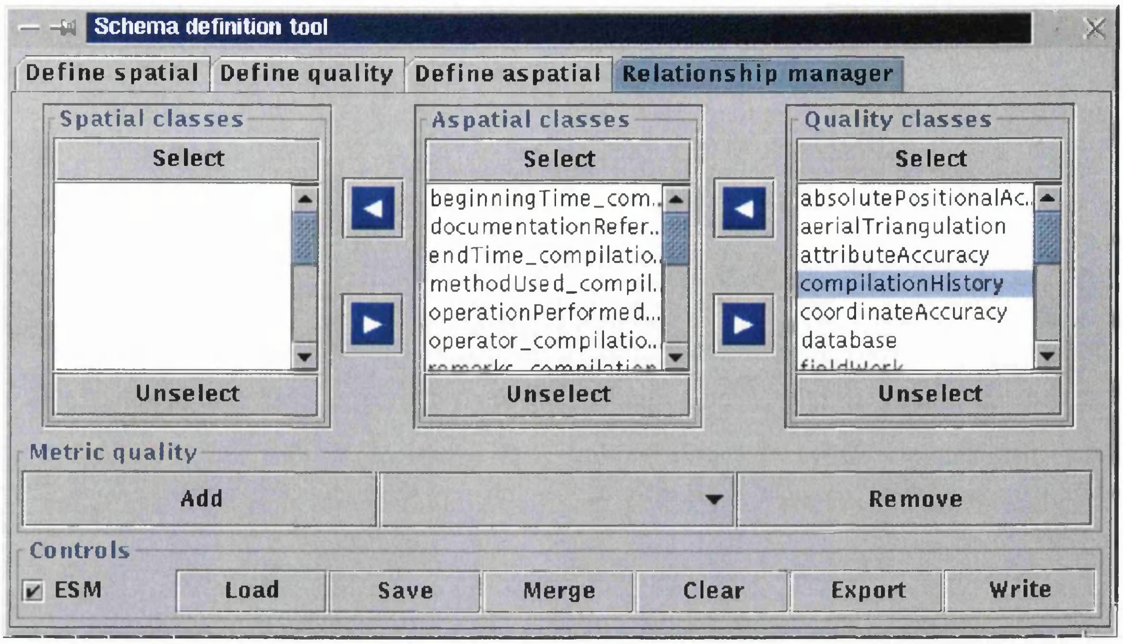


Figure 6.7: Schema definition tool: Associations

However, rather than throw this information away, the schema definition tool does attempt to retain schema meta-data. The Java schema definition objects are programmed to be persistent, using serialisation introduced in §5.1. Consequently, schema meta-data can be stored on a computer file-system. Further, in view of the importance of schema information, the Java middleware was also extended to be able to access this schema meta-data. The schema meta-data associated with a particular data set can be supplied to any client tool that requests it. Despite this, the schema objects still cannot be written directly to the Gothic database, although the Gothic database does hold this information in a non-OO form. It is important to note that a consequence of this approach is that the schema is not part of the error-sensitive data model and does not enjoy full participation in the error-sensitive database. To be consistent, schema definition object classes ought to inherit from the **uncertainty** class in the error-sensitive database in the same way as quality does. Were this the case, meta-quality information, such as information about who produced the schema, what methods they used and what compromises they made, could also be stored within the error-sensitive database.

6.4.2 Data browser

In addition to the schema definition tool a simple data browser was implemented using the distributed architecture illustrated in figure 6.8. The browser tool has a main window that consists of a geometry window, a simple zoom and pan tool and a selected object list. The tool can be used to select, view, create and update database objects, in much the same way as might be available in conventional GIS. The key advantage of using the distributed system is that it guarantees the separation between data and application. Changing or replacing the data browser will have no

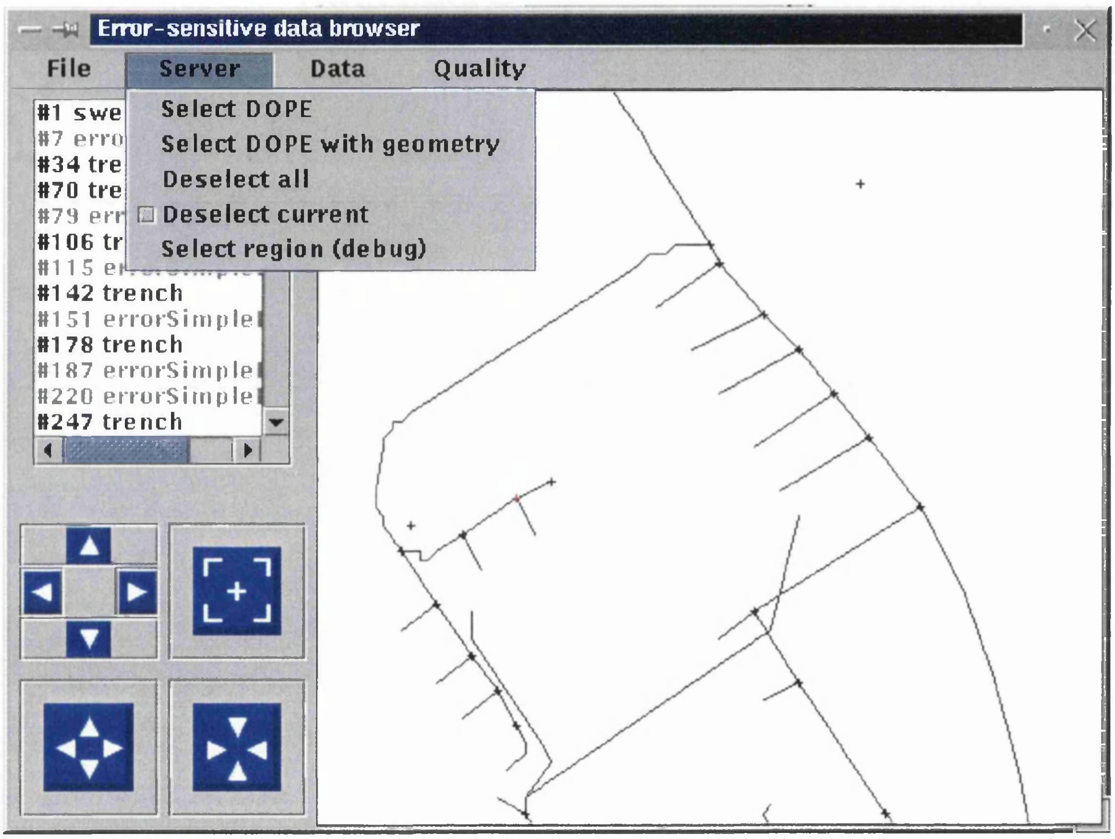


Figure 6.8: Data browser tool

effect upon the database whilst as long as the client interface remains constant, even the database can be changed or replaced without affecting front-end software, like the data browser.

6.5 Conclusions

The move toward distributed component architectures and away from monolithic systems is generally evident in software development. GIS in particular are ideally suited to take advantage of this new wave of software architecture. High software complexity and high data collection costs have in the past conspired to produce monolithic insular GIS that offer minimal cross-platform support, effectively locking-in GIS users and their data into one software system. By adopting interoperable component software architectures the GI software landscape could be completely transformed, encouraging both niche software development, data sharing, and mass market GI software (Frank and Kuhn 1995). In addition, distributed technology is naturally converging on OO. The client/server contractual metaphor explored in §6.2.2 is so close that the development of OO and client/server systems are inextricably linked. In fact the development of Java was to a large extent driven by the need for an OOPL with very strong networking support. This technological convergence could not come at a better time for GI science, which is itself steadily moving toward OO database technology. GIS are therefore currently well positioned to take advantage of this convergence and embrace component architectures.

The use of a component architecture in implementing an error-aware GIS, however, has one further crucial advantage. The attempt to integrate the diverse software extremes of an error-sensitive database and error-aware tools depends upon a robust yet flexible architecture for its success. Component systems certainly offer distributed interoperability, but it is their robust flexibility that is central to their deployment as a bridge between error-sensitive and error-aware GIS. The task of error-aware software programming is considerably simplified by being able to take advantage of error-sensitive functionality and access error-sensitive database objects transparently. Through the use of a distributed component architecture, there is now a clear path to the implementation of an error-aware GIS developed for a specific application. The next three chapters explore just such an example application, using the error-aware GIS architecture to address the error handling needs of a telecommunications legacy data capture project.

Chapter 7

Error-aware GIS: quality schema

The distributed systems architecture described in the previous chapter allows error-aware applications to be closely integrated with the error-sensitive database, yet developed with a high degree of independence from the error-sensitive GIS implementation. This powerful combination of integration and independence is the driving force behind error-aware software development. By building on the core functionality supplied by the error-sensitive GIS, error-aware software can afford to take advantage of domain specific and intelligent systems technology. The next three chapters describe the design and implementation of three separate error-aware GIS tools based around an example telecommunications database. Each error-aware tool is designed to address specific quality issues arising from this telecommunications example. Although they might well find use in other application areas with only limited modifications, the error-aware GIS architecture makes it feasible to develop and use such domain specific software. This chapter begins with an introduction to the telecommunications application used for this study, based in Kingston-upon-Hull, UK. The chapter continues with a look at the development and use of the first error-aware software tool, which uses an expert system to help telecommunications database designers to develop error-sensitive quality schema alongside their telecommunications schema.

7.1 Telecommunications application

To be successful, error handling in GIS must be able to provide practical advantages over conventional GIS. Whilst the aim of this research has been the development rather than the application of error handling in GIS, the research would not be complete without an example application. The application chosen here is that of a telecommunications network in Kingston-upon-Hull, UK. As already stated, utility applications are amongst the most important commercial uses of GIS, and telecommunications is one of the most active utility GIS application areas. Advances in and increased use of telecommunications technology, coupled with the ongoing deregulation of the UK telecommunications industry, which began with the Telecommunications Act in 1984, have led to a steady increase in the use of digital mapping in UK telecommunications companies. Consequently, many telecommunications companies are currently engaged in the process of transition from legacy paper-based mapping practices to digital GIS.

7.1.1 Application background

Kingston Communications (KC) PLC formed out of the Hull Corporation Telephone Department in 1987, in response to deregulation. In January 1997 KC embarked upon a project to migrate their legacy paper telecommunications plans to digital mapping. They enlisted the help of a computer consultancy, Informed Solutions, and a data capture and conversion company, Survey and Development Services (SDS), who were also the industrial sponsors for this research. These three companies together undertook the digital data capture of the entire Kingston-upon-Hull telecommunications network. This study focused on the capture of a small region typical of the total area being captured and approximately half of a 1km^2 area covered by UK National Grid coordinates (510,000, 434,000) to (511,000, 435,000).

The first stage of the work was the development of an OO database design that covered the important telecommunications features. Table 7.1 describes some of the key geospatial classes represented in this database design, produced primarily by Informed Solutions and implemented within Smallworld OOGIS. Prior to 1997, KC used Ordnance Survey of Great Britain (OSGB) 1:1250 base maps with telecommunications features marked on by hand for spatial data management, called 'plant-on-plan' maps. More detailed plans of the underground ducts containing telecommunications cables, called 'duct-plans', were also used in addition to plant-on-plan maps in some areas. The database design produced by Informed Solutions covers the features found on these plans. While this database was designed for use with Smallworld, since both Gothic and Smallworld use an OO approach it was a straightforward task to implement the Informed Solutions database design within Gothic via the error-sensitive schema definition tool described in §6.4.1.

The perceived advantage of migrating from legacy to digital data for utility companies often revolves around the use of the topological model for network maintenance and fault-finding. Topological consistency was excluded from the discussion of error-sensitive GIS in §5.3.4.3, since topology is generally well-understood and well-developed. Indeed, the database designed by Informed Solutions during the project does make extensive use of feature topology, in many ways vindicating the decision to restrict the scope of this study to less well understood areas of data quality. Following Informed Solutions' database design, features from these plans were captured by SDS. The first phase of data capture involved simply scanning and georeferencing the plans and returning them to KC. The reason for initially scanning plans was so that KC could start using digital georeferenced spatial data almost from the first day of the project. Subsequent to scanning, the second more lengthy data capture phase captures spatial and attribute information on the plans using conventional digitising techniques, to produce a full OO vector telecommunications

Geospatial feature name	Description
Cabinet	Cabinets are street level boxes used to joint main cables onto distribution cables.
Jointing chamber	Jointing chambers are similar to cabinets but are smaller brick or pre-cast concrete structures that join more minor cables.
Track route	A track route is the underground geographical route taken by a single duct or collection of ducts between two network features.
Coupling	A coupling is an openable concrete box that joins four underground ducts.
Tee	A tee is a plastic joint for three underground ducts. Tees have more recently replaced couplings.
Reducer	A reducer is an underground connector between two cables of different diameter.
Wallbox	A wallbox is a metal box recessed into a building wall which is usually attached to a duct.
Pole	A pole is the familiar telegraph pole, used to suspend cables above ground.
Distribution point	Distribution points are small joints which connect distribution cables to a drop cable, which in turn connects to a customer's premises.
Kiosk	A kiosk is the familiar telephone kiosk housed in a street call box or public building.
Cable route	A cable route is the geographical route taken by a cable or group of cables.
Miscellaneous	A number of miscellaneous geospatial features are also maintained by the KC database, such as way-leaves, hazards and site restrictions.

Table 7.1: KC telecommunications features

GIS within Smallworld. Once completed, the vector data for each region was substituted for the scanned maps, thereby smoothing the transition from legacy to digital mapping practices. Based on experiences with all the companies involved, KC, SDS and Informed Solutions, three error-aware GIS tools were developed to compliment the requirements of the data capture project, the first of which is explored in the following section.

7.2 Intelligent quality schema definition tool

One of the goals of the error-sensitive GIS development was to provide only core error-sensitive functionality and not to restrict error-sensitive GIS users to predefined data quality elements or standards. Therefore, the first task facing any error-sensitive GIS application, such as in the ς-calculus example in §5.4, is to define the quality schema to be used in the error-sensitive database. The process of quality schema definition is essentially another mini-OOAD of the quality implications of a particular application, based on the core error-sensitive OO data model. In common with any OOAD, this process inevitably entails a degree of subjectivity and judgment (see §3.2.1). While companies such as SDS and Informed Solutions may be adept at designing OO geospatial databases, they are highly unlikely to have experience in designing error-sensitive geospatial databases, since most commercial GIS offer no error handling functionality. Without this experience, the error-sensitive database designer will lack the basis on which to make subjective judgments about quality schema design.

One way to address this problem is to provide tools that help error-sensitive database designers to incorporate data quality classes into their overall database design. The danger is that without providing such tools, the error-sensitive functionality will never be used. An error-sensitive GIS provides functionality additional to basic GIS functionality. Consequently, there is nothing to prevent a database designer using an error-sensitive GIS as a conventional OOGIS. The error-aware GIS architecture allows the development and use of simple domain specific database design tools that should encourage the database designer to take full advantage of the error-sensitive GIS.

7.2.1 Tool architecture

The problem addressed by the intelligent schema definition tool is to perform a simple OOAD based on the geospatial object schema. The basic tool design attempts to simulate the OOAD process illustrated in figure 7.1, after Booch (1994). In fact, while Booch (1994) presents mechanisms for analysis of both classes and individual objects which are instances of those classes, with respect to quality schema definition it is the classes alone that are of particular interest. The analysis and design process begins by establishing rough boundaries for the classes that cover the problem domain. By incrementally refining the core ideas, introducing more detailed semantics, structures and relationships, the process can arrive at an implementation for these core classes. Finally, the process feeds back into further class definition and refinement, producing practical results only after a number of iterations.

In order to simulate this process, the intelligent schema definition tool uses a hierarchy of inter-related terms drawn from across the literature on data quality, which attempt to describe the 'quality solution space'. These terms are organised so that more general 'super-terms' appear near the root of the hierarchy, and more specific 'sub-terms' appear nearer the leaves of the hierarchy. For example, in figure 7.2 general terms like 'lineage' may be associated with more specific terms like 'source', 'usage' and 'compilation history'. In turn, these more specific terms may be associated with more specific terms still, while lineage covers only one aspect of data quality. The error-sensitive schema definition process takes the form of a user-led attempt to impose restrictions, relationships and additional structure on this loose hierarchy through the identification of the important quality implications and characteristics of a particular geospatial object schema. The result of this process should be an OO quality schema that is integrated with the original geospatial schema and addresses the quality requirements of that geospatial schema.

The intelligent schema design tool uses the hierarchy of quality terms as a background to two complementary technologies which each tackle a portion of the OOAD cycle in figure 7.1. The first of these technologies is a traditional expert system which is able to structure information in

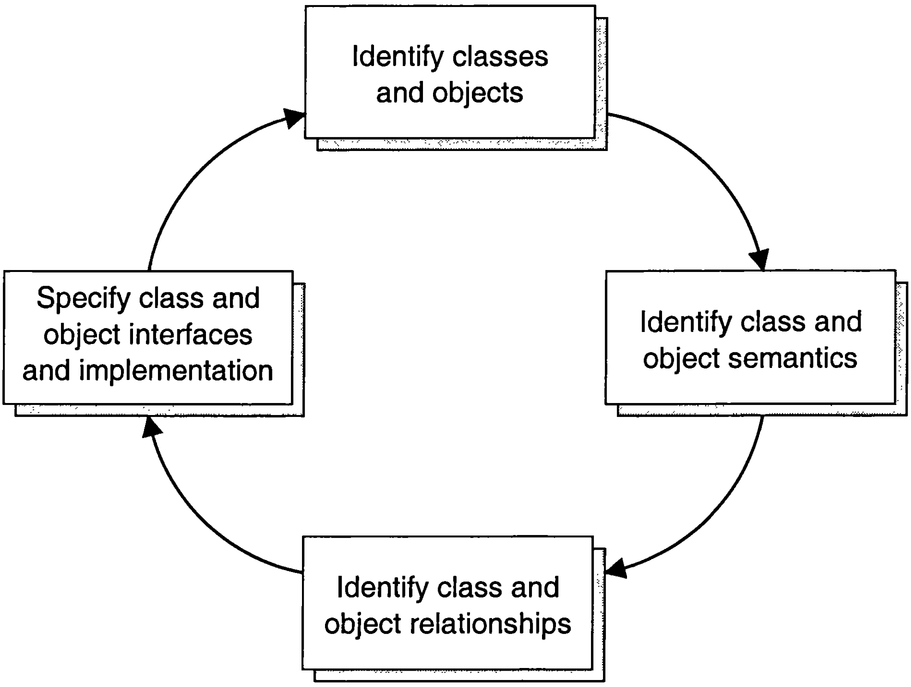

Identify classes and objects
Identify class and object semantics
Identify class and object relationships
Specify class and object interfaces and implementation

Figure 7.1: Object-oriented analysis and design process (Booch 1994)

the quality term hierarchy into useable class interfaces and implementations. The second technology involves related but more recent innovations in the use of hypertext intelligent systems. A hypertext-based system is used to promote user identification of quality terms from the hierarchy and their relation to the geospatial object schema. The relationship between these two technologies and the OOAD process is illustrated in figure 7.3. The following discussion introduces these two AI technologies and highlights the relative strengths and weaknesses of expert systems and hypertext and their importance to GIS.

7.2.1.1 Expert systems, hypertext and GIS

The introduction to this section (§7.2) indicated the importance of experience and judgment in the task of error-sensitive schema definition. Such qualities are not easily encoded into conventional software. However, a range of AI techniques that can claim to address these needs does exist. Russell and Norvig (1995) distinguish between *human intelligence* and *rationality* (an idealised form of intelligence) and identify four distinct strands of research in AI: systems that act like humans, systems that think like humans, systems that act rationally and systems that think rationally. It is the first of these, the ability to act like an intelligent human, that is of the greatest importance in the development of software able to mimic human qualities such as experience and judgment. This ability is primarily the preserve of one of the longest established areas of AI technology, expert systems.

Definitions of expert systems in the literature are legion. At the core of most definitions is

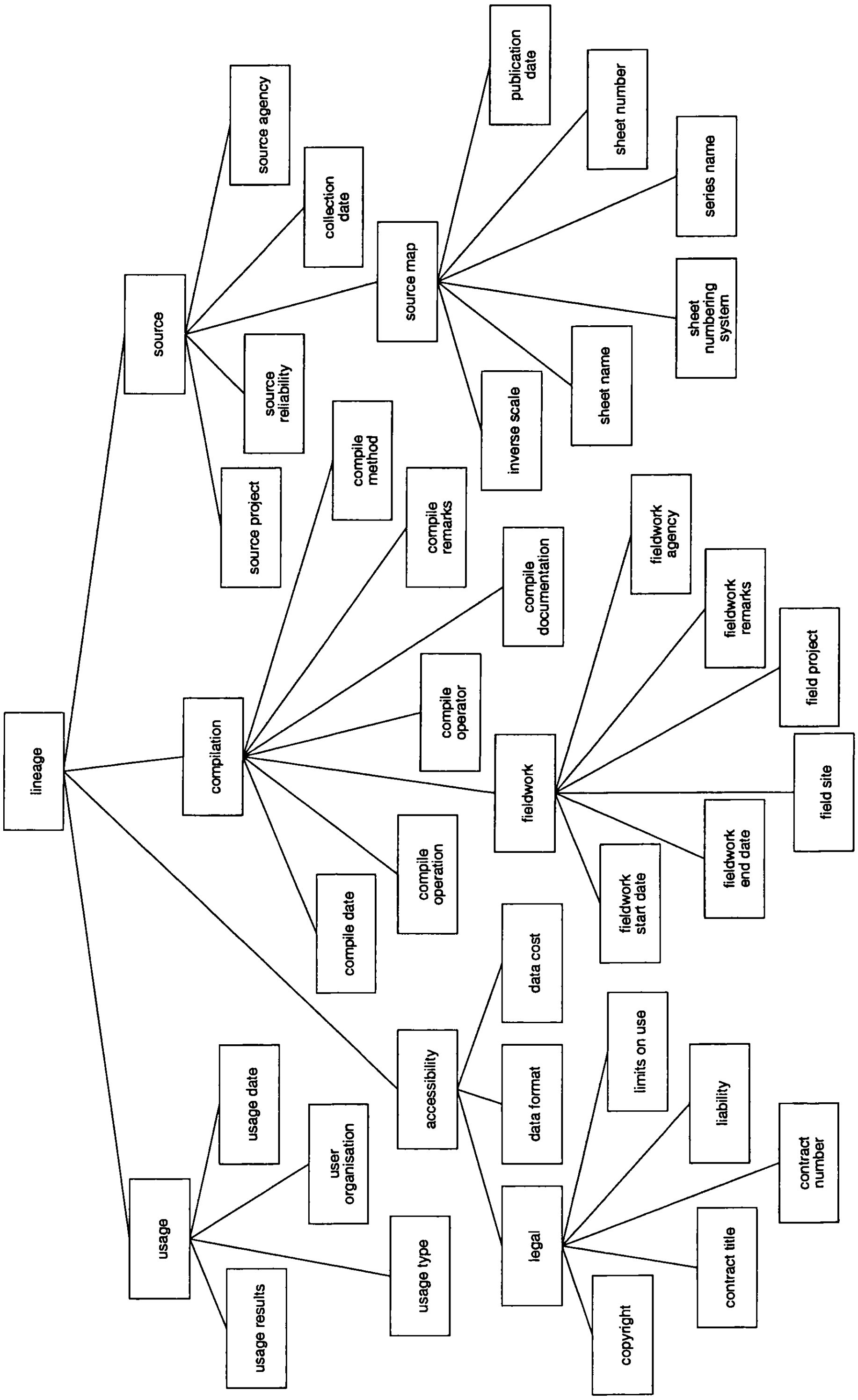


Figure 7.2: Quality hierarchy solution space for lineage

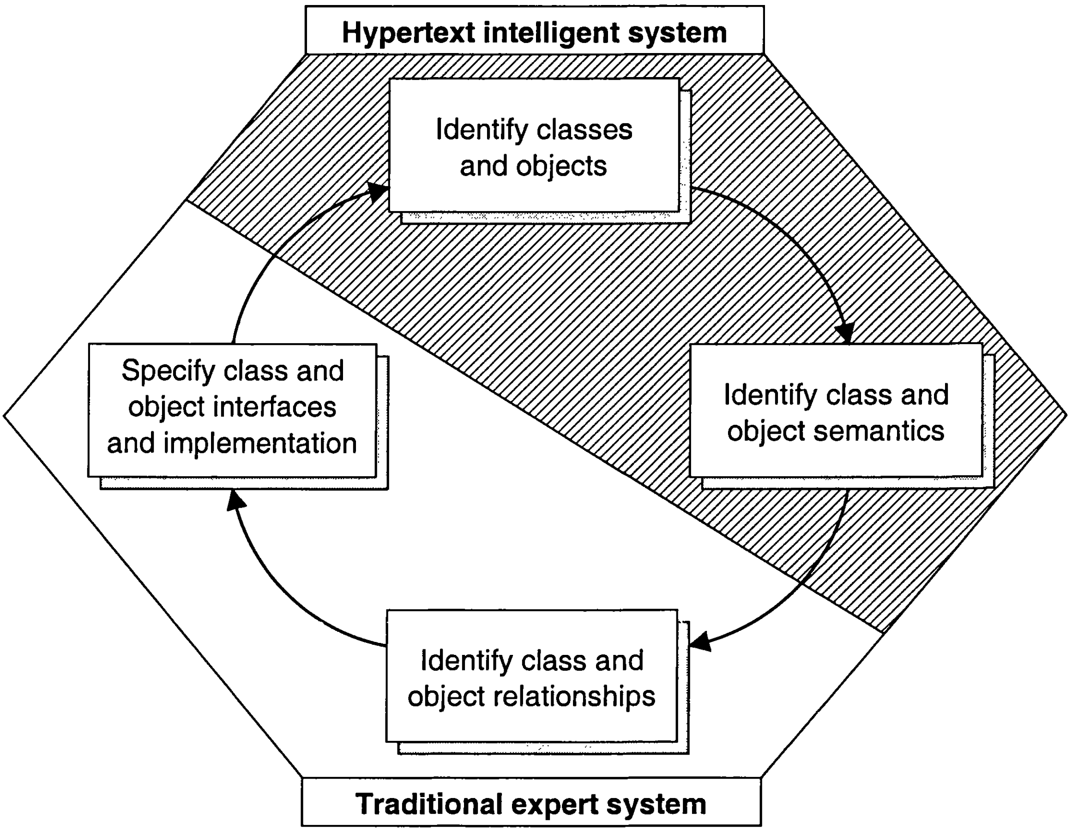

Hypertext intelligent system
Identify classes and objects
Identify class and object semantics
Identify class and object relationships
Specify class and object interfaces and implementation
Traditional expert system

Figure 7.3: Twin-track expert system architecture

the idea that an expert system is a computer system designed to emulate the behaviour of a human expert in a specific narrow domain (see for example Denning 1986; Maggio 1987; Frost et al. 1994). Expert systems, and the related field of *knowledge-based systems,* have already enjoyed extensive use within GI science, for example in the fields of cartography (Robinson and Jackson 1985; de Jong and van der Wel 1990; Fisher and Mackaness 1993; Forrest 1993), remote sensing and image classification (Fisher et al. 1988; Civco 1989; Srinivasan and Richards 1993; Johnsson 1994), database access and query (Egenhofer and Frank 1990; Smith and Yiang 1991; Zhu 1996), planning and natural resource management (Stanton and MacKenzie 1989; Lein 1992; Skidmore et al. 1996), map reliability and uncertainty (Fisher 1989; Dutton 1996) and more generally as a technique for incorporating intelligent behaviour into GIS (Lilburne et al. 1996).

However, traditional expert system designs do have failings and in particular tend toward weak user interfaces (Harris-Jones 1995). One important technique that has been used to address this failing is the use of hypertext. Hypertext is essentially a network of structured natural language information, although the term is increasingly used in place of *hypermedia* to denote a structured network of multi-media information more generally. In either event, hypertext comprises 'flat documents', termed *nodes,* and connections between those documents, termed *links* (Agosti 1996). Information stored as hypertext is familiar to most computer users, as the majority of information on the Internet is written and organised using the hypertext format, HTML.

The importance of hypertext to human-computer interaction is that it provides an intuitive user interface for non-sequential navigation of informal knowledge. The potential for using hypertext as an interface to GIS has prompted some research interest (Linsey and Raper 1993). However, it is as an expert system interface that hypertext has proved most useful. Hypertext can provide a bridge between informal human knowledge and the more formal (albeit symbolic) knowledge required by expert systems (Woodhead 1991). The benefits of using hypertext as an expert system user interface are expounded by Bielawski and Lewand (1991). The key advantages for the quality schema definition tool are two-fold. First, the non-sequential nature of hypertext allows for *ad hoc* context sensitive information access in a manner which may not have been foreseen by the system designers. Second, hypertext has developed from cognitive and psychological research and is itself a form of knowledge representation (KR). In the same way as OO has its roots in AI, the provenance of hypertext can be traced to a number of AI disciplines, such as frames (touched upon in the discussion of the features of OO in §3.1.1.3) and semantic networks (Kaindl and Snaprud 1991; Woodhead 1991). Hypertext represents a structured form of knowledge which, while it falls short of being an expert system in itself, is complementary to expert systems.

7.2.2 Traditional expert system development

While expert systems have been used to tackle experience, judgment and heuristic laden tasks across a variety of spatial and non-spatial domains, expert system practitioners are agreed that a key component of successful expert system development is a clear definition of the problem domain (Denning 1986; Waterman 1986). The responsibilities of the traditional expert system component of the intelligent schema definition tool are clearly set out in figure 7.3. While subjective, the task of OOAD is also clearly defined; the expansive OOAD literature is in accord over the fundamentally iterative and incremental nature of the analysis and design process (see §3.2.1). Despite clear problem definition, OOAD has proved relatively difficult to automate using expert systems, reflected in a paucity of literature on automated OOAD. As a result of the high degree of inventiveness and the breadth of 'common sense' knowledge that is often associated with OOAD only a few studies have had any success in producing automated OOAD (eg Belkhouche and Gamiño 1998).

Fortunately, the OOAD of an error-sensitive quality schema is a sufficiently narrow problem sub-domain as to make the development of semi-automated expert system tools entirely feasible. Indeed, the expert system explored in the remainder of this section is intentionally and unapologetically simple. A number of authors have noted a dichotomy in expert system design, between the extremes of highly sophisticated, multi-purpose 'deep' systems and single-use, rapid development, 'shallow' systems (Denning 1986; Harris-Jones 1995). The error-aware GIS architecture is ideally suited for lightweight, targeted, rapid development intelligent agents which fulfill specific needs. By taking advantage of this architecture, the expert system component of the intelligent schema definition tool can be tailored to the highly specialised task of error-sensitive schema design at the same time as minimising development time and costs.

7.2.2.1 Expert system design

The traditional expert system component of the quality schema definition tool aims to identify relationships, specify interfaces and structure previously defined classes, as shown in figure 7.3. The starting point for this process was to formalise the hierarchy of quality terms using first-order predicate calculus (FOPC). In the same way as the use of ς-calculus was proposed in earlier chapters as an important tool in OO system development, so formal logic is recognised as important during expert system development (Robinson and Frank 1987). The hierarchy of quality terms can be constructed using two *predicates*, labelled $QTerm$ and $SubTerm$. These predicates are used to make statements about quality terms (such as 'lineage') or variables (such as x). For example, the term in 7.1 below asserts that the quality term 'usage' is a sub-term of 'lineage'.

$$QTerm(usage) \wedge QTerm(lineage) \wedge SubTerm(usage, lineage) \tag{7.1}$$

General rules can be constructed from predicates using standard logical connectives, such as AND ($\wedge$), OR ($\vee$) and IMPLICATION ($\Rightarrow$). For example, we can use the predicates $QTerm$ and $SubTerm$ in a pair of rules that define a new predicate $SubSet$, the transitive closure of $SubTerm$ (equations 7.2 and 7.3).

$$QTerm(x) \wedge QTerm(y) \wedge SubTerm(x, y) \Rightarrow SubSet(x, y) \tag{7.2}$$

$$QTerm(x) \wedge QTerm(y) \wedge QTerm(z) \wedge SubSet(x, y) \wedge SubSet(y, z) \Rightarrow SubSet(x, z) \tag{7.3}$$

An expert system uses these rules to reason about particular situations. For example, given the three increasingly specialised quality terms, 'accuracy', 'absolute_accuracy' and 'rmse', and the rules in equations 7.2 and 7.3, an expert system should be able to infer 'rmse' is a sub-set term of 'accuracy', $SubSet(rmse, accuracy)$, as in equation 7.4 below.

$$\begin{aligned}&QTerm(accuracy) \wedge QTerm(absolute\_accuracy) \wedge QTerm(rmse) \wedge\\ &\quad SubTerm(absolute\_accuracy, accuracy) \wedge SubTerm(rmse, absolute\_accuracy)\end{aligned}$$

We can use equation 7.2 with three of the conjunctive predicates to infer a sub-set relation between 'absolute accuracy' and 'accuracy' as shown below

$$\begin{aligned}&QTerm(accuracy) \wedge QTerm(absolute\_accuracy) \wedge SubTerm(absolute\_accuracy, accuracy)\\ &\quad \Rightarrow SubSet(absolute\_accuracy, accuracy)\end{aligned}$$

Similarly using equation 7.2 we can infer a sub-set relation between 'rmse' and 'absolute accuracy'

$$\begin{aligned}&QTerm(absolute\_accuracy) \wedge QTerm(rmse) \wedge SubTerm(rmse, absolute\_accuracy)\\ &\quad \Rightarrow SubSet(rmse, absolute\_accuracy)\end{aligned}$$

Finally, using the two inferred sub-set relations in combination with equation 7.3 we can infer a sub-set relation between 'rmse' and 'accuracy'

$$\begin{aligned}&QTerm(accuracy) \wedge QTerm(absolute\_accuracy) \wedge QTerm(rmse) \wedge\\ &\quad SubSet(absolute\_accuracy, accuracy) \wedge SubSet(rmse, absolute\_accuracy)\\ &\quad \Rightarrow SubSet(rmse, accuracy)\end{aligned} \tag{7.4}$$

In this way, the expert system stores symbolic knowledge in a *knowledge base* in the form of primary *facts*, such as $SubTerm(rmse, absolute\_accuracy)$, and rules, such as those in equations 7.2 and 7.3. The expert system is then able to derive new facts based on these primary facts and rules. The example in 7.4 implicitly uses *forward chaining*, where facts added to the knowledge base are used to trigger any rules which can deduce new facts. In addition to forward chaining, many expert systems use the converse process termed *backward chaining*. Given a conclusion, such as $SubSet(rmse, accuracy)$, a backward chaining expert system attempts to determine whether the conclusion can be supported by the facts and rules in the knowledge base.

Appendix B gives the core FOPC rules for a modest rule-based expert system able to structure quality term information. The expert system rules depend upon a range of primary facts supplied to the knowledge base, such as $QTerm$ and $SubTerm$ discussed above. The rules in appendix B enable this primary information to be structured into a quality schema. A detailed explanation of each rule would be lengthy and largely unnecessary. Consequently, each rule in appendix B is annotated with a brief sentence which indicates the informal semantics of each of the rules. Overall, the rules embody two antagonistic tendencies. On the one hand there is a tendency during error-sensitive schema definition to include very detailed quality schema using the maximum of quality terms and structure that can support a precise and complete description of the quality of that data set. On the other hand, concise quality schema are less complex and easier to understand. The expert system rules work by attempting to compromise between the use of both a more precise and a more concise quality schema.

The rules in appendix B were arrived at by thinking about the process of quality schema OOA and trying to pinpoint relevant rules that mirrored the essential points of this process. It should be stressed that a deep expert system would require much more sophisticated *knowledge acquisition* techniques for developing both the rules and the hierarchy of quality terms. The results of the naïve knowledge acquisition process used here are only useful in the context of the specific error-aware software being developed and will never be able to provide any meaningful information more generally about the process of OOA. However, as already mentioned, this 'shallow' development process is both rapid and highly compatible with the core error-aware architecture.

7.2.2.2 Expert system implementation

In addition to the knowledge base, fundamental to conventional expert system architecture is the *inference engine*, where rules and symbols can be manipulated according to logical operations. This study made use of a pre-existing inference engine, called the *Java expert system shell* (JESS, Friedman-Hill 1999). JESS is a Java-based version of a popular expert system, called the *C language integrated production system* (CLIPS) developed in the mid-1980s by NASA (Riley 1999). By using JESS, the expert quality schema definition tool can benefit both from the advantages of using an established inference engine and from high levels of integration between the inference engine and other Java code.

Based on the FOPC rules in appendix B, a simple rule-based expert system was implemented using the JESS inference engine. The are some differences between the way JESS manipulates rules and the way these rules are represented in FOPC. JESS uses a forward chaining algorithm based on pattern matching. JESS attempts to match facts in the knowledge base to the left-hand side of rules in the knowledge base. For example, figure 7.4 shows essentially the same rule as given in equation 7.3 rewritten in JESS. The first line of figure 7.4 defines a new rule labelled `SubTrans`. JESS will only activate this rule when it finds facts in the knowledge base that match the patterns on the second line of figure 7.4. Once the rule is activated, JESS uses these facts to assert the new fact for the rule right-hand side, shown on the third line of figure 7.4.

As a consequence of the differences between FOPC and pattern matching, some changes to the rules in appendix B were necessary in order to complete the implementation process. The negated existential quantifiers ($\neg\exists$) in a number of equations in appendix B, for example equation B.6, have no direct equivalent in JESS. Since JESS depends on matching patterns using individual facts, it is not possible to directly formulate a JESS rule based on *all* or *no* facts in the knowledge

```
(defrule SubTrans
  (QTerm ?x)(QTerm ?y)(QTerm ?z)(SubSet ?x ?y)(SubSet ?y ?z)
  => (assert (SubSet ?x ?z)))
```

Figure 7.4: Example JESS rule

base matching a particular pattern. Instead, any rule that depends upon such a pattern needs to be rewritten in JESS as two separate rules. The first rule will find all the occurrences of the specified pattern, and the second can then use any facts that do not have this specified pattern. The general case is illustrated by the FOPC rule in equation 7.5, which cannot be supported in JESS, and the equivalent pair of rules in equation 7.6, which can be supported by JESS. It is important to note that this introduces an element of precedence into the discussion: the first rule in equation 7.6 is implicitly evaluated *before* the second rule. While this precedence cannot really be represented in FOPC, JESS provides a predicate which can be used to control the order in which rules are evaluated, termed *salience*.

$$\forall y \; (\neg \exists x \; P(x,y)) \Rightarrow R(y) \tag{7.5}$$

$$\begin{aligned} \forall x,y \; P(x,y) &\Rightarrow Q(y) \\ \forall y \; \neg Q(y) &\Rightarrow R(y) \end{aligned} \tag{7.6}$$

For example, equation B.6 in appendix B informally states that 'any selected term that has no selected sub-set terms is a quality attribute'. In JESS this can be rewritten as two rules which state 'any selected term that has selected sub-set terms is a non-attribute' and 'any selected term that is *not* a non-attribute is a quality attribute'. The rules that would be used in JESS to represent these two statements are given in figure 7.5 below. These rules also include a salience declaration that ensures the second rule, `QAtt2`, will be evaluated after the first, `QAtt1`. Aside from such relatively straightforward changes, the expert system rules in appendix B were implemented directly within JESS, and the full JESS rule base can be found in appendix C.

7.2.3 Hypertext system design

Given adequate information from the user about the domain of interest, the expert system inference engine and knowledge base, described above, can structure this information into a useful quality schema. However, the rule-based expert system described above still lacks any mechanism for eliciting this primary information in a suitable form from the user. In order to feed the

```
(defrule QAtt1 (declare (salience 100))
  (Selected ?x)(Selected ?y)(SubSet ?x ?y)
  => (assert (NonAttribute ?y)))
(defrule QAtt2 (declare (salience 99))
  (Selected ?x)(not (NonAttribute ?x))
  => (assert (Attribute ?x)))
```

Figure 7.5: Example JESS rules with salience

inference engine with user-defined facts about the problem domain, the second component uses a hypertext user interface. The hypertext interface developed for the intelligent schema definition tool employs a recent innovation in Java-based Internet technology to allow dynamism and state to be incorporated into the hypertext interface, described below.

7.2.3.1 Java servlets and dynamic HTML

Hypertext sites on the Internet are provided by *web-servers*, which are server programs that provide a (relatively) standardised interface for client *web-browsers* to request particular *web-pages*. Traditionally, web-pages are stored as individual static HTML files on a file system accessible by the web-server. Figure 7.6 below illustrates the conventional web-server architecture, where requests for hypertext documents are met by responding with HTML files from a static hierarchy or network stored on a file system. The use of web technology in the intelligent schema definition tool is primarily a result of the very strong support such technology provides for hypertext, rather than to make the tool accessible over the Internet.

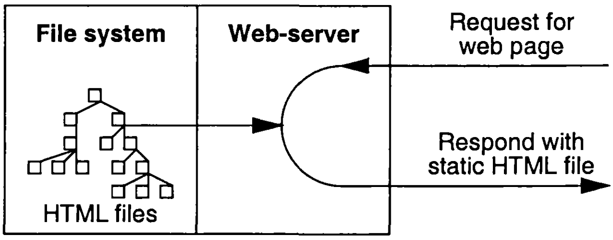


Figure 7.6: Conventional web-server

One difficulty with using hypertext is that it tends to be associated with a static, stateless model of user interaction. Hypertext pages are usually static because the nodes and links in a network of hypertext documents are often pre-defined in a single configuration. While there may be many possible paths through the hypertext network, all possible paths are fixed during the hypertext authoring process. Hypertext is usually stateless because the links merely direct the user to another page, but do not allow the system to 'remember' where the link originated from nor previous occasions the document may have been accessed. This shortcoming can be observed when using the Internet where until recently almost all web-sites have offered only static stateless web-pages.

Both problems of static and stateless web-pages are addressed by a recent web-based innovation called Java *servlets*. As suggested in §6.3.1, the JVM is highly platform independent and can be embedded within a wide range of other hardware and software. Java servlets are the result of embedding a both a JVM and a specialised servlet API within a web-server. Servlet objects within a web-server JVM are in many ways an OO analogue of a web-page; they are HTML documents with both state and behaviour. By using servlets, the intelligent schema definition tool hypertext interface can both exhibit dynamism and retain state. Figure 7.7 attempts to illustrate the Java servlet architecture, where requests for web-pages are dealt with by individual servlet objects in the web-server's JVM. The servlets are free to interact with each other or the file system in responding to a request and dynamically constructing a web page that may be unique to a particular client and a particular request.

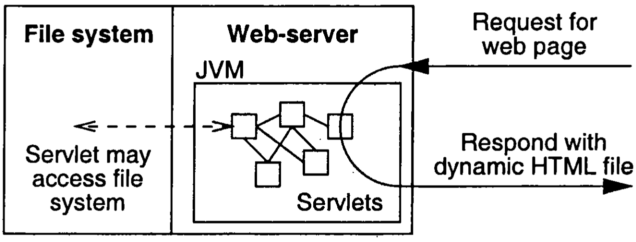

File system
Web-server
JVM
Request for web page
Servlet may access file system
Respond with dynamic HTML file
Servlets

Figure 7.7: Java servlet web-server

7.2.3.2 JESS-servlet interface

Since both servlets and JESS use a Java API, integrating hypertext and traditional expert system components of the intelligent schema definition tool was trivial. The example Java code in figure 7.8 illustrates the key features of a servlet class which communicates with JESS. The JESS API includes a expert system engine class (`jess.Rete`) and a input parser class (`jess.Jesp`). Together objects of these two classes support most of the functionality necessary to create and run and expert system shell within a Java servlet.

7.3 Implementation results

The combination of the two tools explored in the previous section, JESS and servlets, formed the basis of the intelligent schema definition tool. A range of Java servlet web-pages, which can be found in appendix C, were developed to act as the tool's user interface. Through a web-browser, users can 'surf' these servlet pages, incrementally and iteratively building up a picture of the quality features of a data set, in this case the KC telecommunications database. The servlet pages allow users to answer informal natural language questions about their data set. Individual

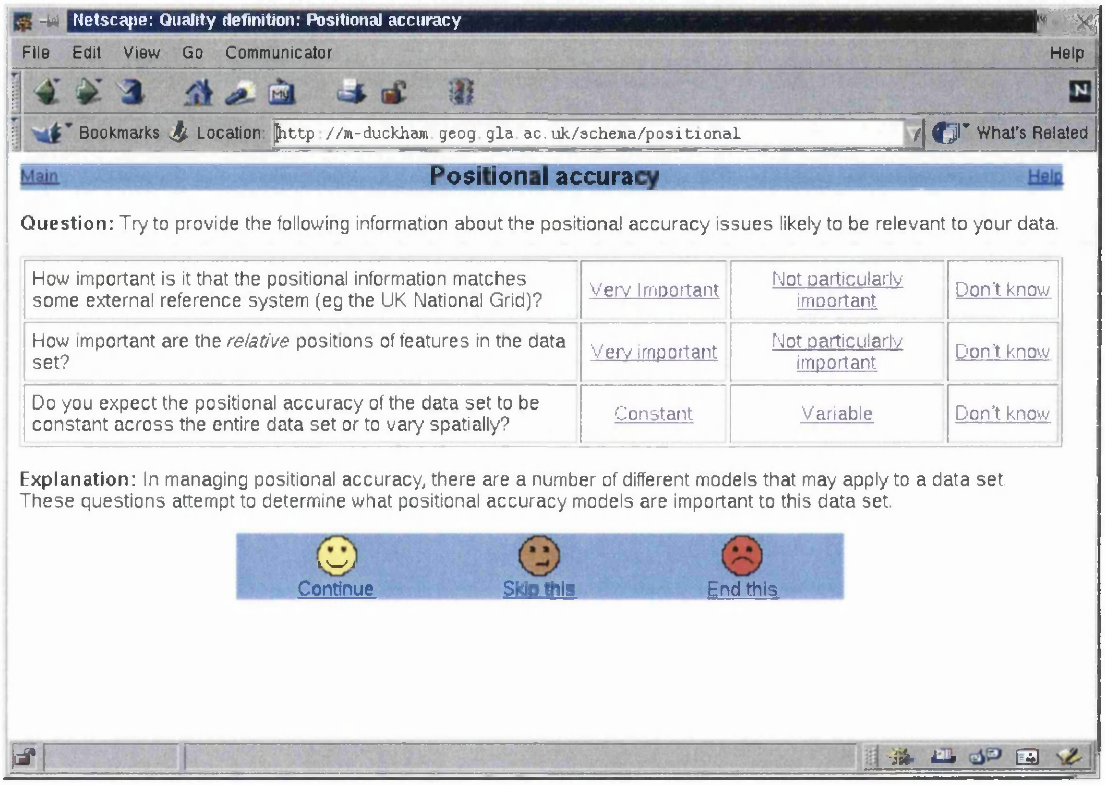
```
import jess.*;
import java.util.*;
import weblogic.html.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class JessServlet extends HttpServlet{
  // Create a new expert system engine
  private Rete rete = new Rete();

  // Constructor initialises connection to Jess
  public JessServlet(){
    // Open the rule-base with a new jess parser
    Jesp j = new Jesp(new FileReader("rb.clip"), rete);
  }

  // doGet called when servlet page requested
  public void doGet(HttpServletRequest req, HttpServletResponse res)
    throws ServletException, IOException{
    // Create HTML page elements
    ServletPage page = new ServletPage();
    TableElement t = new TableElement().setBorder(1);

    // Print out a table of facts in the JESS knowledge base
    Enumeration e = rete.listFacts();
    while(e.hasMoreElements()){
      TableRowElement tr = (new TableRowElement())
         .addElement(((Value)vv.get(RU.CLASS)).stringValue());
      t.addElement(tr);
    }
    page.getBody().addElement(t);
    page.output(res.getOutputStream());
  }
}
```

Figure 7.8: JESS-servlet interface

servlet pages then deposit this knowledge, in the form of symbolic facts, in the JESS knowledge base. This knowledge is manipulated by the JESS expert system to produce a structured quality schema. Once completed, this quality schema can be saved to file and exported to the error-sensitive database via the distributed architecture discussed in the previous chapter.

7.3.1 Intelligent schema definition tool interface

Since the interface for the intelligent schema definition tool interface is entirely HTML-based, it can be accessed using any common web-browser, such as Netscape. From the user's perspective, the key advantages of an hypertext interface are that it is intuitive, simple and familiar. Interaction with the tool interface has three distinct stages. The first stage is to identify a pre-existing geospatial object schema to be used as a basis for an error-sensitive schema, in this case the telecommunications database schema developed by Informed Solutions and outlined in table 7.1. The intelligent schema definition tool expects this geospatial object schema to be in the same persistent object format as produced by the error-sensitive schema definition tool described in the previous chapter (§6.4.1). As hinted in §6.4.1, a future error-sensitive GIS implementation would probably allow schema definitions to be included within the error-sensitive database, and in such a case the intelligent schema definition tool could retrieve schema information directly from the database.


Netscape: Quality definition: Positional accuracy
File Edit View Go Communicator Help
Bookmarks Location: http://m-duckham.geog.gla.ac.uk/schema/positional
What's Related
Main
Positional accuracy
Help

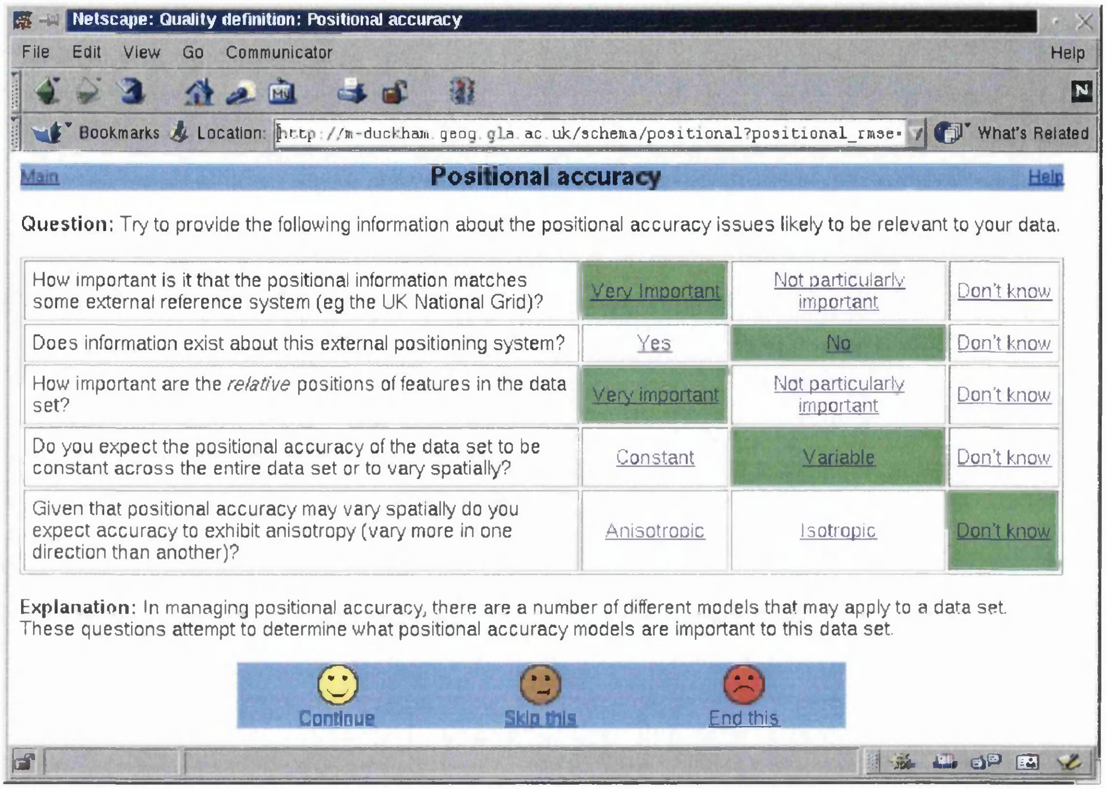
Question: Try to provide the following information about the positional accuracy issues likely to be relevant to your data.

How important is it that the positional information matches some external reference system (eg the UK National Grid)?	Very Important	Not particularly important	Don't know
How important are the *relative* positions of features in the data set?	Very important	Not particularly important	Don't know
Do you expect the positional accuracy of the data set to be constant across the entire data set or to vary spatially?	Constant	Variable	Don't know

Explanation: In managing positional accuracy, there are a number of different models that may apply to a data set. These questions attempt to determine what positional accuracy models are important to this data set.

Figure 7.9: Typical view of the intelligent schema definition interface

The second and most important stage of the tool interface takes the form of a number of web-pages which aim to identify the important quality characteristics of the geospatial data set. A typical page is illustrated in figure 7.9. Here the user is encouraged to provide 'point-and-click' answers to questions about characteristics of the data set which may have implications for the quality schema. Information which has already been provided in answer to questions is highlighted in green and can be changed at any point. Some answers will trigger further, related questions to appear, as shown in 7.10 where the answers to the questions about an external reference system and about spatial variation in accuracy have triggered two further questions. It is not necessary to answer all the questions, although doing so is more likely to result in a well designed quality schema. In common with any web-site, pages and questions can be visited in any


Netscape: Quality definition: Positional accuracy
File Edit View Go Communicator Help
Bookmarks Location: http://m-duckham.geog.gla.ac.uk/schema/positional?positional_rmse•
What's Related
Main
Help
Continue
Skip this
End this

Positional accuracy

Question: Try to provide the following information about the positional accuracy issues likely to be relevant to your data.

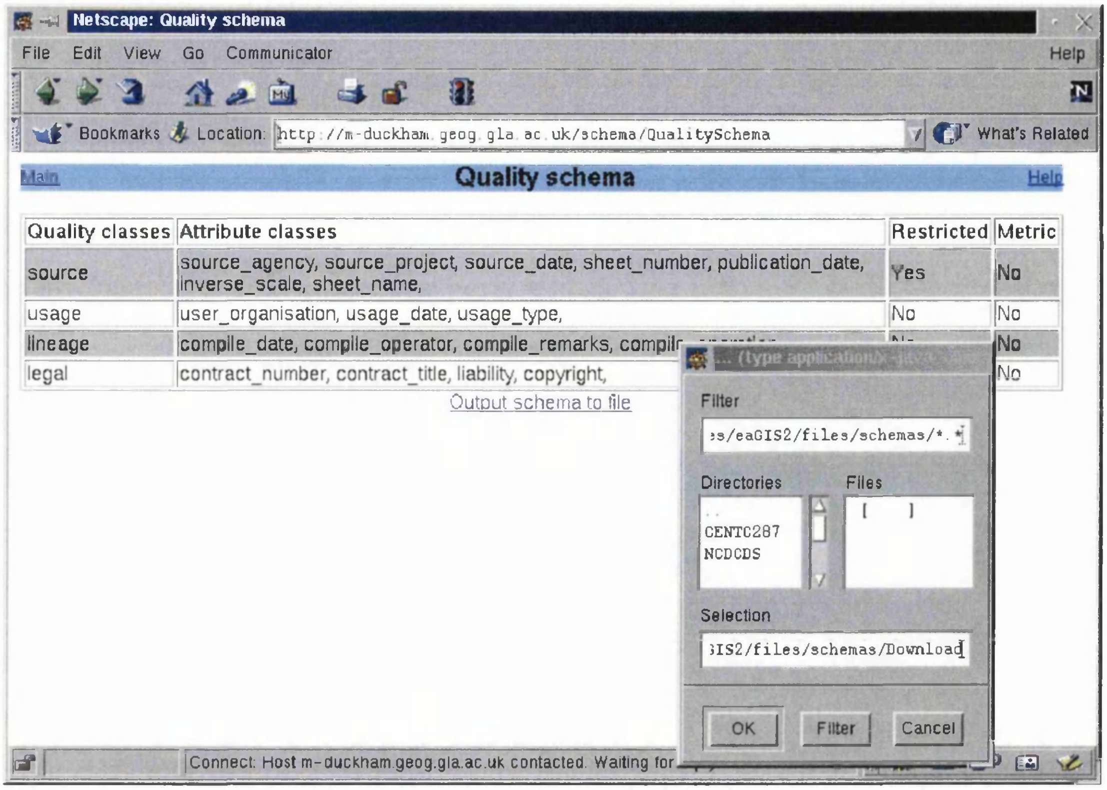
How important is it that the positional information matches some external reference system (eg the UK National Grid)?	[✓ Very Important]	Not particularly important	Don't know
Does information exist about this external positioning system?	Yes	[✓ No]	Don't know
How important are the *relative* positions of features in the data set?	[✓ Very important]	Not particularly important	Don't know
Do you expect the positional accuracy of the data set to be constant across the entire data set or to vary spatially?	Constant	[✓ Variable]	Don't know
Given that positional accuracy may vary spatially do you expect accuracy to exhibit anisotropy (vary more in one direction than another)?	Anisotropic	Isotropic	[✓ Don't know]

Explanation: In managing positional accuracy, there are a number of different models that may apply to a data set. These questions attempt to determine what positional accuracy models are important to this data set.

Figure 7.10: Some answers may trigger further questions

order, and if necessary can be left and returned to later. All the question pages feature a common look-and-feel, with the major page elements (title, questions, explanation and links) located in the same position on each page. At the bottom of every page three links provide control over this stage of the tool interface. The first link, identified by a yellow smiley face, is clicked when the user has completed all the questions they want to answer to their satisfaction. The second link, identified by an orange impassive face, indicates the user may have answered some questions, but is getting tired with the process and wants to move on, although he or she may come back to it. The third link, identified by a red unhappy face, indicates the user has had enough of this line of questioning and wants to move onto something else.

In the third stage of the intelligent schema definition tool user interface, all the information from the previous two stages is pulled together into a quality schema. After reviewing the current state of the quality schema, shown in figure 7.11, the user can either export this schema to the error-sensitive database, save it to file or return to the second stage if the schema is still incomplete. While the stages should be tackled roughly in order, there is considerable flexibility as to how the different stages are completed. The process inevitably entails a high degree of iteration, where a user reviews the quality schema and on the basis of this schema can continue answering further questions or perhaps amend the responses to previously answered questions.



http://m-duckham.geog.gla.ac.uk/schema/QualitySchema

Quality schema

Quality classes	Attribute classes	Restricted	Metric
source	source_agency, source_project, source_date, sheet_number, publication_date, inverse_scale, sheet_name,	Yes	No
usage	user_organisation, usage_date, usage_type,	No	No
lineage	compile_date, compile_operator, compile_remarks, compil	[illegible]	No
legal	contract_number, contract_title, liability, copyright,		No

Output schema to file

Filter: ɔs/eaGIS2/files/schemas/*.*

Directories: ..
CENTC287
NCDCDS

Files: [ ]

Selection: ɔIS2/files/schemas/Download

Connect: Host m-duckham.geog.gla.ac.uk contacted. Waiting for

Figure 7.11: Quality schema review page

7.3.2 Intelligent schema definition tool performance

The aim of this chapter has been to show how an intelligent schema definition tool might be developed quickly and in the context of the error-aware GIS. The actual performance of the resultant tool within the KC telecommunications database was generally positive. The hypertext tool interface is certainly much less complicated to use than the basic schema definition tool introduced in §6.4.1 and the JESS expert system does appear to be able to structure information based on the hierarchy of quality terms and user interaction with the hypertext interface. Some relatively minor difficulties arose. The tool was not designed to include meta-quality classes in the schema definition process, although arguably the same basic approach could allow the tool to be extended to address this need. Additionally, while it was originally the intention for the tool to tackle both abstractive and representative quality, since abstractive quality is often best treated as a component of the OO geospatial schema the final tool implementation is only capable of producing representative quality schema.

The reliance on a pre-existing hierarchy of terms did prove to be a more significant weakness in the tool design. The hierarchy of terms is extensive, but not necessarily exclusive. Since the quality terms are drawn from the literature, the schema produced by the tool tend to be a novel mix of ideas from existing quality standards and schema rather than genuinely introducing new quality schema. For example, the tool enables the redefinition of quality elements like 'lineage' each time the schema definition tool is used, but is not able to produce a schema definition that contains a significantly different definition of lineage from those already found in the literature. The tool could, therefore, be improved by the addition of a facility to append or modify the hierarchy of quality terms during error-sensitive schema definition process. Arguably, underlying these limitations is a more general problem with current data quality standards and research. As already noted in §2.1, most data quality standards can be linked in some way to the NCDCDS draft spatial data quality report, while research into data quality has yielded neither convincing arguments supporting the 'Famous Five' nor any radically different approaches to data quality. Consequently, the limitations evident in the intelligent schema definition tool may stem not from a lack of understanding of the OOA process, but from a lack of understanding and research into the elements of data quality itself. The possibility of further research to redress this lack of understanding is touched on in the final chapter (§10.2.1).

7.4 Conclusions

The intelligent schema definition tool illustrates how the error-aware architecture can be used to allow highly specialised software to be developed to meet highly specific needs. Database designers, such as Informed Solutions, have no background or experience in designing databases that incorporate elements of spatial data quality. The combination of a hypertext gateway to a traditional expert system provides a much 'softer' user interface than could be achieved without AI technology; an interface that is able to accommodate the inevitable inexperience of most database designers in this specialised area. Without such assistance, it seems unlikely that companies, such as Informed Solutions and KC, are going to incorporate into their spatial databases the basic structure necessary to support quality information.

The combination of expert systems and hypertext seems capable of simulating the OOA process, at least in within the narrow problem domain of designing error-sensitive schema. Hypertext allows users to 'surf' hypertext pages in a natural intuitive manner, jumping backwards and forwards and negotiating questions in whatever order is most convenient to them. The layout, questions and links of the tool interface can vary dynamically dependent on the information already provided. Alongside this apparently informal user interface, the information gathered can be processed within the formal semantics of the expert system. While the results are limited by the narrowness of the problem domain and the underlying hierarchy of quality terms, the tool does seem able to address the need for a intelligent system to assist conventional database designers in building error-sensitive schema. Having used AI technology to enable non-expert database designers to develop error-sensitive quality schema alongside conventional geospatial OO schema, the problem arose that in the case of KC, the spatial data quality information with which to populate the error-sensitive database very often did not exist. The development of a error-aware tool to address this next difficulty is the subject of the following chapter.

Chapter 8

Error aware GIS: quality capture

Despite growing acceptance amongst GIS companies and users of the importance of data quality, very often adequate data quality information for a data set will simply not exist. Further, limited expertise and financial restrictions are likely to mean most data producers do not feel in a position to compile such quality information about their spatial data. Experiences during this study suggested that there is, in fact, a high level of informal awareness of data quality issues amongst GIS professionals. UK initiatives such as the National Land Information Service (NLIS) and the Scottish Land Information Service (ScotLIS) are adding momentum to the prospect of integrated LIS in the UK (Smith 1996). Utility companies, such as KC, are well placed to benefit from and contribute to such initiatives. There is a concomitant awareness amongst GIS professionals that the management of data quality may be a vital element of this increased data integration.

Unfortunately, in the telecommunications industry as in so many other industries, this awareness does not yet translate into a desire to commit the high level of resources necessary to perform full quality assessments of their data. There exists no background in digital data quality management within industries like telecommunications, a situation both perpetuated by and leading to continued lack of investment in data quality. In order to break this cycle simple, effective and cheap methods of data quality capture are needed. This chapter looks at the development of a error-aware GIS tool able to assist GIS users with the capture of data quality information, information that otherwise would be discarded.

8.1 Induction and data quality

The error-aware quality capture tool developed during this research takes advantage of a flexible yet powerful AI technique for producing learning systems, called *induction*. Given an example data set, an *inductive learning algorithm* should be able to automatically deduce rules that embody the patterns in that data, rules which, hopefully, correspond to underlying processes governing the data. Induction is not new to GIS and has been used, for example, by Walker and Moore (1988) to identify relationships between spatial objects and help with an automated habitat classification process. Aspinall (1992) also used induction for habitat analysis, while Bennet and Armstrong (1996) used induction to assist with drainage feature extraction from a DEM. However, induction is not widely used in GIS, in part because it is better equipped to deal with discrete categorical example data and rules rather than inherently continuous spatially referenced GI. Happily, quality information is generally aspatial and consequently an inductive quality capture tool does not depend too heavily on the spatial nature of the example data. At the same time, the inductive learning algorithm developed during this research has been adapted to include spatial information wherever possible.

All induction algorithms share a number of features in common. In essence, we can define induction as operating upon a set of objects (for notational convenience ς-calculus objects in this case) $T = \{o | o = [l_k = a_k]^{k \in 1..n}\}$ called the *training set*. Each object in the training set also belongs to one category C_i out of a pairwise disjoint family of categories, $C = \{C_i | \forall o \in T \ (\exists! C_i \ o \in C_i)\}$ where $\forall C_i, C_j \in C \ (i \neq j \Rightarrow C_i \cap C_j = \varnothing)$. An inductive algorithm is able to build a decision tree that embodies the data in the training set using the following three steps, after Quinlan (1983).

i if the training set of objects is empty, $T = \varnothing$, we associate a new leaf in the decision tree arbitrarily with one of the categories $C_i \in C$.

ii if all objects in the training set belong to the same category $T \subseteq C_i$ then we create a new leaf in the decision tree with that category C_i.

iii else we select an attribute l and partition T into disjoint sets $T_j^{j \in 1..m}$ where T_j contains members with the jth value of the selected attribute, $T_j^{j \in 1..m} = \{o | \forall o \in T \quad o.l \rightarrowtail x_j\}$. A new decision node is then created to represent this decision and the algorithm is reiterated using each subset T_j.

Even in this stripped-down form, the induction algorithm is surprisingly powerful and will always successfully categorise a set of objects, provided there are no two objects that have identical attribute values but belong to different categories (Quinlan 1983) — ie as long as the statement $\forall o_x \in C_i \ \forall o_y \in C_j (i \neq j \Rightarrow o_x \nleftrightarrow o_y)$ holds. When this condition does not hold, it indicates that there is not enough attribute information about objects in different categories to tell them apart. In reality this condition will occasionally not hold and a practical inductive learning algorithm will usually need to resort to some heuristic, tackled in §8.2.3, to resolve such conflicts.

The actual performance of the inductive algorithm is dependent to a large extent on how the algorithm selects the attribute l with which to partition the set T in iii above. There are a range of different methods that might be used to achieve this, but one of the most efficient is to use information theory. The mathematical concept of information theory was first defined by Claude Shannon in the late 1940s (Shannon 1948). Shannon's information theory formalises the *information content* of a statement in terms of a number of binary digits or *bits* of information conveyed by the statement. For example, when tossing a coin, the value of knowing the outcome has an information content of 1 bit. However, if it is already known that the coin is biased, the value of knowing the *actual* outcome is reduced. The amount by which the value of knowing the outcome is reduced is related to the probability of each possible outcome. In the extreme case where the outcome is always, say, heads ($P(H) = 1$) the information content for any given coin toss is reduced to zero bits. In general, for a number of possible outcomes v_i each with probability $P(v_i)$, the information content I of knowing the outcome is given by equation 8.1 (Russell and Norvig 1995).

$$I(P(v_1), ..., P(v_n)) = \sum_{i=1}^{n} -P(v_i) log_2 P(v_i) \tag{8.1}$$

Information content can be used as a method for systematically selecting one attribute from a range of possible attributes to use in partitioning the set of objects T. For each possible partition of the set T with respect to a particular attribute l, the information gained by using that attribute can be calculated. This calculation involves estimating a set of probabilities associated with the partitioned sets T_i as a function of the ratio of objects in each partitioned set to the total number of objects (Russell and Norvig 1995). The attribute that results in the largest information gain should be the optimal attribute with which to partition the set T, since it provides more information about the decision tree than any other attribute.

8.1.1 Induction example

It is possible to provide an example of the induction algorithm in operation. The example is based on experience with the KC telecommunications database. The example contains five ς-calculus objects each with just two categorical attributes, density and type, shown in equation 8.2.

$$\begin{aligned} T = \{o_1 &= [density = [\text{"dense"}], type = [\text{"pole"}]], \\ o_2 &= [density = [\text{"dense"}], type = [\text{"kiosk"}]], \\ o_3 &= [density = [\text{"sparse"}], type = [\text{"cabinet"}]], \\ o_4 &= [density = [\text{"sparse"}], type = [\text{"pole"}]], \\ o_5 &= [density = [\text{"sparse"}], type = [\text{"kiosk"}]]\} \end{aligned} \tag{8.2}$$

In the pilot assessment, high feature density tended to be associated with poor positional accuracy. Densely packed features on the plant-on-plan maps are often displaced for cartographic reasons in addition to being harder to understand and digitise. Such lower positional accuracy often persists through to the digital data. In some cases, however, positional accuracy tended to be low regardless of feature density. In particular, cabinet features (see table 7.1) explicitly use a symbology that obscures any precise location. Although the induction algorithm can have no 'understanding' of these sorts of processes, the induction algorithm is sensitive to data exhibiting these types of relationships. When shown a data set where low accuracy and high feature density are coincident it should be able to derive a rule or set of rules that embody this relationship.

The five objects in the set T have been categorised into low (C_l) and high (C_h) accuracy features, shown in equations 8.3 and 8.4 respectively. The categories are broadly speaking as would be expected according to each object's spatial density attribute, with one object, o_3 a cabinet, exhibiting low accuracy C_l despite its low spatial density.

$$C_l = \{o_1, o_2, o_3\} \tag{8.3}$$

$$C_h = \{o_4, o_5\} \tag{8.4}$$

The induction process for this example is illustrated in table 8.1, which expands on each step of the induction process. The result of this induction process is a simple decision tree, shown in figure 8.1. The decision tree is automatically derived from the induction algorithm, but is a reflection of the more general processes behind the training set data. Having used induction to build a decision tree, it is possible to then categorise objects outside the original training set. For example, the object $o_6[\mathit{density} = [\text{"dense"}], \mathit{type} = [\text{"cabinet"}]]$ was not part of the training set, but an examination of the decision tree in figure 8.1 reveals that such an object would be categorised as having low accuracy.

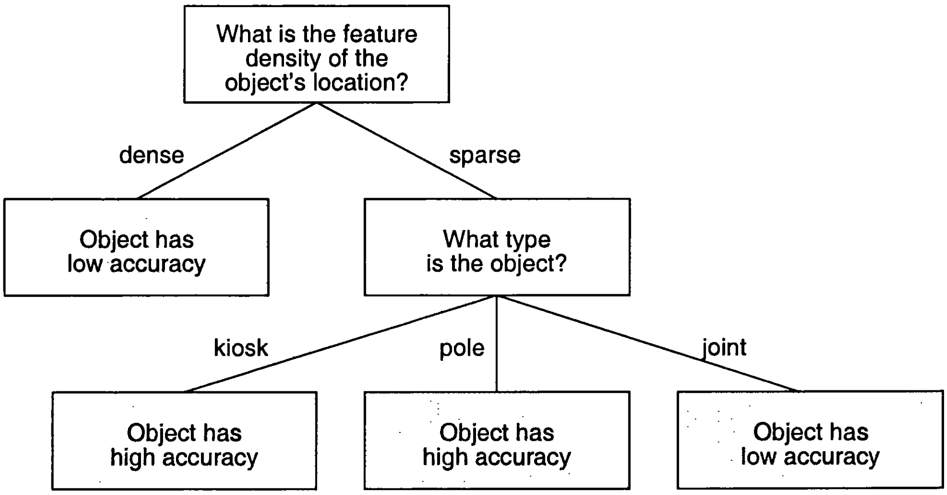


Figure 8.1: Example induction process results

	Induction step	Details
0.1	Start induction process with T, C_l and C_h	$T = \{o_1, o_2, o_3, o_4, o_5\}, C_l = \{o_1, o_2, o_3\}, C_h = \{o_4, o_5\}$
1.1	Check for empty T	$T \neq \varnothing$
1.2	Check whether T contains objects of only one category	$T \nsubseteq C_l \qquad T \nsubseteq C_h$
1.3	Partition T with first attribute, type.	$T_p = \{o_1, o_4\} \quad T_k = \{o_2, o_5\} \quad T_j = \{o_3\}$
1.4	Calculate information gain for type	$Gain(type) =$ $I(\frac{3}{5}, \frac{2}{5}) - (\frac{2}{5}I(\frac{1}{2}, \frac{1}{2}) + \frac{2}{5}I(\frac{1}{2}, \frac{1}{2}) + \frac{1}{5}I(\frac{1}{1}, \frac{0}{1})) = 0.171$ bits
1.5	Partition T with second attribute, density.	$T_d = \{o_1, o_2\} \quad T_s = \{o_3, o_4, o_5\}$
1.6	Calculate information gain for density	$Gain(density) =$ $I(\frac{3}{5}, \frac{2}{5}) - (\frac{3}{5}I(\frac{2}{3}, \frac{1}{3}) + \frac{2}{5}I(\frac{2}{2}, \frac{0}{2})) = 0.420$ bits
1.7	Create new decision node using attribute with highest information gain and reiterate process.	Reiterate with T_d (2.1) and T_s (3.1)
2.1	Check for empty T_d	$T_d \neq \varnothing$
2.2	Check whether T_d contains objects of only one category	$T_d \subseteq C_l$ so iteration terminates with new leaf
3.1	Check for empty T_s	$T_s \neq \varnothing$
3.2	Check whether T_s contains objects of only one category	$T_s \nsubseteq C_l \qquad T_s \nsubseteq C_h$
3.3	Partition T_s with first attribute, type.	$T_{s,p} = \{o_4\} \quad T_{s,k} = \{o_5\} \quad T_{s,j} = \{o_3\}$
3.4	Calculate information gain for type	$Gain(density, type) =$ $I(\frac{2}{3}, \frac{1}{3}) - (\frac{1}{3}I(\frac{1}{1}, \frac{0}{1}) + \frac{1}{3}I(\frac{1}{1}, \frac{0}{1}) + \frac{1}{3}I(\frac{1}{1}, \frac{0}{1})) = 0.918$ bits
3.5	Partition T_s with second attribute, density.	$T_{s,d} = \varnothing \quad T_{s,s} = \{o_3, o_4, o_5\}$
3.6	Calculate information gain for density	$Gain(density, density) =$ $I(\frac{3}{3}, \frac{0}{3}) - (\frac{1}{3}I(\frac{1}{1}, \frac{0}{1}) + \frac{1}{3}I(\frac{2}{2}, \frac{0}{2})) = 0.000$ bits
3.7	Create new decision node using attribute with highest information gain and reiterate process.	Reiterate with $T_{s,p}$ (4.1), $T_{s,k}$ (5.1) and $T_{s,j}$ (6.1)
4.1	Check for empty $T_{s,p}$	$T_{s,p} \neq \varnothing$
4.2	Check whether $T_{s,p}$ contains objects of only one category	$T_{s,p} \subseteq C_h$ so iteration terminates with new leaf
5.1	Check for empty $T_{s,k}$	$T_{s,k} \neq \varnothing$
5.2	Check whether $T_{s,k}$ contains objects of only one category	$T_{s,k} \subseteq C_h$ so iteration terminates with new leaf
6.1	Check for empty $T_{s,j}$	$T_{s,j} \neq \varnothing$
6.2	Check whether $T_{s,j}$ contains objects of only one category	$T_{s,j} \subseteq C_l$ so iteration terminates with new leaf

Table 8.1: Example induction process iterations

8.2 Optimising the induction algorithm

While naïve, the example in §8.1.1 above does illustrate how the core induction algorithm can operate for a very simple quality assessment. However, before the induction algorithm can be considered for practical application it is necessary to address some more pragmatic implementation issues. The remainder of this section is devoted to a number of optimisation routines that were implemented within the inductive quality capture tool as part of the Java source code, which can be found in appendix C.

8.2.1 Support for non-categorical attributes

A common feature of all induction algorithms is that they are essentially categorical and operate only upon qualitative attributes. While a categorical induction algorithm can be useful in many contexts, most spatial data demands some quantitative capabilities. For example, imagine three polygon objects with 'area' attribute values of 10.0m^2, 10.1m^2 and 100.0m^2 used to train the inductive algorithm discussed above. The algorithm would by default treat each area attribute value as a *separate* category. This is technically undesirable since treating continuous attributes as discrete attributes quickly results in large fragmented decision trees riddled with decisions that yield minimal information gain. However, it is also semantically undesirable since we would probably intuitively expect 10.0m^2 and 10.1^2 to appear in a different category to 100.0m^2, but the same category as each other.

To combat this tendency, the inductive quality capture tool uses a heuristic to categorise quantitative attributes, an approach also used by Walker and Moore (1988). The heuristic uses simple measures of spread to categorise the population of values for a particular numerical attribute into up to five separate categories. In the example above, the heuristic should create, say, two new categories of 'areas of less than or equal to 30m^2' and 'areas of greater than 30m^2'. In deciding whether an attribute is categorical or quantitative, the inductive quality capture tool is able to consult the geospatial object schema. Any attribute class that inherits from 'string' should be qualitative, whilst in the case of the KC database sub-classes of 'integer' and 'real' attributes were quantitative. Unfortunately, the latter rule may not always hold, so a more sophisticated heuristic would be needed for databases where numerical classes are used for categorical information.

8.2.2 Spatial parameters

Since the induction algorithm cannot deal directly with quantitative attributes it also cannot deal directly with many spatial parameters, such as coordinate location (although potentially it would be able to deal directly with topological information). Consequently, spatial parameters are included in the induction process by producing summary spatial statistics for every object and then treating these as categorised numerical attributes, as discussed above. For example, an estimate of feature density is automatically calculated for objects in the training set. This attribute can then be categorised by the heuristic described above. In the same way, other spatial parameters can be incorporated into the induction process using summary spatial statistics such as spatial complexity, distance, area or length.

8.2.3 Majority classification

There are two points within the induction process when arbitrarily categorisations need to be used. The first point occurs when the training set for a particular iteration is empty, $T = \varnothing$ (§8.1). $T = \varnothing$ occurs when a training set has no objects that exhibit a particular value for an attribute being used to partition that training set. The second point, as suggested in §8.1, occurs when conflicting information exists and two objects with identical attributes belong to different categories. In reality both cases do occur, and the inductive quality capture tool uses a majority classification heuristic to provide a basis for an otherwise arbitrary categorisation. By looking at the range of different outcomes in the training set, or in the training set of the parent iteration in the case of $T = \varnothing$, the inductive quality capture tool assigns a new decision with the most populous outcome in that set. The assumption is that, on balance and in the absence of better information, the category with the majority of instances is the more likely outcome.

8.2.4 Overfitting

The inductive learning algorithm is far from infallible. A problem common to learning algorithms generally occurs when a learning algorithm infers meaningless patterns from a data set, termed *overfitting* (Russell and Norvig 1995). In particular, if the training set is unrepresentative or too small, the algorithm is much more likely to derive rules that relate to no particular processes or are entirely coincidental. In order to provide some guidance as to whether the training process has been successful, the induction algorithm reserves a portion of the training set, approximately one-third of the data, for cross-validation purposes. Having produced a decision tree using two-thirds of the training set, the decision tree is then used to deduce the correct categorisation for the remaining third of the training set. These results can be compared with the actual categorisations in the reserved third of the training set, to provide a guide as to how successful the training process was. Low cross-validation accuracies indicate the training set is unrepresentative or too small and the training set needs to be extended.

8.2.5 Spatial inference

Following training, the decision tree should be able to make reasonable decisions regarding the quality of the geospatial objects from which the training set was drawn, even ones with attribute values that the decision tree has not encountered during training. Inevitably, the trained decision tree may come across objects that it cannot categorise because a key attribute value was not encountered within the training set. Assuming the training set was representative, such situations should be infrequent. Rather than just abandon the automatic data quality assessment for these objects, the induction algorithm attempts to match the problem object with a similar nearby object for which the attribute can be resolved. By further assuming the existence of spatial autocorrelation, it should be reasonable to substitute the nearest similar object for the problem object if the decision process stalls occasionally. Unfortunately, the assumption of autocorrelation does not always hold. Many geographic features are not autocorrelated and in such cases the spatial inference mechanism should not be used. However, autocorrelation is undoubtedly an important factor in a wide range of geographic phenomena (Tobler 1970) and the assumption of autocorrelation will normally be a valid one. As an illustration of the spatial inference mechanism, in the example in §8.1.1 the trained decision tree in figure 8.1 would have difficulty with an object $o_7 = [\mathit{density} = [\text{"very dense"}], \mathit{type} = [\text{"cabinet"}]]$, since the attribute value "very dense" did not occur in the training set. In such a case, the spatial inference mechanism would be free to substitute the density attribute value of the nearest "cabinet" object. If such an object exists and the attribute is spatially autocorrelated, by virtue of being nearby the object is more likely to have the density attribute value "dense" rather than "sparse" and consequently ought to be a reasonable substitute.

8.2.6 Parallel induction

The inductive learning algorithm as described so far would be very useful for deriving decision trees which could be used infer the quality of geospatial objects in terms of a *single* quality element. For example, the algorithm could train a decision tree to infer accuracy *or* to infer lineage for a data set. However, it is very likely that for a given set of geospatial objects, accuracy, lineage and indeed any other quality element may vary independently of each other. Further, a particular quality element may have a number of attributes that also vary independently. The categorisation task can be viewed as a number of parallel induction tasks based on a training set categorised according to each attribute on each of the quality elements present in the training set.

This study developed a simple extension to the conventional induction algorithm outlined above, which is able to perform the induction process in parallel for several categorisations. In common with the conventional induction algorithm, the parallel induction algorithm uses a single training set and produces a single decision tree. However, at any given induction step the attribute used to partition the training set can be selected according to the total information gain produced by that attribute. Since information content is additive, the total information gain can be calculated from the summed information gain for each individual category family. Attributes can then be selected on the basis of maximal information gain across a range of categories. The result is that while a decision may be sub-optimal for an individual category at an individual iteration, overall the system still results in an efficient decision tree that should be able to resolve a number of categorisations in a single step, effectively performing several categorisations at once.

8.3 Implementation results

An inductive quality capture tool was implemented using the inductive learning algorithm outlined above. The tool offers a simple interface to help error-aware GIS users to incorporate representative quality information into their data set during data capture. The tool is restricted to deducing representative quality, since the information available to the algorithm is based on individual objects rather than on classes of objects which would be required to assess abstractive quality. The quality capture tool is intended to work alongside conventional spatial data capture streams. In particular, it is aimed at legacy data capture projects, such as that undertaken by KC.

8.3.1 Choosing the training set

Use of the inductive quality capture tool begins with a pilot assessment of the quality of a small area of the legacy map data being captured. This pilot quality assessment forms the training set for the inductive learning algorithm. In the case of KC, it proved entirely feasible to derive a picture of the history and accuracy of a pilot quality assessment of the KC data without the need for resurvey. Simply by looking through the contract documentation, familiarity with the source maps and by talking with the KC, SDS and Informed Solutions employees it was possible to produce a credible pilot quality assessment. Perversely, a significant body of quality information associated with legacy paper maps will usually be lost during the migration to digital mapping. Lineage information on the provenance of maps and map features is well known to engineers used to handling those maps. Levels of accuracy precision and detail are often implied by the physical limitations of the map, limitations which do not apply once the map is digitised. In other cases it might be necessary to embark upon relatively expensive resurvey. However, in the case of legacy data capture the value of such informally developed quality information should never be discounted.

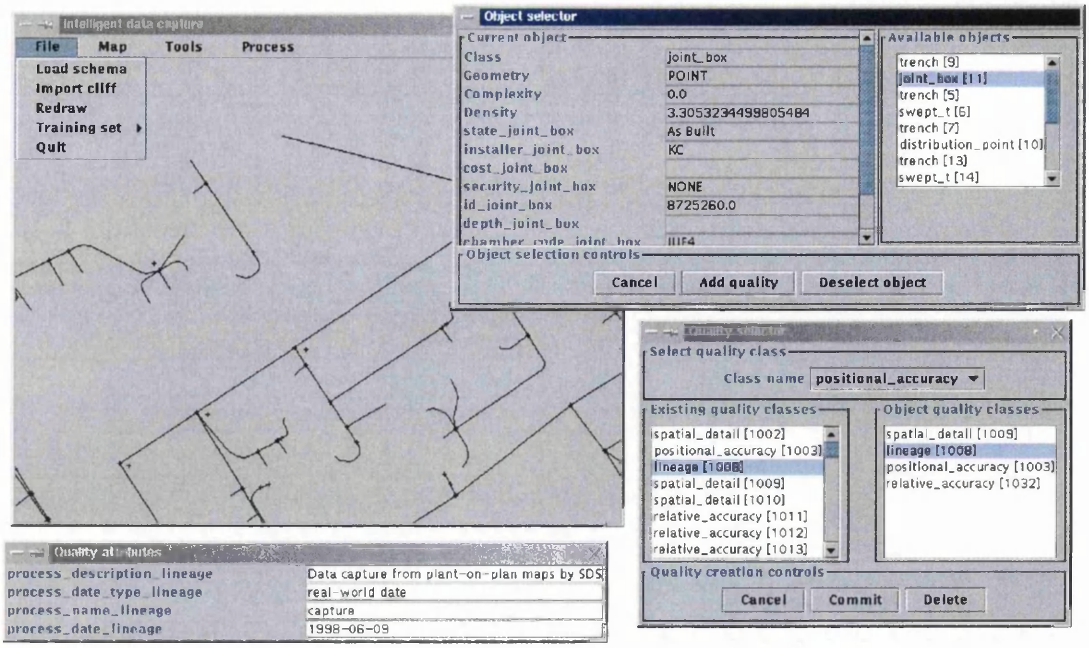
As already mentioned in §2.3 this study makes the simplifying assumption that all geographic information refers directly to the real world. This assumption is at its weakest when dealing with the data captured from legacy paper maps. The paper maps themselves are real world objects, and information will be lost via the processes abstraction and representation both during the initial capture of geographic information for the paper map and when the information on the paper map is recaptured digitally. In the future legacy data capture from paper maps is likely to be a rarity (see§2.3), so the conceptual problems posed by the capture information about the real world from indirectly paper sources should dissolve.

It is worth noting that the pilot quality assessment used for the training set need not be a single contiguous geographic area. For the purposes of the core induction algorithm, the pilot quality assessment can operate using a training set composed of features that are geographically dispersed across the study area. However, two practical considerations militate against using such dispersed training sets. First, it will usually be much more efficient from the point of view of data capture to perform the pilot quality on a single contiguous sub-set of the study area rather than perform a piecemeal assessment over the entire study area. Second, the induction optimisation routines may assume that the training set is not spatially dispersed. In particular, the calculation of feature density mentioned in §8.2.2 assumes that the spacing of features in the training set is characteristic of the study area generally. If the training set is spatially dispersed this assumption will not hold.

8.3.2 Quality capture

The inductive quality capture tool interface, shown in figure 8.2, is similar to the data browser discussed in §6.4.2. The tool acts as a sort of data import filter, allowing the pilot data set to be imported and quality assessment information added to this pilot data set. The pilot data set for this study was drawn from the KC data supplied by SDS in the form of CLIFF files, the intermediate text file format used by the KC project for data transfer. The tool as depicted in figure 8.2 has four linked windows. The main map window, in the top left of figure 8.2, shows a portion of the pilot data set and four menus needed to operate the tool. The 'file' menu offers basic tool functions like quit, redraw and load schema. The 'map' menu allows the current state of a pilot quality assessment to be saved to file. The 'tools' menu offers 'zoom', 'pan' and 'select' tools for the map window. Finally, the 'process' menu controls the training and operation of the induction algorithm described above. Having loaded a basic schema into the tool, which will happen automatically if one of the schema definition tools described in the previous two chapters has been used, the pilot CLIFF data set can be imported into the main map window. This pilot data set is then annotated with quality information gathered, in this case, from the informal sources described above. In order to annotate the pilot data set with quality information a further three different types of window are needed, shown in figure 8.2. Clockwise from the main map window, a geospatial object selection window displays the attributes of geospatial objects selected from the main map window, while a selected quality object and a quality attribute window allow individual quality objects to be associated with selected geospatial objects.

At this point no data has yet been written to the Gothic database. It is all stored as persistent serializable Java objects accessible to the quality capture tool as well as any other Java applications. Once the quality assessment information has been added to the pilot data, this information can be used as the basis of a wider quality assessment. The tool uses the pilot data set as a training set for the inductive learning algorithm. The geospatial data in the training set is categorised into a number of separate category families according to its associated quality objects' attributes. Using this training set the quality capture inductive learning algorithm looks for patterns in the geospatial data that imply patterns in the quality data. The product of the induction algorithm is a decision tree tailored to the particular features of the telecommunications data being captured. Once created, this decision tree can be applied to the remainder of the data capture process, automatically deducing quality information. Both geospatial information and deduced quality infor-



Class	joint_box
Geometry	POINT
Complexity	0.0
Density	3.3053234499805484
state_joint_box	As Built
installer_joint_box	KC
cost_joint_box	
security_joint_box	NONE
id_joint_box	8725260.0
depth_joint_box	
chamber_code_joint_box	IIIF4

trench [9]
joint_box [11]
trench [5]
swept_t [6]
trench [7]
distribution_point [10]
trench [13]
swept_t [14]

Class name positional_accuracy

spatial_detail [1002]
positional_accuracy [1003]
lineage [1008]
spatial_detail [1009]
spatial_detail [1010]
relative_accuracy [1011]
relative_accuracy [1012]
relative_accuracy [1013]

spatial_detail [1009]
lineage [1008]
positional_accuracy [1003]
relative_accuracy [1032]

process_description_lineage	Data capture from plant-on-plan maps by SDS
process_date_type_lineage	real-world date
process_name_lineage	capture
process_date_lineage	1998-06-09

Figure 8.2: Pilot quality assessment

mation can be used to populate the error-sensitive database again via the component architecture already explored in chapter 6.

The discussion in §8.2.4 highlighted the problems with overfitting where unrepresentative or small data sets infer meaningless patterns. Following training the tool interface immediately displays a dialogue box that reports the cross-validation accuracy of the training process, along with some guidance as to what that accuracy means and whether the pilot data set should be extended. As a rule of thumb, this study suggested that the best results were produced by pilot assessments covering between 5 and 10% of the total number of features. Assessments of less than 5% of the total number of features were much more likely to be unpredictable or unreasonable. As a last line of defence, however, all automatically generated quality objects are associated with a meta-quality object that reports both the fact that the quality object was automatically generated and a simple justification of the inductive process leading to the decision to use that quality object. This information provides the basis of 'quality audit', so that following the quality capture process the original data sets can still be retrieved, and automatically derived quality information can always be identified from manually derived quality information.

8.4 Conclusions

The failure to collect quality information is a self-perpetuating reason for a widespread failure to incorporate quality management procedures in digital data capture projects. The already high cost of collecting spatial data, coupled with the high levels of competition in industries like telecommunications, mean that such industries are unlikely to embrace quality management of geospatial data on short-term financial grounds alone. The value of the inductive quality capture tool within the error-aware GIS architecture is that it maximises the efficiency of quality assessment, requiring only a small fraction of the information produced during full quality assessment to operate. Potentially, introducing low-cost quality capture techniques is the first step in breaking the cycle that prevents companies collecting and using data quality information for geospatial data sets.

The induction algorithm at the heart of the inductive quality capture tool uses a pilot data set to infer general rules relating quality to geospatial objects. Individual geospatial objects in the pilot data set are categorised according to their representative quality objects. The induction algorithm is able to build a decision tree based on the spatial and aspatial characteristics of the geospatial objects, while performing a self-check on a reserved portion of the pilot data set. Assuming the self-check indicates the training process has been successful, this decision tree can be used to automatically infer quality more generally across the geospatial data set.

There is a question mark, however, over how the results of such an inductive quality capture exercise should be interpreted. Even assuming the training set is representative of the full data set, it is a moot point as to what extent the quality information produced by the induction algorithm can be considered 'correct'. There is a dearth of research addressing the reliability of quality assessments, and it is difficult to see how the reliability of quality information could be tested using conventional experimental methods. Conceivably, a comparison between repeated independent quality assessments would yield an idea of how accurate a particular quality assessment procedure is. Such experiments have not been performed and, given the difficultly in encouraging companies to perform a single quality assessment, it is implausible to expect the same companies to perform a statistically representative set of quality assessments in order to derive meta-quality information about the reliability of their quality assessment procedure. In the absence of any extensive independent quality assessment with which to compare the results of an induction process it is difficult to provide an unequivocal statement of how accurate the inductive quality capture process actually is. At the very least, the results of the implementation process in this chapter suggest that induction when applied to automated quality capture can produce reasonable results. The penultimate chapter looks at how the error-aware architecture can help in actually using quality information such as that produced by the data quality capture tool.

Chapter 9

Error-aware GIS: quality use

The intelligent schema definition tool, explored in chapter 7, and the inductive quality capture tool, explored in chapter 8, illustrate the the efficacy of the error-aware GIS architecture in assisting data producers with designing databases and capturing spatial information with data quality built in. However, a discussion of the application of the error-aware GIS architecture to the KC telecommunications database would be incomplete without looking at how the error-aware architecture might be used to help data users actually apply this data quality information. This chapter explores the development of a data integration and accuracy visualisation tool that enables Internet-based access to the error-sensitive database. In particular, the tool allows data users to assess the accuracy of positional information provided by the error-sensitive database. Unlike the previous error-aware tools in chapters 7 and 8, the tool presented in this chapter does not utilise AI technology. However, it does utilise a highly domain specific design, which is dependent on the error-aware architecture and is consequently still treated as part of the discussion of error-aware GIS software. The chapter begins with a look at the importance of data integration, followed by a discussion of the integration and visualisation tool interface. The tool architecture and the error model embedded within the tool are explored in §9.2 and §9.3 respectively.

9.1 Data integration and quality mapping

The introduction to chapter 8 posited the prospect of integrated LIS as one reason for the increased awareness of data quality issues amongst utilities companies, such as KC, and data suppliers, such as SDS. Data quality management has been identified as an important component of any data integration task (Ehlers et al. 1991; Shepherd 1991; Vckovski 1998). Flowerdew (1991) goes as far as to suggest that executing the difficult task of GIS data integration may be salutary in that it forces GIS users to face up to issues of data quality. Without basic data quality information regarding data source, collection methods, map projections, generalisation, transformations, accuracy and precision contradictions and conflicts between different data sets can present a major impediment to data integration. As such integration becomes more common, error-sensitive GIS should provide the raw materials with which to manage the integration process.

However, even without undertaking ambitious data integration projects, such as NLIS and ScotLIS touched on in the introduction to the previous chapter, it is possible to use the error-aware GIS architecture to provide a degree of lightweight integration. For example, when undertaking maintenance or installation work on site, construction and utility companies need to ensure their work does not damage or interfere with existing infrastructure. Very often, clues to the precise locations of other infrastructure may be evident on site, perhaps from the pattern of previous excavations left on a tarmac surface. Even if such evidence is available, work still needs to proceed with care. Unfortunately, site plans from a GIS are unlikely to indicate where thematic and locational information are more uncertain, and so where more care needs to be taken. This chapter describes the development of a simple data integration and accuracy visualisation tool that can help address some of these needs.

The integration and visualisation tool is designed to provide quick and simple mapping able to reflect the locational uncertainty associated with information in the telecommunications database. The tool aims to allow individuals responsible for planning or carrying out work in a particular geographic area to access information about the telecommunications features already in that area. The information made available by the tool includes both the location of these telecommunications features and the accuracy of location of those features. The tool is also designed to allow new features to be integrated with the selected telecommunications data, and the effects of these features upon the accuracy of location visualised. In this way, the tool aims to highlight areas where locational information is particularly uncertain and as a consequence the risk of damaging existing infrastructure is particularly high.

9.2 Internet-based tool design

The data integration and accuracy visualisation tool uses an Internet-based architecture similar to that employed by the schema definition tool interface in §7.3. In §7.3, the combination of webserver and Java servlet technology was advantageous because it supported the development of a sophisticated hypertext interface. In the case of the integration and visualisation tool the emphasis is on the use of Internet technology to allow widespread, flexible access to the error-sensitive database. While the use of Internet technology means potentially anyone on the Internet could be allowed to access the tool's web-site, the constraints of commercial confidentiality, legal restrictions and the cost of data production mean that the tool would be likely to be restricted to privileged or trusted users. In either event, using an Internet-based architecture can allow flexible access for users both within KC and in other utility and construction companies connected to the web.

9.2.1 Encapsulation of error models

The importance of fitness for use to error handling in GI science has been emphasised from the beginning of this thesis as the central aim of data quality management. A key component of fitness for use is the clear apportionment of responsibility for data quality between data producers and data users. While it is the data user's responsibility to use data appropriately, the data producer needs to supply sufficient quality information along with data to ensure a user is able to assess fitness for use (§1.2). However, it is increasingly argued that the data producer has a further responsibility to provide appropriate tools for assessing fitness for use (see Goodchild 1998; Goodchild et al. 1999; Shortridge and Goodchild 1999). Goodchild et al. (1999) show how a simple Monte-Carlo simulation can be encapsulated with data downloaded from the Internet using Java applets, to assist users with understanding the quality implications of a data set.

This concept of 'bundling' data and functionality together is highly compatible with the concept of an error-aware GIS. Just as the error-sensitive GIS aims to encapsulate error-sensitive behaviour within every database object, so the error-aware GIS aims to encapsulate error-aware behaviour within every application that accesses the error-sensitive database. Here again, the convergence between OO and other trends within IS is evident: there is a clear analogy between the OO concept of an object as state plus behaviour and moves toward the encapsulation of geospatial data alongside geospatial functionality. The integration and visualisation tool presented below follows on from the concept of encapsulated error models in Goodchild et al. (1999). The Internet-based architecture can be used to respond to client requests both with the requested data and the tools to process that data.

9.2.2 Internet architecture

The tool architecture used for the data integration and accuracy visualisation tool extends the basic distributed system architecture introduced in chapter 6 and used by the other error-aware tools in chapters 7 and 8. The architecture for the data integration and accuracy visualisation tool is illustrated in figure 9.1 below. The Gothic error-sensitive database, running on a Sun SPARCstation 20, provides error-sensitive objects and services to the Java RMI middleware acting as a client. The Java middleware is then free to offer Java-based distributed services to any other Java applications, in this case to Java servlet objects operating within a JVM that itself is within an Apache web-server. Both the Java servlets and the Java middleware happen to run on the same Intel Pentium (i586) computer, although within different JVM that could equally be located on different physical machines. The Java servlets are then able to offer error-sensitive services via the web-server to other client applications such as web-browsers across the Internet more generally.

The difference between this architecture and that used by the error-aware tool in chapter 7 is the addition of a Java *applet* to the web-browser client in figure 9.1. Related to Java servlet technology, Java applets are Java application objects which operate within a web-browser's JVM

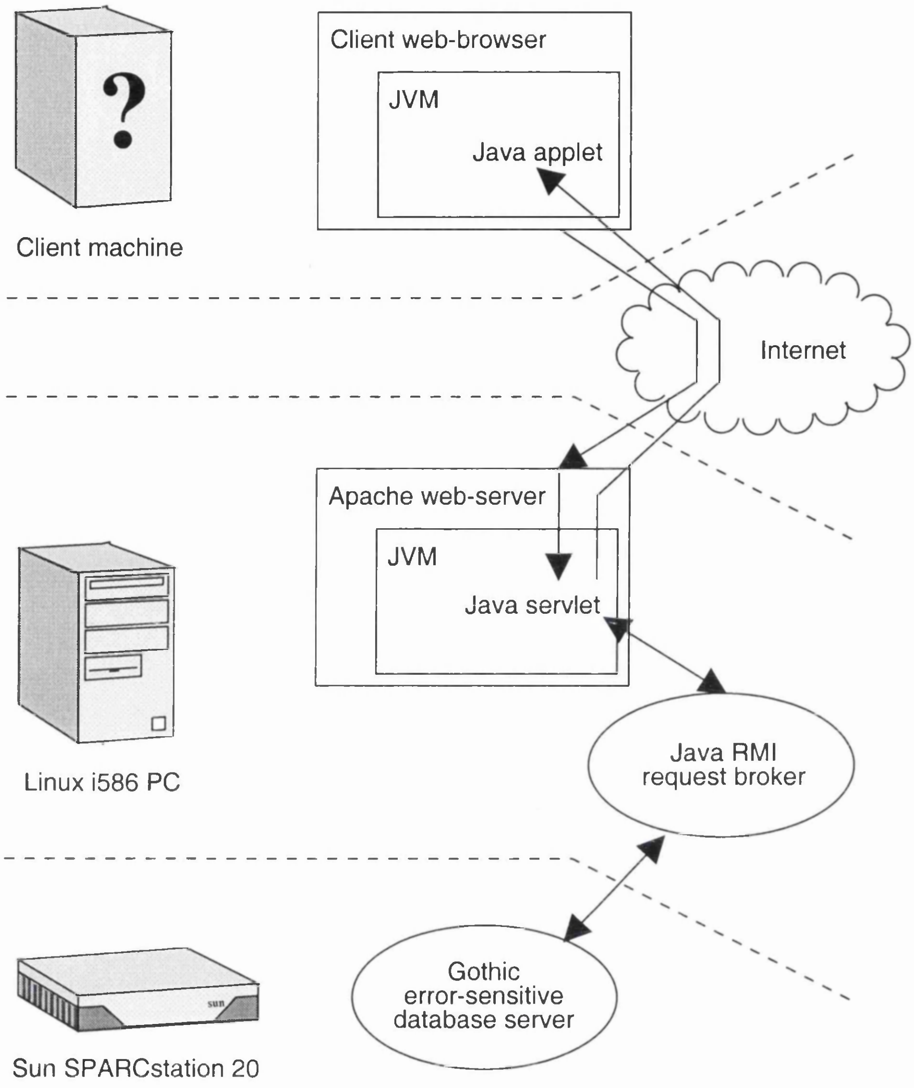


Figure 9.1: Internet-based tool architecture

using a specialised applet API. Unfortunately, Java applets have been caught up in something of a battle for domination of the Internet. The result has been some severe incompatibilies between the different JVM embedded in web-browsers produced by the main protagonists, Sun, Microsoft and Netscape. Careful programming is therefore a necessary part of Java applet development to ensure compatibility with the majority of web-browser JVM. However, using applets allows the additional functionality needed by the data integration and visualisation tool. The tool interface, described in §9.4, allows sophisticated spatially references user interaction that could not be supported by HTML and servlets alone.

9.3 Error propagation model

In order to produce accuracy maps for the integration and visualisation tool it is necessary to adopt a specific error model able to integrate and propagate positional accuracy information from the different sources into the combined data set. Early versions of the software developed during this research attempted provide error propagation as a component of the error-sensitive GIS functionality. This initial approach was not unsuccessful: the error-sensitive GIS code presented in appendix C incorporates the C code for simple variance and Monte-Carlo error propagation algorithms. However, it quickly became apparent that this approach was flawed. The problem arose that there are simply far too many error models and error propagation techniques to ever implement them all in one GIS. Monte-Carlo simulation is undoubtedly the leading contender for a generic error propagation technique, and despite humble beginnings can now be regarded as a sound, "well-understood and respectable" mathematical technique (Green 1995, p185). However, in addition to Monte-Carlo simulation, variance propagation has already been cited as an important error propagation methodology in §2.2.1. The same section mentions the influential work of Heuvelink (1998), who makes a compelling case for using analytical error propagation techniques *in addition* to Monte-Carlo simulation. Heuvelink's argument is a pragmatic one: while analytical techniques are computationally efficient and reflect the underlying mechanisms of error propagation, in many cases Monte Carlo simulation is the only practical error-propagation technique available.

This lack of a single error propagation technique suitable for all situations can be addressed by encapsulating error propagation models along with geospatial data. By tailoring Java applets and servlets to offer error propagation functionality, clients accessing data using the Internet can benefit from error propagation specific to that data set. While Goodchild et al. (1999) made use of Monte-Carlo simulation in their work, in the context of an Internet environment this approach is open to criticism. Web-browsers and web-servers are by nature complex multi-threaded environments that are not well suited to the sorts of computationally intensive algorithms needed by Monte-Carlo simulation. There is an argument that the continuing advances in low-cost computing power make concerns over processing speed at worst transitory; less than 10 years ago spatial Monte-Carlo simulation itself was beyond the computational resources of many personal computers users. However, irrespective of computer power the central reason for using Monte-Carlo simulation, that it is widely applicable to error propagation problems, is not critical in the case of web-based encapsulated error models. Since the error propagation functionality is controlled by the data producer and tailored to the data, there is no need to provide generic error propagation algorithms. Consequently, the integration and visualisation tool is able to take advantage of a far less computationally intensive deterministic analytical locational error model.

9.3.1 Locational error model

Given that computationally intensive Monte-Carlo simulation does not translate well to complex web-based environments where there are already many demands upon the system, it remains to define an analytical error model that can be used in its place. The discussion in §2.2.1 compared many of the leading models. A distinction was drawn between conceptually attractive informal error models, such as the ϵ band of Chrisman (1983), and the stochastic models that tend toward counter intuitive results, such as the 'bow-tie' approach of Ehlers and Shi (1996). The approach of Leung and Yan (1998) was presented as a useful compromise between these two extremes. The basic premise of Leung and Yan (1998) is that positional error can be modelled by a circular, although univariate, normal distribution. For any point (a, b) the probability density function (PDF) for any point (x, y) being the true location of (a, b) is given by equation 9.1 (Zelen and Severo 1965).

$$f(x,y) = \frac{1}{\sigma\sqrt{2\pi}}\exp^{-\left[\frac{(x-a)^2+(y-b)^2}{2\sigma^2}\right]} \tag{9.1}$$

Unfortunately, Leung and Yan (1998) restrict themselves to the assumption that standard deviation σ is constant across the entire database. However, a minor modification to the technique produces a relaxation in this restriction can allow σ to vary between features. The PDF in equation 9.1 can be used to construct probability surfaces describing the location of individual features in the database, where each feature may have a different standard deviation. Under the assumption that each surface represents the probability of statistically independent events, it is then possible to recombine the surfaces using general probability laws, such as the union (∪) and intersection (∩) in equations 9.2 and 9.3 below (Hugill 1988).

$$P(A \cup B) = P(A) + P(B) - P(A \cap B) \tag{9.2}$$

$$P(A \cap B) = P(A) \cdot P(B) \quad \text{iff A and B are independent} \tag{9.3}$$

Using these basic probability laws, a set of coincident probability surfaces can be recombined on a per-pixel basis, to derive a probability surface describing the locational uncertainty associated with the underlying features. Figure 9.2 illustrates the process diagrammatically, showing two independent probability surfaces derived from two different line features with differing σ values, combined using the intersection operator. The resulting bottom layer in figure 9.2 represents the probability of both features coinciding at that location.

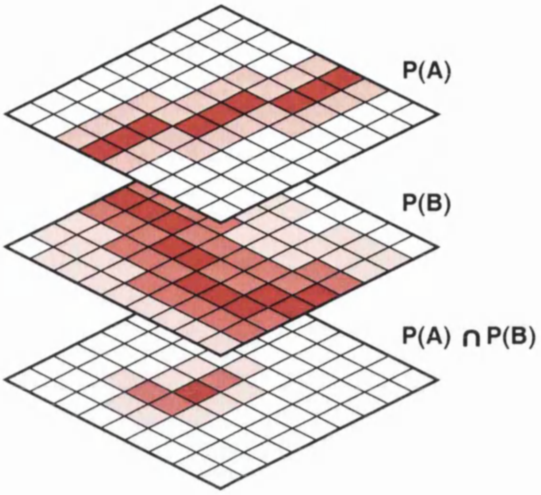
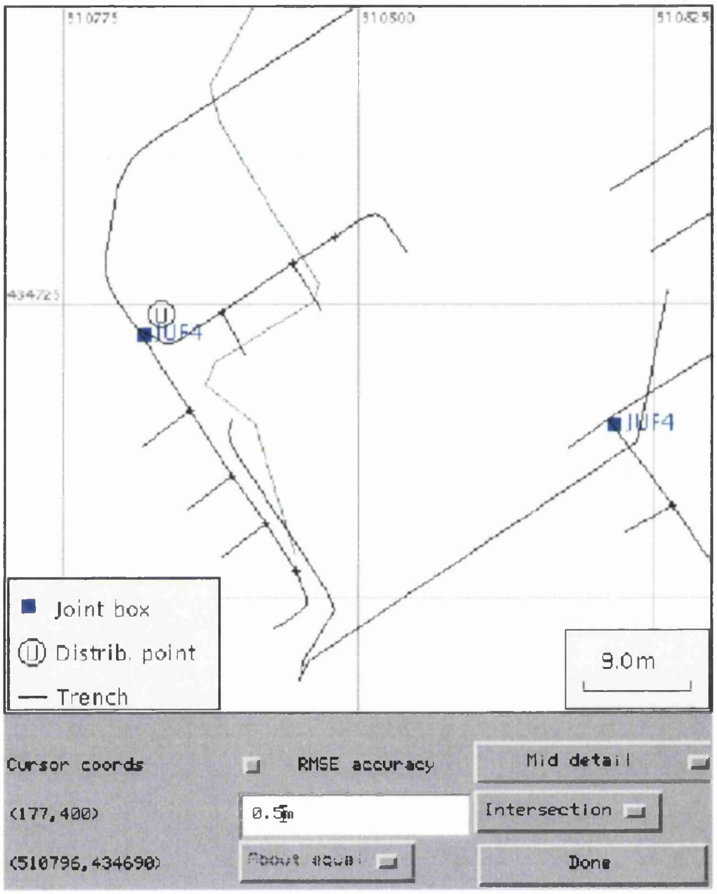


Figure 9.2: Per-pixel probability surface overlay

This error model offers all the characteristics necessary to build the data integration and accuracy visualisation tool. By using spatial information in addition to information about the accuracy of that spatial information stored in the error-sensitive database, the tool can derive compound maps where accuracy information associated with each input feature is propagated to an output map. The error propagation is able to cope with features with different accuracy values, although it can only deal with features that exhibit constant accuracy across their length. A more flexible error model that did not impose this last restriction would undoubtedly be preferable. However, in the case of the KC telecommunications data no information exists about accuracy below the level of spatial primitives. Accuracy within the KC data set may vary spatially, but does not vary within individual spatial features and consequently this restriction imposed by adopting the Leung and Yan error model has no impact upon the example application. Finally, since the Leung and Yan error model uses probability, the error model is conceptually simple and immediately lends itself to intuitive visualisation techniques, where for example the rubber-sheet error model of Kiiveri (1997) in 2.2.1 would be more difficult to visualise.

9.4 Implementation results

Based on the error model and Internet architecture set out above, a data integration and accuracy visualisation tool was implemented. The tool allows users to construct maps from the KC telecommunications database that convey both locational information and the accuracy of that locational information. The integration and visualisation tool interface presupposes that users accessing the web-site have a clear idea of their geographical region of interest. In most cases this is probably a reasonable assumption: the tool is designed to address positional accuracy concerns for specific locational queries. Access to the servlets and applets which comprise the tools is through the familiar medium of a web-browser, such as Netscape. A user accessing the tool's web-site is first presented with a hypertext form that allows the geographical area of interest to be selected using UK National Grid coordinates, although a graphical region selection interface could also be developed as a simple extension to the tool. Once a set of grid coordinates have been entered they are 'remembered' by the tool and the user can return to the page at any time to revise or alter the region of interest.



Cursor coords
(177,400)
(510796,434690)

RMSE accuracy: 0.5m

Figure 9.3: Interactive addition of new features

Having selected a geographical region of interest, pressing the submit button presents the user with an interactive map of the features within the selection region, as in figure 9.3. This map is created dynamically from the database and is georeferenced. The position of the cursor in terms of screen coordinates and national grid coordinates is displayed in the bottom left hand corner of the applet in figure 9.3. At this point the user is optionally able to enter the coordinates of a new feature they wish to integrate with existing features in the selected region. For example, the proposed route of a gas or water pipeline could be input by the supervisor of that work to highlight any areas likely to interfere with existing KC installations. The user is able to interactively mark on the map the position or route of new features to be integrated with the KC telecommunications data, as in figure 9.3 (depicted with a green line in colour illustrations). The accuracy of the location of this proposed feature is an important component of the error analysis. If known, the RMSE of a new feature can be entered, otherwise the user is encouraged to provide an estimate of the relative accuracy of the new feature, also shown in figure 9.3.

Appendix C contains Java code for the applet in figure 9.3, called ImageMapApplet. The telecommunications map of the selected area with which the user interacts is produced by a number of Java servlets, also in appendix C. The ImageMapApplet communicates with these Java servlets to display a map object which uses the common image format, graphics interchange format (GIF). The applet could, in fact, use database objects served by the Java servlets directly. However, the reason for using GIF rather than database objects is to ensure greater data security and to minimise applet complexity and download time, since the GIF format offers data compression. The GIF images are produced by the Java servlet MapGIF in appendix C and could be displayed by any web browser without the need for a Java applet. The Java applet is needed, however, to enable the image to be georeferenced and to let users overlay their own data on top of the telecommunications data.

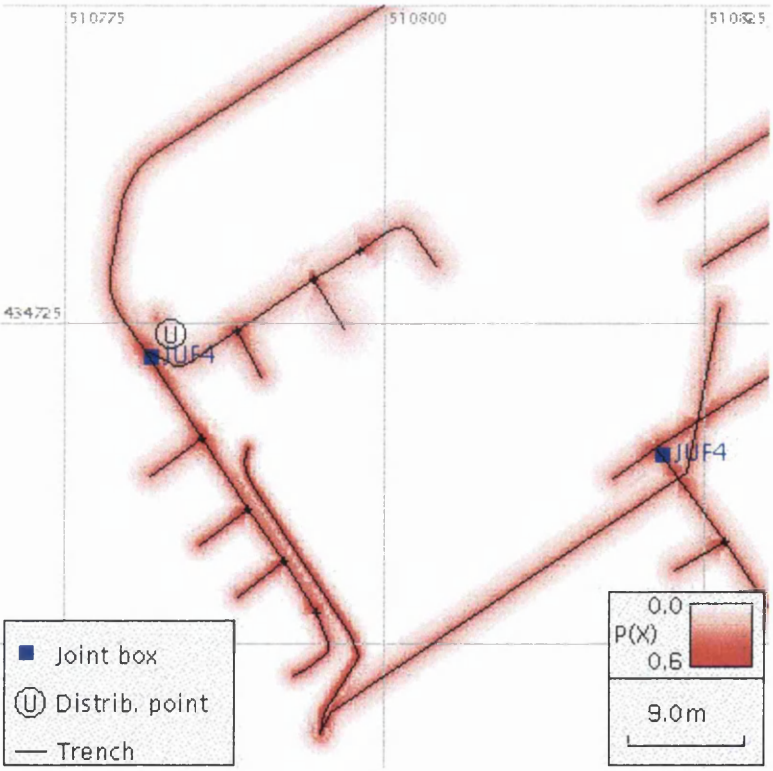


Figure 9.4: Accuracy of location map

Finally, the 'done' button in figure 9.3 takes all the information supplied along with positional accuracy information retrieved from the error-sensitive telecommunications database and produces a locational accuracy map. The locational error map is again produced by a Java servlet, called ErrorGIF and given in appendix C. The servlet uses the deterministic location error model discussed in the previous section to produce a image of the propagated error for each input feature overlaid by the original location map. If no additional features were entered by the user, the servlet uses the logical OR operator to produce an accuracy map which shows the probability of finding an existing telecommunications feature at each point on the map, as in figure 9.4. Otherwise, the servlet uses the AND operator to produce a map which shows the probability of the new feature intersecting existing telecommunications installations, as in figure 9.5.

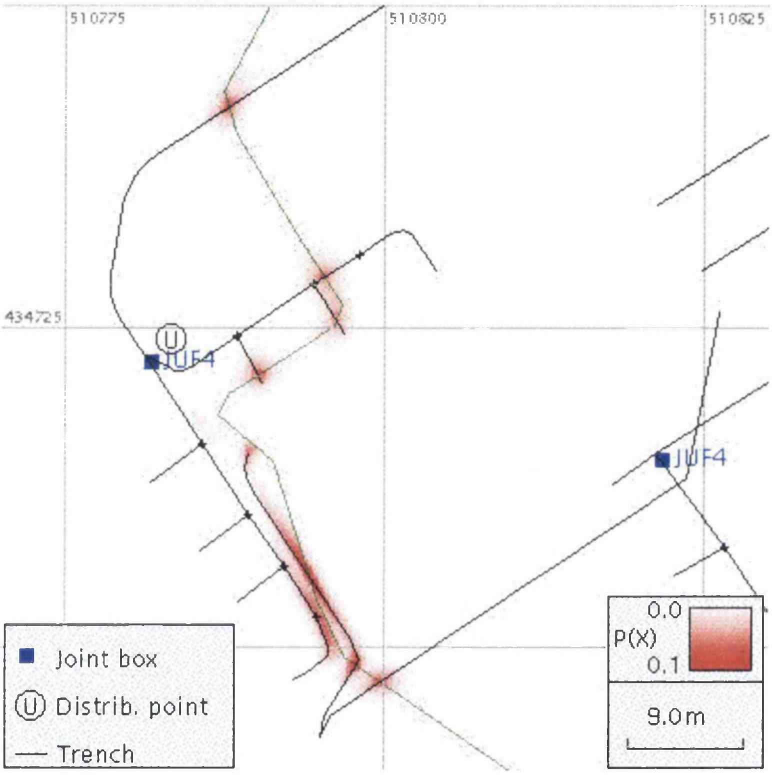


Figure 9.5: Accuracy of intersection map

9.5 Conclusions

Developing clear uses for data quality information is a vital step in gaining widespread acceptance of error handling software and of quality management generally. The KC telecommunications application illustrates that even in an industry dominated by a traditional cartographic approach to spatial information, digital error propagation of accuracy information can be included during a user's access to map-based information. The example is currently only two-dimensional since the KC telecommunications database only contains information about planimetric location of features. Potentially, were information about feature depth available this also could be incorporated into the tool. The core error-sensitive GIS design makes no assumptions about the nature of an error-sensitive object's geometry, although Laser-Scan Gothic software is not currently capable of supporting 3-D geometries. Further, the error model presented by Leung and Yan (1998) does not assume the geometries used exist only in two dimensions. Consequently, extending the tool to deal with 3-D locational uncertainty would be a possibility for future work.

This chapter has indicated how the error-aware GIS architecture can be used to encapsulate an error model alongside error-sensitive geospatial data to distribute the encapsulated data and model over the Internet. While not a core requirement of fitness for use, the opportunity for data producers to provide both data quality information and practical error handling tools for that data may have important financial implications. Data producers who can offer both data quality information and the tools to process that quality information can circumvent the current lack of error handling capabilities in commercial GIS. At the same time, the approach can allow data producing companies to provide quality-enabled 'value added' services that represent a competitive advantage over other data producers. The error-aware GIS is well suited to supporting this approach: the component system architecture can actively encourage the production of lightweight specialised error propagation and processing software integrated with and isolated from the underlying error-sensitive database. Finally, from a research point of view the approach renders long running arguments over the various advantages of different error models largely irrelevant. Research into error propagation and different error models will always be important. However, by using of geospatial data encapsulated within its own error model means it is of far less concern which error model is "better", since potentially software incorporating any error model could be integrated with the error-aware GIS architecture.

Chapter 10

Summary and conclusions

This thesis has attempted to provide a 'road map' for developers of practical error handling software within a GIS context. The development of an OO error-sensitive data model provides core error handling functionality. Intelligent and domain specific error-aware software can be quickly developed, accessing the error-sensitive database via an open GIS interface. The application of this architecture to a telecommunications example indicates a number of areas where the error-aware GIS may be a vital component of an overall quality management strategy. This chapter reviews the contribution of the work presented in the course of this thesis, in addition to the potential contribution of future related research.

10.1 Error handling review

The discussion in §1.4 set out four general error handling research aims, relating to the need to reuse existing GIS functionality, the need for flexibility and for efficiency and the importance of effective user interfaces in error handling systems. This section reviews those aims in the context of this research and explores to what extent each aim has been met.

10.1.1 Error handling and existing GIS

A key aim of this research has been to show how practical error handling tools can be built using existing or at least emerging GIS technology. The core error-sensitive functionality is encoded in an OO data model, which this research indicates can be integrated with any existing OO geospatial data model. High-level error-aware software can take advantage of this core functionality through a distributed systems interface, related to the open GIS model. Both error-sensitive and error-aware GIS build on existing GIS technologies and augment rather than replace basic GIS functionality. The result is a system that can be constructed from software components that may already be familiar to GIS users. Perhaps surprisingly, the only area where current GIS technology does have significant difficulty supporting the error-sensitive and error-aware software is in the use of OODBMS. Experiences during this research with using Laser-Scan Gothic GIS have highlighted a tendency in GIS software vendors to make superficial concessions to OO rather than adopt an underlying OO data model in their DBMS. Other 'OOGIS', such as Smallworld, appear similarly unable to make the jump to fully OODBMS. It is evident that there currently exist no major software vendors who can truthfully claim to offer fully OOGIS.

The lack of truly OOGIS did lead to a number of key compromises during the error-sensitive GIS development. While undesirable, the current situation in GIS software can be characterised as a transitional phase. Software companies are understandably uncomfortable deserting the relational tradition of GIS altogether for a number of reasons. First, there exists a wealth of theory and technology to support the relational model that is only now beginning to be matched by OO. Further, development periods for GIS have in the past been very long. The results of decisions taken 15 years ago are evident in Laser-Scan Gothic software; with hindsight, some of those decisions would undoubtedly have been different. The efficiency gains accrued through increasing use of OOP is in itself likely to dramatically reduce software development times for the next generation of OO software; again Gothic was built using the procedural programming language C, rather than an OOPL.

The error-sensitive GIS architecture presented here does not allow reuse of existing relational technology. There are very real financial concerns for those still using a relational systems, and software companies will need to support legacy relational GIS for many years to come. The tensions between OO and relational GIS have to an extent retarded the development of OOGIS. However, the semantic modelling advantages of OO over other software paradigms, set out in chapter 3, present a compelling reason to believe the reticence of GIS software companies in developing fully OOGIS is only a temporary hiatus. In addition to the trend toward OOGIS identified in chapter 3, throughout this thesis a number of other areas have been linked with a convergence on OO concepts and technology. A number of chapters have mentioned the significance of the AI and KR roots of OO. Chapter 6 pointed to the metaphor of the contract in both OO programming and client/server systems while chapter 9 highlighted the analogy between OO and the encapsulation of data and functionality within error-aware applications. Whether this convergence on OO is significant or serendipitous, it seems likely that the required growth of OOGIS is inevitable. Therefore, it seems reasonable to claim that an error handling system based on an OOGIS does indeed extend rather than replace existing and emerging GIS technology.

10.1.2 Error handling must be flexible

Building flexible error handling software systems involves a calculated compromise between flexibility and practicality. On the one hand error handling systems that are highly specific are also likely to be highly inflexible. At the other extreme, systems that are too general can achieve flexibility at the cost of practical error handling concerns. For example, a highly flexible approach is taken by Wesseling and Heuvelink (1993) in the development of the ADAM error-propagation tool. ADAM offers error-propagation functionality that Heuvelink (1998) convincingly argues would be entirely compatible with any GIS given the absolute minimum of functionality. However, ADAM gives no indication of how information used in error propagation should be retrieved, structured or stored and offers no mechanisms for storing the information produced by error propagation. While this was undoubtedly intentional, it does illustrate the wider problem that highly flexible approaches tend to fall short of practical solutions. At the other extreme, §2.1.4 highlighted a number of practical GIS implementations with error handling capabilities that were based solely on storing the five SDTS quality elements. Such GIS are certainly valuable practical systems, but only as far as the discussion of data quality does not range outside the rather restrictive boundaries of SDTS.

Flexibility has been a key factor behind the error-sensitive GIS development at a number of points during this research. The initial decision to restrict the discussion of error handling to OO systems alone is itself a compromise aimed at maximising software flexibility at the same time as providing a practical software solution. The resultant OO error-sensitive GIS is not flexible enough to be implemented within any DBMS, but is flexible enough to be implemented within any OODBMS. By using an OODBMS error-sensitive behaviour can be encapsulated within objects, ensuring that any object can be interrogated regarding its own quality. Further, there is considerable flexibility built into the types of quality objects that can be associated with other database objects. In the past, reliance on existing data quality standards can be cited as a limiting factor in the development of error handling in GIS. The error-sensitive GIS severs this link between error handling and data quality, allowing existing data quality standards, user defined quality and meta-quality schema to coexist in the same database. In this way, error-sensitive database objects carry around not simply information about their state, but meta-information about the state of the model they comprise. At the same time as being flexible, chapter 5 illustrates how the software can actually be implemented and employed in a telecommunications application. The approach used can claim to offer the best compromise between practical and flexible error-sensitive software.

Finally, the component architecture explored in chapter 6 aims to safeguard the flexibility of the error-sensitive GIS by providing a clear separation between data model and data usage. The component architecture allows a data producer to make information available to users without the need to try and pre-empt how quality information may actually be used. Specialised lightweight error-aware applications can be developed quickly and at low cost on top of the core quality functionality encapsulated in error-sensitive objects. The example application of the error-aware GIS to a telecommunications database clearly illustrates how such highly specialised and intelligent software can take advantage of error-sensitive functionality without the need to make any modifications or additions to the underlying error-sensitive database.

10.1.3 Error handling must be efficient

Object orientation is an efficient development paradigm as the same consistent concepts are reused throughout the analysis, design and implementation process (see §3.2.3). The conceptual efficiency of the error-sensitive and error-aware GIS is largely due to the use of OO throughout the development process. Fundamental to OO is the management of complexity. The combination of classification, encapsulation and inheritance, introduced in chapter 3, enable core error-sensitive behaviour to be transmitted throughout an entire OO geospatial database, using a data model that can be exhaustively and explicitly expressed using a handful of ς-calculus terms. At the same time, the component architecture allows objects in the error-sensitive database to be accessed directly by error-aware applications. At every level of the error-sensitive and error-aware GIS the same OO concepts are reused, concepts that have resulted from an analysis that aims to correspond directly with the real world. The same cannot be said of relational GIS, for example, where different conceptual approaches are used throughout the analysis, design and implementation process, ranging from ordered lists, tables and keys, to layers and coverages, to real world observations.

Computational efficiency can also be built into the error-sensitive and error-aware GIS. The discussion of the error-sensitive GIS in chapter 4 introduced an extension to the error-sensitive GIS able to pack data quality information efficiently around geospatial objects in the database. The efficient quality storage model was based on the observation that geospatial concepts tend to be organised hierarchically, where complex geographical objects are often aggregated from less complex ones. The efficient quality storage model is able to infer quality for component objects from the quality of more complex objects. It is worth noting that the idea could be significantly extended, discussed later in §10.2. Finally, the component architecture also contributes to the computational efficiency of the system, as it spreads the load of computationally intensive error handling operations, such as error propagation, across an arbitrary number of networked computers. Component error-aware software running, say, a Monte-Carlo simulation and the error-sensitive database can and usually will be operating on physically separate computers.

10.1.4 Error handling and user interfaces

The need for exploratory, user friendly interfaces for the error-sensitive database is addressed by the error-aware GIS. The provision of mechanisms for actually using data quality information once it has been collated and stored in a GIS is arguably one of the most neglected area of research into spatial error. Given that the error-sensitive GIS augments rather than replaces core GIS functionality (discussed in 10.1.1), providing help with using and understanding data quality information assumes even greater significance if error handling capabilities are ever to be used. To address this need, error-aware GIS tools access core error-sensitive functionality though the component architecture. This allows error-aware tools to be specialised enough to offer user-oriented interfaces. This research has indicated that the use of artificial intelligence technologies can help such interfaces offer assistance with complex data quality issues, with which most users have little or no familiarity. At the same time, familiar user interfaces, such as hypertext-based web pages, can minimise user inertia and ease users facing a new error-aware tool for the first time through the learning curve.

The results of developing the three error-aware tools in chapters 7–9 were generally positive. The tools seem able to address the particular needs of the example telecommunications application, ultimately assisting users not simply in using the data, but considering the fitness of that data for a particular use. However, there exists a caveat. All the error-aware software developed as part of this research are more a proof of concept than an attempt to produce viable software. The software has not been tested 'in anger', and the KC telecommunications data capture project has now been completed without having used any error handling capabilities. Whether the results of the KC data capture process would have been significantly different, KC's current use of their data would have been more efficient, or the future cost of data maintenance and integration for KC would have been any lower if the error handling software developed in this research had been available for use by KC remains something of an open question. It seems reasonable to suggest that adopting the architecture and approaches proposed in this thesis would be beneficial both to research and commercial GIS applications. However, before error handling capabilities in GIS are likely to become commonplace, further work still exists in a number of areas, many of which are highlighted in the next section.

10.2 Further work

A range of further work is suggested by this research. This section highlights the important error handling research questions that follow from this thesis and to what extent these research strands are already being tackled. In the light of these as yet unanswered research questions the section concludes with a look at the feasibility of commercial error handling systems.

10.2.1 Data quality information

While this research has looked at *how* to store, manage and use quality information, the question of *what* that quality information should be stored remains largely unanswered. The OOA of data quality provides a template for quality information defining the basic modes of interaction between quality and other information. Further, this research has added weight to the growing dissatisfaction with existing spatial data quality standards, which are largely inappropriate for use in an IS context. In their current form, many quality elements proposed by data quality standards, in particular elements like lineage, source, usage, abstraction modifier, are largely useless for computational purposes, whilst others, for example completeness, reliability, semantic accuracy, are so ill-defined as to negate any intrinsic utility they may possess. Despite this, the basic 'Famous Five' elements of spatial data quality continue to dominate both applications in this research and most other research into spatial data quality. Yet, more than a decade after the inception of the NCDCDS, there exists no convincing research forwarding detailed and supported arguments for *why* standard quality elements such as those set out in the 'Famous Five' are appropriate indicators of spatial data quality. There is a clear and pressing need for a reappraisal of exactly what constitutes spatial data quality.

10.2.2 Extending the error-sensitive GIS

The core concepts put forward during the construction of the error-sensitive GIS could benefit from extension and further work. The efficient quality storage model, discussed in §4.2.4.1 and §10.1.3, depends upon the assumption of hierarchical geospatial object aggregation, illustrated in figure 10.1a, based on figure 4.3. However, such an assumption may not always hold. For example, a single geometry object may be a component of several more complex geospatial objects in some cases, as illustrated by figure 10.1b. Further, the efficient storage model can only support inference in one direction on the aggregation hierarchy. It is conceivable that in some situations, such as that shown in figure 10.1c the quality of a complex geospatial object might be some function of the quality of its component objects. Potentially, there are a many different ways in which an extended version of the efficient storage model might offer both low data quality volumes and more sophisticated inference mechanisms. As long as inferred data quality information can be distinguished from primary quality information, probably by attaching meta-quality information to all retrieved data quality objects, it would be reasonable to embed relatively complex inference or analysis mechanisms within the geospatial data. The idea of a 'measurement-based GIS' proposed by Goodchild (1999), where original geodetic observations are retained and derived quality information is created 'on-the-fly' using adjustment, is not entirely new but could certainly be implemented within an error-sensitive environment in much the same way as the efficient quality storage model.

10.2.3 Relational error-sensitive GIS

The work reported in this thesis is based on the assertion that OO represents a better model for geospatial information than any other currently available. Nevertheless, it is certainly the case that there will continue to be a rôle for relational and extended-relational GIS technology for many years to come. Indeed, new research into error handling in relational GIS should continue apace. Qiu and Hunter (1999) are using a hierarchical approach to data quality information, related to the OO approach used here, but implemented within a relational environment. Unfortunately, the shortcomings of the relational model are likely to be disadvantageous to such research, and indeed the work presented by Qiu and Hunter (1999) is subject to the same criticisms about reliance on a particular data quality standard common to most other research into relational error

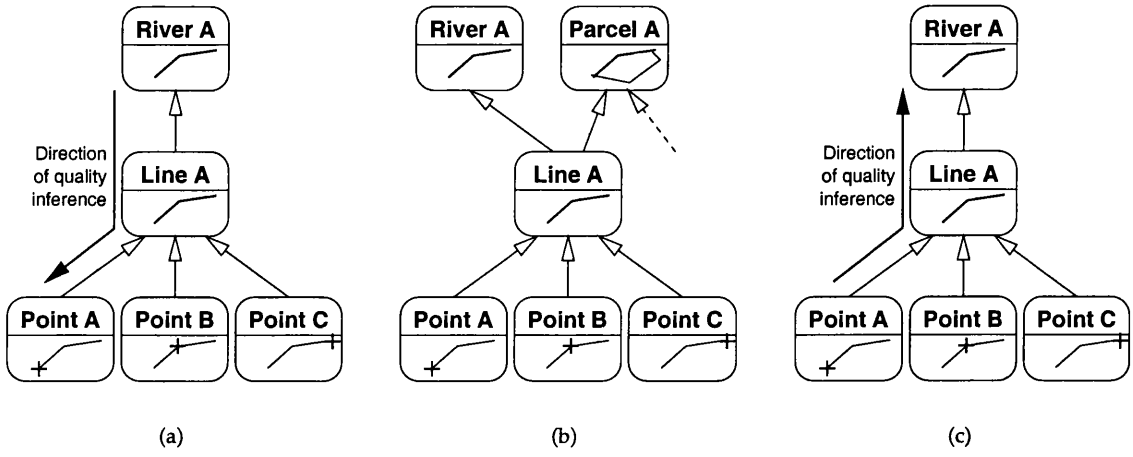


Figure 10.1: Extended efficient storage models

handling GIS. However, it is not the intention to suggest that valid relational GIS research is not possible, and both for practical reasons and in the pursuit of a rounded research agenda, such research is indeed essential.

10.2.4 OOAD and the ς-calculus

The construction in this thesis of a formal OO model of spatial data quality using the ς-calculus is a new departure for GI science. While the ς-calculus application in this research have been largely illustrative rather than fundamental, the potential exists to revolutionise OOAD both within GI science and information science more generally. Even with the greatest care, informal OO software engineering can be a volatile, unpredictable and subjective exercise. The ς-calculus offers a sound theoretical basis for practical OO software analysis and design, bringing OOP and OO database design in line with common practice in functional programming and relational database design. The separation of formal object-theory from OO software and database development is a crucial step toward addressing the underlying contradictions that can occur without formal methods, such as those explored in §4.2.5. Considerable further work could follow from this initial use of ς-calculus in integrating the OO software development process with ς-calculus and in developing links between spatial theory and ς-calculus.

10.2.5 Further applications

The example telecommunications application explored in this thesis goes some way toward redressing the lack of research into the use of data quality information. The error-aware GIS tools developed in chapters 7–9 take advantage of a variety of techniques aimed at offering high-level error handling functionality able to break down the barriers between data quality and the practical use and understanding of error-prone spatial information. In spite of the contributions in this thesis and some ongoing research efforts, for example Hunter (1999) and Reinke and Hunter

(1999), the understanding and use of spatial data quality information remains greatly under-researched. In particular, there exists a clear idea of neither how the user's perspective nor application area may affect the necessary quality communication and visualisation tools. The KC application was useful because it provided a series of clear example problems encountered during legacy data capture; problems common to the telecommunications industry at the current time. However, the application is by no means representative of quality issues generally. Natural resources, for example, exhibit very different quality aspects when compared with the utility industry. As has already been touched upon, natural resources managers may be interested in different types of quality issues, related to vagueness and indeterminacy in information. In contrast, a utility network manager may be primarily interested in accuracy, imprecision and topology. There exists considerable scope for further work to address issues surrounding the importance of application area and user perspective in understanding spatial data quality.

10.2.6 Feasibility of error handling in commercial GIS

Not all the further work suggested in this chapter needs to be undertaken in a research context. The results of this research strongly suggest that error handling capabilities can be incorporated within GIS using current technology. Furthermore, it seems likely that current trends in software development are likely to make the development of error handling within GIS even less problematic in the future. Unfortunately, such capabilities do not yet command a high priority with GIS software suppliers or commercial GIS users, and there is no indication of that this undesirable situation is subject to imminent change. Some of the factors contributing to this unsatisfactory state of affairs have been highlighted in this chapter. The lack of fully OOGIS, a dependence on data quality standards, the need for flexible yet practical error handling GIS software have all been touched on. Further, there is often general unease amongst software vendors, data producers and data users surrounding the issue of error. Error is often perceived as value-laden: data quality has been synonymous with poor quality. Hopefully this research goes some way to addressing each of these difficulties. Anecdotal evidence, at least, suggests that the fear that error handling software may become associated with low-quality data has in the past been enough for software producers to withhold GIS software with error handling capabilities. The results of this research suggest that practical error handling capabilities can be incorporated within GIS software and software vendors, data producers and data users also need to take some responsibility for ensuring future GIS software embodies some of these features.

10.3 Closing remarks

Error is intrinsic to geospatial information. Currently commercial and research GIS projects often ignore this error or at best treat it as a separate phenomena that can be appended to existing geospatial information. This research has tried to show how GIS that are able to model this intrinsic error as an integral component of the geospatial data model can be developed. Further, the research has argued that the availability of high level tools able to capture, store, and apply information about error is an essential addition to this core functionality if it is to be used. The results of this research indicate that such error handling capabilities can be achieved using largely conventional technology. This research suggests that the ultimate goal of error handling GIS able to provide both a context for GI and the tools with which to make use of that contextual information in the assessment of fitness for use is, in technological terms, entirely attainable.

Inevitably, much work still remains in determining how to represent, propagate and communicate this quality information. However, the error-sensitive and error-aware GIS architecture proposed in this thesis provides a flexible software framework that should be capable of supporting further research as it appears. The attempt has been made in this thesis to take this first step in addressing the technological and practical aspects of building error into GIS and the results can claim to offer a prototype for wider research and commercial error handling GIS architectures.

Bibliography

Aalders, H. (1996). Quality metrics for GIS. In M. Kraak and M. Molenaar (Eds.), *Advances in GIS research II; Proceedings Seventh International Symposium on Spatial Data Handling*, Volume 1, pp. 5B1–10.

Aalders, H. (1999). The registration of quality in a GIS. In W. Shi, M. Goodchild, and P. Fisher (Eds.), *Proceedings of the International Symposium on Spatial Data Quality*, pp. 23–32.

Abadi, M. and L. Cardelli (1995a). An imperative object calculus. *Theory and Practice of Object Systems 1*(3), 151–166.

Abadi, M. and L. Cardelli (1995b). A theory of primitive objects: second-order systems. *Science of Computer Programming 25*(2-3), 81–116.

Abadi, M. and L. Cardelli (1996a). *A theory of objects*. New York: Springer-Verlag.

Abadi, M. and L. Cardelli (1996b). A theory of primitive objects: untyped and first-order systems. *Information and Computation 125*(2), 78–102.

Abbott, R. (1987). Knowledge abstraction. *Communications of the ACM 30*(8), 664–671.

Abler, R. (1987). The National Science Foundation National Center for Geographic Information and Analysis. *International Journal of Geographical Information Systems 1*(4), 303–326.

Adler, R. (1995). Distributed computation models for client/server computing. *IEEE Computer 28*(4), 14–22.

Agosti, M. (1996). *An overview of hypertext*, Chapter 2, pp. 27–47. Boston: Kluwer Academic Publishing.

Agumya, A. and G. Hunter (1997). Determining fitness for use of geographic information. *ITC Journal* (2), 109–113.

Altman, D. (1994). Fuzzy set theoretic approaches for handling imprecision in spatial analysis. *International Journal of Geographical Information Systems 8*(3), 271–289.

Andrienko, G. and N. Andrienko (1999). Interactive maps for visual data exploration. *International Journal of Geographical Information Science 13*(4), 355–374.

Arnold, K. and J. Gosling (1996). *The Java programming language*. Massachusetts: Addison-Wesley.

Aspinall, R. (1992). An inductive modelling procedure based on Bayes theorem for analysis of pattern in spatial data. *International Journal of Geographical Information Systems 6*(2), 105–121.

Aspinall, R. (1996). Measurement of area in GIS: a rapid method for assessing the accuracy of area measurement. In *Proceedings of the GIS Research UK 1996 conference*, pp. 135–142.

Atkinson, M., P. Bailey, K. Chisholm, P. Cockshott, and R. Morrison (1983). An approach to persistent programming. *Computer Journal 26*(4), 360–365.

Atkinson, M., L. Daynes, M. Jordan, T. Printenzis, and S. Spence (1996). An orthogonally persistent Java. *ACM Sigmod Record 25*(4).

Atkinson, M. and R. Morrison (1985). Procedures as persistent data objects. *ACM Transactions on Programming Languages and Systems 7*(4), 539–559.

Bannister, A., S. Raymond, and R. Baker (1992). *Surveying* (Sixth ed.)., pp. 239–274. Essex: Longman.

Barendregt, H. (1984). *The lambda calculus: its syntax and semantics* (Revised ed.). Amsterdam: Elsevier Science.

Bastin, L., J. Wood, and P. Fisher (1999). Visualisation of fuzzy spatial information in spatial decision making. In K. Lowell and A. Jaton (Eds.), *Spatial accuracy assessment: land information uncertainty in natural resources*, Chapter 18, pp. 147–150. Michigan: Ann Arbor.

Batty, P. (1992). GIS databases — a distributed future. *Mapping Awareness 6*(5), 34–37.

Becker, L., A. Voigtmann, and K. Hinrichs (1996). Developing applications with the object-oriented GIS-kernel Goodac. In M. Kraak and M. Molenaar (Eds.), *Advances in GIS research II; Proceedings Seventh International Symposium on Spatial Data Handling*, Volume 1, pp. 5A1–18.

Belkhouche, B. and A. Gamiño (1998). Object-oriented analysis through a knowledge based system. *Journal of Object-Oriented Programming 11*(7), 52–59.

Bennet, D. and M. Armstrong (1996). An inductive based approach to terrain feature extraction. *Cartography and GIS 23*(1), 3–19.

Bhaskar, K. (1983). How object-oriented is your system? *SIGPLAN Notices 18*(10), 8–11.

Bielawski, L. and R. Lewand (1991). *Intelligent systems design: integrating expert systems, hypermedia, and database technologies.* New York: Wiley.

Blakemore, M. (1984). Generalisation and error in spatial databases. *Cartographica 21*(2/3), 131–139.

Blaschek, G. and J. Frölich (1998). Recursion in object-oriented programming. *Journal of Object-Oriented Programming 11*(7), 29–35.

Booch, G. (1994). *Object oriented analysis and design with applications* (Second ed.). California: Benjamin–Cummings.

Booch, G., I. Jacobson, and J. Rumbaugh (1999). *The Unified Modeling Lanuguage user guide.* Massachusetts: Addison-Wesley.

Bowen, J. (1996). *Formal specification and documentation using Z: a case study approach.* London: International Thomson Computer Press.

Bowen, J. and M. Hinchey (1995). Ten commandments of formal methods. *IEEE Computer 28*(4), 56–63.

Brodie, M. (1984). On the development of data models. In M. Brodie, J. Mylopoulos, and J. Schmidt (Eds.), *On conceptual modelling*, Chapter 2, pp. 19–47. New York: Springer-Verlag.

Brunsdon, C. and S. Openshaw (1993). Simulating the effects of error in GIS. In P. Mather (Ed.), *Geographical information handling: research and applications.* New York: Wiley.

Budd, T. (1991). *An introduction to object-oriented programming.* Massachusetts: Addison-Wesley.

Burrough, P. (1986). *Principles of geographical information systems for land resources assessment.* Oxford: Oxford University Press.

Burrough, P. and R. McDonnell (1998). *Principles of geographical information systems* (Second ed.). Oxford: Clarendon.

Campari, I. and A. Frank (1993). Cultural differences in GIS: a basic approach. In *Proceedings Fourth European Conference on Geographical Information Systems*, Volume 1, pp. 10–16.

Cardelli, L. (1984). A semantics of multiple inheritance. In G. Goos and J. Hartmanis (Eds.), *Semantics of data types*, Number 173 in Lecture notes in computer science, pp. 51–67. New York: Springer-Verlag.

Cardelli, L. and P. Wegner (1985). On understanding types, data abstraction and polymorphism. *ACM Computing Surveys 17*(4), 471–522.

CEN/TC287 (1996). Draft european standard: geographic information — quality. Technical Report prEN 287008, European Committee for Standardisation.

Cho, G. (1998). *Geographic information systems and the law.* New York: Wiley.

Chrisman, N. (1982). A theory of cartographic error and its measurement in digital data bases. In *Proceedings Auto-Carto 5*, pp. 159–168.

Chrisman, N. (1983). The role of quality information in the long term functioning of a geographic information system. *Cartographica 21*(2/3), 79–87.

Chrisman, N. (1987). The accuracy of map overlays: a reassessment. *Landscape and Urban Planning 14*(5), 427–439.

Chrisman, N. (1989). Modeling error in overlaid categorical maps. In M. Goodchild and S. Gopal (Eds.), *Accuracy of spatial databases*, Chapter 2, pp. 21–34. London: Taylor and Francis.

Chrisman, N. (1991). The error component in spatial data. In D. Maguire, M. Goodchild, and D. Rhind (Eds.), *Geographical information systems*, Volume 1, Chapter 12, pp. 165–174. Essex: Longman.

Chrisman, N. (1997). *Exploring geographic information systems.* New York: Wiley.

Civco, D. (1989). Knowledge based land use and land cover mapping. In *Remote Sensing*, Volume 3 of *ASPRS/ASCM technical papers annual convention*, pp. 276–291.

Coad, P. (1992). Object-oriented pattens. *Communications of the ACM 35*(9), 152–159.

Coad, P. and E. Yourdon (1991a). *Object-oriented analysis.* New Jersey: Yourdon Press.

Coad, P. and E. Yourdon (1991b). *Object-oriented design.* New Jersey: Yourdon Press.

Cohen, A. (1984). Data abstraction, data encapsulation and object-oriented programming. *SIGPLAN Notices 19*(1), 31–35.

Cohn, A. and N. Gotts (1996). The 'egg-yolk' representation of regions with indeterminate boundaries. In Burrough, P.A. and Frank, A.U. (Eds.), *Geographic objects with indeterminate boundaries*, GIS Data 2. London: Taylor and Francis.

Coleman, D. (1999). Geographic information systems in a networked environment. In P. Longley, M. Goodchild, D. Maguire, and D. Rhind (Eds.), *Geographical information systems* (Second ed.), Volume 1, Chapter 22, pp. 317–329. Essex: Longman.

Collins (1997). New english dictionary. Harper Collins.

Date, C. (1990). *An introduction to database systems* (Fifth ed.), Volume 1. Massachusetts: Addison-Wesley.

Date, C. (1995). *Relational database writings 1991–1994.* Massachusetts: Addison-Wesley.

David, B., M. van den Herrewegen, and F. Salgé (1996). Conceptual models for geometry and quality of geographic information. In P. Burrough and A. Frank (Eds.), *Geographic objects with indeterminate boundaries*, GIS Data 2. London: Taylor and Francis.

Davis, C. J. and K. Borges (1994). Object oriented GIS in practice. *Urban and Regional Information Association*, 786–795.

Davis, T. and C. Keller (1997). Modelling uncertainty in natural resource analysis using fuzzy sets and Monte Carlo simulation: slope stability prediction. *International Journal of Geographical Information Science 11*(5), 409–434.

de Champeaux, D. and P. Faure (1992). A comparative study of object-oriented analysis methods. *Journal of Object-Oriented Programming 5*(1), 21–33.

de Jong, W. and F. van der Wel (1990). Embedded artificial intelligence and spatial data handling: some research and prototyping experiences. In *Proceedings Fourth International Symposium on Spatial Data Handling*, pp. 723–731.

Denning, P. (1986). Towards a science of expert systems. *IEEE Expert 1*(2), 80–83.

Drummond, J. (1995). Positional accuracy. In S. Guptill and J. Morrison (Eds.), *Elements of spatial data quality*, Chapter 3, pp. 31–58. Oxford: Elsevier Science.

Drummond, J. (1996). GIS: the quality factor. *Surveying World 4*(6), 26–27.

Dutton, G. (1996). Improving locational specificity of map data — a multi resolution, metadata-driven approach and notation. *International Journal of Geographical Information Systems 10*(3), 253–268.

Egenhofer, M. and A. Frank (1989). Object-oriented modeling: inheritance and propagation. In *Proceedings Auto-Carto 9*, pp. 588–598.

Egenhofer, M. and A. Frank (1990). LOBSTER: combining AI and database techniques for GIS. *Photogrammetric Engineering and Remote Sensing 56*(6), 919–926.

Egenhofer, M. J. and A. U. Frank (1992). Object-oriented modeling for GIS. *URISA Journal 4*(2), 3–19.

Ehlers, M., D. Greenlee, T. Smith, and J. Star (1991). Integration of remote sensing and GIS: data and data access. *Photogrammetric Engineering and Remote Sensing 57*(6), 669–675.

Ehlers, M. and W. Shi (1996). Error modelling for integrated GIS. *Cartographica 33*(1), 11–21.

Elmes, G., G. Cai, and M. Twery (1994). Estimating spatial data error by inference on a causal network. In T. Waugh and R. Healey (Eds.), *Advances in GIS research; Proceedings Sixth International Symposium on Spatial Data Handling*, Volume 1, London, pp. 254–277. Taylor and Francis.

Emmi, P. and C. Horton (1995). A Monte Carlo simulation of error propagation in a GIS based assessment of seismic risk. *International Journal of Geographical Information Systems 9*(4), 447–461.

Epstien, E., G. Hunter, and A. Agumya (1998). Liability insurance and the use of geographical information. *International Journal of Geographical Information Science 12*(3), 203–214.

Epstien, E. and H. Roitman (1990). Liability for information. In D. Peuquet and D. Marble (Eds.), *Introductory readings in Geographic Information Systems*, Chapter 26, pp. 364–371. London: Taylor and Francis.

Evans, C., D. Lacey, D. Harvey, D. Gibbons, and A. Krasun (1995). *Client/server: a handbook of modern computer design*. New York: Prentice Hall.

Fiadeiro, J. and T. Maibaum (1991). Describing, structuring and implementing objects. In G. Goos and J. Hartmanis (Eds.), *Foundations of Object-Oriented Languages*, Number 489 in Lecture notes in computer science, pp. 274–310. New York: Springer-Verlag.

Fisher, P. (1989). Knowledge based approaches to determining and correcting areas of unreliability in geographic databases. In M. Goodchild and S. Gopal (Eds.), *Accuracy of spatial databases*, Chapter 4, pp. 45–54. London: Taylor and Francis.

Fisher, P. and W. Mackaness (1993). Are cartographic expert systems possible? In *Proceedings Fourth European Conference on Geographical Information Systems*, Volume 1, pp. 530–534.

Fisher, P., W. Mackaness, G. Peacegood, and G. Wilkinson (1988). Artificial intelligence and expert systems in geodata processing. *Progress in physical geography 12*(3), 371–388.

Flanagan, D. (1996). *Java in a nutshell*. Sebastopol: O'Reilly & Associates.

Flewelling, D., M. Egenhofer, and A. Frank (1992). Constructing geological cross sections with a chronology of geological events. In *Fifth International Symposium on Spatial Data Handling*, Volume 2, pp. 544–553.

Flowerdew, R. (1991). Spatial data integration. In D. Maguire, M. Goodchild, and D. Rhind (Eds.), *Geographical information systems* (First ed.), Volume 1, Chapter 24, pp. 375–387. Essex: Longman.

Forier, F. and F. Canters (1996, May). A user friendly tool for error modelling and error propagation in a GIS environment. In H. Mowrer, L. Raymond, and R. Hamre (Eds.), *Spatial accuracy assessment in natural resources and environmental sciences: Second International Symposium*, pp. 225–234.

Forrest, D. (1993). Expert systems and cartographic design. *Cartographic Journal 30*(2), 143–148.

Frank, A. and W. Kuhn (1995). Advances in spatial databases. In M. Egenhofer and J. Herring (Eds.), *Specifying open GIS with functional languages*, Number 951 in Lecture notes in computer science, pp. 184–195. New York: Springer-Verlag.

Friedman, A. and D. Cornford (1989). *Computer systems development: history, organisation and implementation*. New York: Wiley.

Friedman-Hill, E. (1999). JESS home page. http://herzberg.ca.sandia.gov/jess. Last modified 27 November 1999, last accessed 21 December 1999.

Frost, D., M. Gillenson, and M. Kilpatrick (1994). The OODB as a self contained expert system. *Journal of Object-Oriented Programming* 6(9), 31–36.

Geographic Data BC (1996). Spatial Archive and Interchange Format release 3.2 formal definition. http://www.env.gov.bc.ca/gdbc/saif32/. Last modified 14 September 1999, last accessed 21 December 1999.

Godwin, L. (1999, March). ISO 15046-13: a framework for quality information. Technical paper, 1999 ACSM Annual Convention.

Goldsack, S. (1996). *Formal methods in object technology*, Chapter 1, pp. 3–16. London: Springer-Verlag.

Goodchild, M. (1989). Modeling errors in objects and fields. In M. Goodchild and S. Gopal (Eds.), *Accuracy of spatial databases*, Chapter 10, pp. 107–114. London: Taylor and Francis.

Goodchild, M. (1995). Sharing imperfect data. In H. Onsrud and G. Rushton (Eds.), *Sharing geographic information*, pp. 413–425. Jersey: Rutgers.

Goodchild, M. (1998). Different data sources and diverse data structures: metadata and other solutions. In P. Longley, S. Brooks, R. McDonnel, and B. MacMillan (Eds.), *Geocomputation: a primer*, pp. 61–73. New York: Wiley.

Goodchild, M. (1999). Measurement-based GIS. In W. Shi, M. Goodchild, and P. Fisher (Eds.), *Proceedings of the International Symposium on Spatial Data Quality*, pp. 1–9.

Goodchild, M. and O. Dubuc (1986). A model of error for chloropleth maps with applications to geographic information systems. In *Proceedings Auto-Carto 8*, pp. 165–174.

Goodchild, M. and S. Gopal (1989). *Accuracy of spatial databases*, pp. xi–xv. London: Taylor and Francis.

Goodchild, M. and R. Jeansoulin (1998). Editorial. *GeoInformatica* 2(3), 211–214.

Goodchild, M. and P. Longley (1999). The future of GIS and spatial analysis. In P. Longley, M. Goodchild, D. Maguire, and D. Rhind (Eds.), *Geographical information systems* (Second ed.), Volume 1, Chapter 40, pp. 567–580. Essex: Longman.

Goodchild, M. and J. Proctor (1997). Scale in a digital geographic world. *Geographical and environmental modelling* 1(1), 5–23.

Goodchild, M., A. Shortridge, and P. Fohl (1999). Encapsulating simulation models with geospatial data sets. In K. Lowell and A. Jaton (Eds.), *Spatial accuracy assessment: land information uncertainty in natural resources*, Chapter 14, pp. 123–130. Michigan: Ann Arbor.

Goodchild, M., Sun Guoqing, and Yang Shiren (1992). Development and test of an error model for categorical data. *International Journal of Geographical Information Systems* 6(2), 87–104.

Gordon, A. and G. Rees (1996). Bisimilarity for a first-order calculus of objects with subtyping. In *Proceedings 23rd Annual ACM Symposium on Principles of Programming Languages*, pp. 386–395.

Green, P. (1995). Overview: Monte Carlo methods. In D. Tittering (Ed.), *Complex Stochastic Systems and Engineering*, Number 54 in IMA Conference series, pp. 183–190. Oxford: Oxford University Press.

GTE Interworking (1999). OpenMap home page. http://openmap.bbn.com/. Last modified 4 November 1999, last accessed 21 December 1999.

Guptill, S. (1989). Inclusion of accuracy data in a feature based object oriented data model. In M. Goodchild and S. Gopal (Eds.), *Accuracy of spatial databases*, Chapter 8, pp. 91–98. London: Taylor and Francis.

Guptill, S. (1992). The Sequoia 2000 approach to managing large spatial object databases. In *Proceedings Fifth International Symposium on Spatial Data Handling*, Volume 2, pp. 642–651.

Guptill, S. and R. Fegeas (1988). Feature based spatial data models — The choice for global databases in the 1990's. In H. Mounsey and R. Tomlinson (Eds.), *Building databases for global science*, pp. 279–295. London: Taylor and Francis.

Guptill, S. and J. Morrison (Eds.) (1995). *Elements of spatial data quality.* Oxford: Elsevier Science.

Hankin, C. (1994). *Lambda calculi: a guide for computer scientists.* Oxford: Clarendon Press.

Harold, E. (1997). *Java network programming.* Sebastopol: O'Reilly.

Harris-Jones, C. (1995). *Knowledge based systems methods: a practitioner's guide.* London: Prentice Hall.

Haythorn, W. (1994). What is object-oriented design? *Journal of Object-Oriented Programming 7*(1), 67–78.

Heuvelink, G. (1993). *Error propagation in quantitative spatial modelling: applications in Geographical Information Systems.* University of Utrecht.

Heuvelink, G. (1998). *Error propagation in environmental modelling with GIS.* Research monographs in GIS. London: Taylor and Francis.

Heuvelink, G., P. Burrough, and A. Stein (1989). Propagation of errors in spatial modeling in GIS. *International Journal of Geographical Information Systems 3*(4), 303–322.

Hugill, M. (1988). *Advanced statistics.* London: Unwin Hyman.

Hunter, A. (1996). *Uncertainty in information systems.* Advanced topics in computer science. London: McGraw Hill.

Hunter, G. (1999). Reporting spatial data quality: from concepts to reality. In W. Shi, M. Goodchild, and P. Fisher (Eds.), *Proceedings of the International Symposium on Spatial Data Quality,* pp. 343–353.

ISO/TC211 (1999). Draft international standard: geographic information — quality principles. Technical Report ISO/CD 19113.3, International Standards Organisation.

Jacobson, I., G. Booch, and J. Rumbaugh (1999). *The unified software development process.* Massachusetts: Addison-Wesley.

Johnsson, K. (1994). Segement-based land-use classification for SPOT data. *Photogrammetric Engineering and Remote Sensing 60*(1), 47–53.

Kaindl, H. (1994). Object-oriented approaches in software engineering and artificial intelligence. *Journal of Object-Oriented Programming 6*(8), 38–45.

Kaindl, H. and M. Snaprud (1991). Hypertext and structured object representation: a unifying view. In *Proceedings 3rd ACM conference on hypertext,* pp. 345–358.

Kainz, W. (1995). Logical consistency. In S. Guptill and J. Morrison (Eds.), *Elements of spatial data quality,* Chapter 6, pp. 109–137. Oxford: Elsevier Science.

Keefer, B., J. Smith, and T. Gregoire (1988). Simulating manual digitising error with a statistical model. In *Proceedings GIS/LIS,* Volume 2, Falls Church, pp. 475–483. ASPRS/ACM.

Kernighan, K. and D. Ritchie (1988). *The C programming language* (Second ed.). New Jersey: Prentice Hall.

Kiiveri, H. (1997). Assessing, representing and transmitting positional uncertainty in maps. *International Journal of Geographical Information Science 11*(1), 33–56.

Kösters, K., B.-U. Pagel, and H.-W. Six (1997). GIS-application development with GEOOOA. *International Journal of Geographical Information Science 11*(4), 307–335.

Lakoff, G. (1987). *Women, fire and dangerous things: what categories reveal about the mind.* Chicago: Iniversity of Chicago Press.

Lakoff, G. and M. Johnson (1980). *Metaphors we live by.* Chicago: University of Chicago Press.

Lanter, D. and H. Veregin (1992). A research paradigm for propagating error in layer-based GIS. *Photogrammetric Engineering and Remote Sensing 58*(6), 825–833.

Laser-Scan (1996). *Gothic module reference manual.* Version 2.1.

Lein, J. (1992). Modelling environmental impact using an expert-geographic information system. In *GIS/LIS,* Volume 1, pp. 436–443.

Leung, Y. (1987). On the imprecision of boundaries. *Geographical Analysis 19*(2), 125–151.

Leung, Y. and J. Yan (1998). A locational error model for spatial features. *International Journal of Geographical Information Science 12*(6), 607–620.

Lewis, J., S. Henry, D. Kafura, and R. Schulman (1992). On the relationship between the object-oriented paradigm and software reuse: an empirical investigation. *Journal of Object-Oriented Programming 5*(4), 35–41.

Lilburne, L., G. Benwell, and R. Buick (1996). GIS, expert systems and interoperability. In *Proceedings of the first international conference on geocomputation,* Volume 2, pp. 527–541.

Linsey, T. and J. Raper (1993). HyperArk: a task-oriented hypertext GIS interface. *International Journal of Geographical Information Systems 7*(5), 435–452.

Loftus, C., E. Sherratt, R. Gautier, P. Grandi, D. Price, and M. Tedd (1995). *Distributed software engineering.* London: Prentice Hall.

Lundin, B., J. Yan, and J.-P. Parker (1990). Data quality reporting methods for digital geographical products at Statistics Canada. In *Challenge for the 1990s: GIS. Proceedings national conference,* Ottawa, pp. 236–251. CISM.

Maffini, G., M. Arno, and W. Bitterlich (1989). Observations and comments on the generation and treatment of error in digital GIS data. In M. Goodchild and S. Gopal (Eds.), *Accuracy of spatial databases,* Chapter 5, pp. 55–68. London: Taylor and Francis.

Maggio, R. (1987). The role of the geographic information system in the expert system. In *GIS '87. Proceedings of the 2nd International Conference,* Volume 2, pp. 685–692.

Maguire, D. and J. Dangermond (1991). The functionality of GIS. In D. Maguire, M. Goodchild, and D. Rhind (Eds.), *Geographical information systems,* Volume 1, Chapter 21, pp. 319–335. Essex: Longman.

Mark, D. and F. Csillag (1989). The nature of boundaries on 'area-class' maps. *Cartographica 26*(1), 65–78.

Mark, D. and F. Zhan (1992). Object oriented spatial knowledge representation and processing: formalisations of core classes and their relationships. In *Proceedings Fifth International Symposium on Spatial Data Handling,* Volume 2, pp. 662–671.

Merchant, J. (1994). GIS based groundwater pollution hazard assessment: a critical review of the DRASTIC model. *Photogrammetric Engineering and Remote Sensing 60*(9), 1117–1128.

Merriam-Webster (1999). Webster WWW dictionary. http://www.m-w.com/dictionary. Last modified 18 October 1999, last accessed 21 December 1999.

Meyer, B. (1992). Applying "design by contract". *IEEE Computer 25*(10), 40–51.

Mikhail, E. (1978). *Observations and least squares.* New York: IEP Dun Donnely.

Milne, P., S. Milton, and J. Smith (1993). Geographical object-oriented databases: a case study. *International Journal of Geographical Information Systems 7*(1), 39–55.

Minsky, M. (1975). A framework for representing knowledge. In P. Winston (Ed.), *The psychology of computer vision,* pp. 211–277. New York: McGraw-Hill.

Moellering, H. (Ed.) (1997). *Spatial database transfer standards 2: characteristics for assessing standards and full descriptions of the national and international standards in the world.* Oxford: Elsevier Science.

Monarchi, D. and G. Puhr (1992). A research typology for object-oriented analysis and design. *Communications of the ACM 35*(9), 35–47.

Morrison, J. (1995). Spatial data quality. In S. Guptill and J. Morrison (Eds.), *Elements of spatial data quality,* Chapter 1, pp. 1–12. Oxford: Elsevier Science.

National Committee for Digital Cartographic Data Standards (1988). The proposed strandard for digital catographic data. *American Cartographer 15*(1), 11–142.

Nerson, J. (1992). Applying object-oriented anlysis and design. *Communications of the ACM 35*(9), 63–74.

Newcomer, J. and J. Szajgin (1984). Accumulation of thematic map error in digital overlay analysis. *American Cartographer 11*(1), 58–62.

Nyerges, T. (1991). Geographic information abstraction: conceptual clarity for geographical information systems. *Environment and Planning A 23,* 1483–1499.

Open GIS Consortium (1999). OpenGIS specifications. http://www.opengis.org/techno/specs.htm. Last modified 27 August 1999, last accessed 21 December 1999.

Open GIS Consortium Technical Committee (1999a). Introduction to interoperable geoprocessing and the OpenGIS specification. http://www.opengis.org/techno/guide/guide980615/Chap1.ht. Chapter 1, last modified 30 June 1999, last accessed 21 December 1999.

Open GIS Consortium Technical Committee (1999b). Introduction to interoperable geoprocessing and the OpenGIS specification. http://www.opengis.org/techno/guide/guide980615/Chap4.ht. Chapter 4, last modified 30 June 1999, last accessed 21 December 1999.

Openshaw, S., M. Charlton, and S. Carver (1991). Error propagation: a Monte Carlo simulation. In I. Masser and M. Blakemore (Eds.), *Handling geographical information,* pp. 78–101. New York: Longman.

Partridge, C. (1994). Modelling the real world: are classes abstractions or objects? *Journal of Object-Oriented Programming 7*(7), 39–45.

Peckham, J. and F. Maryansk (1988). Semantic data models. *ACM Computing Surveys 20*(3), 153–189.

Perkal, J. (1966). On the length of empirical curves. *Discussion paper 10.* Michigan InterUniversity Community of Mathematical Geographers.

Professional Geo Systems (1999). Professional Geo Systems home page. http://www.pgs.nl/. Last modified 29 October 1999, last accessed 21 December 1999.

Qiu, J. and G. Hunter (1999). Managing data quality information. In W. Shi, M. Goodchild, and P. Fisher (Eds.), *Proceedings of the International Symposium on Spatial Data Quality,* pp. 384–395.

Quinlan, J. (1983). Learning efficient classification procedures and their application to chess end games. In R. Michalski, J. Carbonell, and T. Mitchell (Eds.), *Machine learning: an artificial intelligence approach,* Chapter 15, pp. 463–482. California: Morgan Kauffmann.

Ralston, B. (1994). Object oriented spatial analysis. In S. Fotheringham and P. Rogerson (Eds.), *Spatial analysis and GIS,* pp. 165–186. London: Taylor and Francis.

Ramlal, B. and J. Drummond (1992). A GIS uncertainty subsystem. In *Archives ISPRS Congress XVII,* Volume 29.B3, pp. 356–362.

Raper, J. and D. Livingston (1995). Development of a geomorphological spatial model using object oriented design. *International Journal of Geographical Information Systems 9*(4), 359–384.

Rational Software (1999). UML resource centre. http://www.rational.com/uml. Last accessed 21 December 1999.

Reinke, K. and G. Hunter (1999). Communicating quality in spatial information: notification — the first step. In W. Shi, M. Goodchild, and P. Fisher (Eds.), *Proceedings of the International Symposium on Spatial Data Quality,* pp. 66–75.

Révész, G. (1988). *Lambda calculus, combinators, and functional programming.* Cambridge: Cambridge University Press.

Riley, G. (1999). CLIPS: a tool for building expert systems. http://www.ghg.net/clips/CLIPS.html. Last modified 28 August 1999, last accessed 21 December 1999.

Robinson, G. and M. Jackson (1985). Expert systems in map design. In *Proceedings Auto-Carto 7,* pp. 430–437.

Robinson, V. and A. Frank (1987). Expert systems applied to problems in geographic information systems. In *Proceedings Auto-Carto 8,* pp. 510–519.

Rosenberger, J. (1998). *Teach yourself CORBA.* Indiana: Sams Publishing.

Rosenfield, G. (1986). Analysis of thematic map classification error matrices. *Photogrammetric Engineering and Remote Sensing 52*(5), 681–686.

Rumbaugh, J., M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen (1991). *Object-oriented modeling and design*. New Jersey: Prentice Hall.

Russell, B. (1912). *The problems of philosophy*. London: Oxford University Press.

Russell, S. and P. Norvig (1995). *Artificial Intelligence: a modern approach*. New Jersey: Prentice Hall.

Russomanno, D. (1998). Utility network derivation from legacy source data for feature based AM/FM systems. *International Journal of Geographical Information Science 12*(5), 445–463.

Shannon, C. (1948). A mathematical theory of communication. *The Bell System Technical Journal 27*, 379–423, 623–656.

Shepherd, I. (1991). Information integration and GIS. In D. Maguire, M. Goodchild, and D. Rhind (Eds.), *Geographical information systems* (First ed.), Volume 1, Chapter 22, pp. 337–360. Essex: Longman.

Shi, W. (1998). A generic statistical approach for modelling error of geometric features in GIS. *International Journal of Geographical Information Science 12*(2), 131–143.

Shi, W. and W. Guo (1999). Modeling topological relationships of spatial objects with uncertainties. In W. Shi, M. Goodchild, and P. Fisher (Eds.), *Proceedings of the International Symposium on Spatial Data Quality*, pp. 487–495.

Shortridge, A. and M. Goodchild (1999). Communicating uncertainty for global data sets. In W. Shi, M. Goodchild, and P. Fisher (Eds.), *Proceedings of the International Symposium on Spatial Data Quality*, pp. 59–65.

Skidmore, A., F. Walford, P. Luckananurug, and P. Ryan (1996). An operational expert system for mapping forest soils. *Photogrammetric Engineering and Remote Sensing 62*(5), 501–511.

Smith, B. (1996). NLIS in 1996: the pilot project expands. *Mapping Awareness 10*(2), 22–24.

Smith, T. and J. Yiang (1991). Knowledge-based approaches in GIS. In D. Maguire, M. Goodchild, and D. Rhind (Eds.), *Geographical information systems*, Chapter 27, pp. 413–425. Essex: Longman.

Sondheim, M., K. Gardels, and K. Buehler (1999). GIS interoperability. In P. Longley, M. Goodchild, D. Maguire, and D. Rhind (Eds.), *Geographical information systems* (Second ed.), Volume 1, Chapter 24, pp. 327–358. Essex: Longman.

Srinivasan, A. and J. Richards (1993). Analysis of GIS spatial data using knowledge-based methods. *International Journal of Geographical Information Systems 7*(6), 479–500.

Stanton, R. and H. MacKenzie (1989). Expert system techniques in spatial planning problems. In R. Quinlan (Ed.), *Applications of expert systems*, Volume 2, Chapter 10, pp. 170–181. Glasgow: Turing Institute Press.

Stoms, D. (1987). Reasoning with uncertainty in intelligent GIS. In *GIS '87. Proceedings of the 2nd International Conference*, Volume 2, pp. 693–699.

Story, M. and R. Congalton (1986). Accuracy assessment: a user's perspective. *Photogrammetric Engineering and Remote Sensing 52*(3), 397–399.

Sun Microsystems (1999a). JDBC technology.
http://www.javasoft.com/products/jdbc/index.html. Last modified 15 December 1999, last accessed 21 December 1999.

Sun Microsystems (1999b). What is the java platform?
http://www.javasoft.com/nav/whatis/index.html. Last modified 19 October 1999, last accessed 21 December 1999.

Tang, A., T. Adams, and E. Usery (1996). A spatial data model design for feature-based geographical information systems. *International Journal of Geographical Information Systems 10*(5), 643–659.

Thapa, K. and J. Bossler (1992). Accuracy of spatial data used in geographic information systems. *Photogrammetric Engineering and Remote Sensing 58*(6), 835–841.

Tobler, W. (1970). A computer movie: simulation of population change in the Detriot region. *Economic Geography 46*, 234–240.

Unwin, D. (1995). Geographical information systems and the problem of error and uncertainty. *Progress in Human Geography 19*(4), 549–558.

US Geological Survey (1999a). SDTS logical specification: completeness. http://mcmcweb.er.usgs.gov/sdts/SDTS_standard_nov97/part1b16.html. Last modified 20 November 1997, last accessed 21 December 1999.

US Geological Survey (1999b). SDTS logical specification: logical consistency. http://mcmcweb.er.usgs.gov/sdts/SDTS_standard_nov97/part1b15.html. Last modified 20 November 1997, last accessed 21 December 1999.

US Geological Survey (1999c). SDTS standard. http://mcmcweb.er.usgs.gov/sdts/. Last modified 16 June 1999, last accessed 21 December 1999.

van der Wel, F., R. Hootsmans, and F. Ormeling (1994). Visualization of data quality. In A. MacEachren and D. Taylor (Eds.), *Visualization in modern cartography*, pp. 313–331. Oxford: Pergamon.

Vckovski, A. (1998). *Interoperable and distributed processing in GIS*. Research monographs in GIS. London: Taylor and Francis.

Veregin, H. (1989). Error modeling for the map overlay operation. In M. Goodchild and S. Gopal (Eds.), *Accuracy of spatial databases*, pp. 3–18. London: Taylor and Francis.

Veregin, H. (1995). Developing and testing an error propagation model for GIS overlay. *International Journal of Geographical Information Systems 9*(6), 595–616.

Veregin, H. (1996). Error propagation through the buffer operation for probability surfaces. *Photogrammetric Engineering and Remote Sensing 62*(4), 419–428.

Voss, G. (1991). *Object-oriented programming: an introduction*. Berkeley: Osborne McGrw-Hill.

Walker, P. and D. Moore (1988). SIMPLE: an inductive modelling and mapping tool for spatially-oriented data. *International Journal of Geographical Information Systems 2*(4), 347–363.

Wang, F. and G. Hull (1996). Fuzzy representation of geographical boundaries in GIS. *International Journal of Geographical Information Systems 10*(5), 573–590.

Waterman, D. (1986). *A guide to expert systems*. Massachusetts: Addison-Wesley.

Wesseling, C. and G. Heuvelink (1993). Manipulating qualitative attribute accuracy in vector GIS. In *Proceedings Fourth European Conference on Geographical Information Systems*, Volume 1, pp. 675–684.

Woodhead, N. (1991). *Hypertext and hypermedia: theory and applications*. Wilmslow: Sigma Press.

Worboys, M. (1992). A generic model for planar geographical objects. *International Journal of Geographical Information Systems 6*(5), 353–372.

Worboys, M. (1994). Object-oriented approaches to geo-referenced information. *International Journal of Geographical Information Systems 8*(4), 385–399.

Worboys, M. (1995). *GIS: a computing perspective*. London: Taylor and Francis.

Worboys, M. (1998). Imprecision in finite resolution spatial data. *GeoInformatica 2*(3), 257–279.

Worboys, M. (1999). Relational databases and beyond. In P. Longley, M. Goodchild, D. Maguire, and D. Rhind (Eds.), *Geographical information systems* (Second ed.), Volume 1, Chapter 26, pp. 373–384. Essex: Longman.

Worboys, M., H. Hearnshaw, and D. Maguire (1990). Object-oriented data modelling for spatial databases. *International Journal of Geographical Information Systems 4*(4), 369–383.

Yourdon, E. (1989). *Modern structured analysis*. London: Prentice-Hall International.

Zelen, M. and N. Severo (1965). Probability functions. In M. Abramowitz and I. Stegun (Eds.), *Handbook of mathematical functions*, Chapter 26, pp. 925–996. New York: Dover.

Zhan, F. (1991). Structuring the knowledge of cartographic symbolisation — an object-oriented approach. In *Proceedings Auto-Carto 10*, pp. 247–260.

Zhan, F. and B. Buttenfield (1995). Object-oriented knowledge-based symbol selection for visualizing statistical information. *International Journal of Geographical Information Systems 9*(3), 293–315.

Zhu, X. (1996). An architecture for knowledge-based spatial decision support systems. In M. Kraak and M. Molenaar (Eds.), *Advances in GIS research II; Proceedings Seventh International Symposium on Spatial Data Handling*, Volume 1, pp. 3A13–24.

Appendix A

Object systems

A.1 Equational theory

Equational theory of untyped ς-calculus (Abadi and Cardelli 1996a, p63).

(Eq Symm)

$$\frac{\vdash b \leftrightarrow a}{\vdash a \leftrightarrow b}$$

(Eq Select)

$$\frac{\vdash a \leftrightarrow a'}{\vdash a.l \leftrightarrow a'.l}$$

(Eq x)

$$\frac{}{\vdash x \leftrightarrow x}$$

(Eq Trans)

$$\frac{\vdash a \leftrightarrow b \qquad \vdash b \leftrightarrow c}{\vdash a \leftrightarrow c}$$

(Eq Update)

$$\frac{\vdash a \leftrightarrow a' \qquad \vdash b \leftrightarrow b'}{\vdash a.l \Leftarrow \varsigma(x)b \leftrightarrow a'.l \Leftarrow \varsigma(x)b'}$$

(Eval Select) where $a \equiv [l_i = \varsigma(x_i)b_i\ ^{i \in 1..n}]$

$$\frac{j \in 1..n}{\vdash a.l_j \leftrightarrow b\langle x \leftarrow a \rangle}$$

(Eq Object) (l_i distinct)

$$\frac{\vdash b_i \leftrightarrow b'_i \quad \forall i \in 1..n}{\vdash [l_i = \varsigma(x_i)b_i\ ^{i \in 1..n}] \leftrightarrow [l_i = \varsigma(x_i)b'_i\ ^{i \in 1..n}]}$$

(Eval Update) where $a \equiv [l_i = \varsigma(x_i)b_i\ ^{i \in 1..n}]$

$$\frac{j \in 1..n}{\vdash a.l_j \Leftarrow \varsigma(x)b \leftrightarrow [l_j = \varsigma(x)b, l_i = \varsigma(x_i)b_i\ ^{i \in (1..n)-\{j\}}]}$$

A.2 Simple object fragments

Simple ς-calculus typing and equational rules (Abadi and Cardelli 1996a, p329).

(Type Object)

$$\frac{\Gamma \vdash B_i \quad \forall i \in 1..n}{\Gamma \vdash [l_i : B_i\ ^{i \in 1..n}]}$$

(Val Select)

$$\frac{\Gamma \vdash a : [l_i : B_i\ ^{i \in 1..n}] \quad j \in 1..n}{\Gamma \vdash a.l_j : B_j}$$

(Sub Object)

$$\frac{\Gamma \vdash B_i \quad \forall i \in 1..n+m}{\Gamma \vdash [l_i : B_i\ ^{i \in 1..n+m}] <: [l_i : B_i\ ^{i \in 1..n}]}$$

(Val Object) where $A \equiv [l_i : B_i\ ^{i \in 1..n}]$

$$\frac{\Gamma, x_i : A \vdash b_i : B_i \quad \forall i \in 1..n}{\Gamma \vdash [l_i = \varsigma(x_i : A)b_i\ ^{i \in 1..n}] : A}$$

(Val Update) where $A \equiv [l_i : B_i\ ^{i \in 1..n}]$

$$\frac{\Gamma \vdash a : A \quad \Gamma, x : A \vdash b : B_j \quad j \in 1..n}{\Gamma \vdash a.l_j \Leftarrow \varsigma(x : A)b : A}$$

A.3 Untyped error-sensitive object system

The three classes *Unc, Rep* and *List* make up the core of an untyped error-sensitive object system.

$$\begin{array}{l}
List \triangleq [new = \varsigma(z)[init = \varsigma(s)z.init(s),\ get\_size = \varsigma(s)z.get\_size(s),\\
\qquad get\_match = \varsigma(s)z.get\_match(s),\ remove\_match = \varsigma(s)z.remove\_match(s),\\
\qquad add = \varsigma(s)z.add(s),\ size = \varsigma(s)z.size(s),\\
\qquad get = \varsigma(s)\lambda(i)[\,],\ match = \varsigma(s)\lambda(w)\lambda(l)l,\\
\qquad remove = \varsigma(s)\lambda(w)\lambda(l)l,\ obj = \varsigma(s)[\,],\\
\qquad tail = \varsigma(s)[add = \varsigma(t)\lambda(o)((s.obj \Leftarrow o).tail \Leftarrow List.new).init,\ size = \varsigma(t)\lambda(i)i]],\\
\quad init = \lambda(s)\\
\quad ((s.get \Leftarrow \varsigma(t)List.get(t)).match \Leftarrow \varsigma(t)List.match(t)).remove \Leftarrow \varsigma(t)List.remove(t),\\
\quad get\_match = \lambda(s)\lambda(w)s.match(w, List.new),\\
\quad remove\_match = \lambda(s)\lambda(w)s.remove(w, List.new),\\
\quad add = \lambda(s)\lambda(o)s.tail.add(o),\\
\quad get\_size = \lambda(s)s.tail.size(0),\\
\quad size = \lambda(s)\lambda(i)s.tail.size(i+1),\\
\quad get = \lambda(s)\lambda(i)\underline{\text{if}}\ i \equiv 0\ \underline{\text{then}}\ s.obj\underline{\text{else}}\ s.tail.get(i-1),\\
\quad match = \lambda(s)\lambda(w)\lambda(l)\underline{\text{if}}\ w \equiv s.obj.name\ \underline{\text{then}}\ s.tail.match(w, l.add(s.obj))\\
\qquad \underline{\text{else}}\ s.tail.match(w, l),\\
\quad remove = \lambda(s)\lambda(w)\lambda(l)\underline{\text{if}}\ w \equiv s.obj.name\ \underline{\text{then}}\ s.tail.remove(w, l)\\
\qquad \underline{\text{else}}\ s.tail.match(w, l.add(s.obj))]
\end{array}$$

$$\begin{aligned}
&\mathit{Unc} \triangleq [\mathit{new} = \varsigma(z)[\mathit{qlist} = \varsigma(s)z.\mathit{qlist}(s),\ \mathit{mlist} = \varsigma(s)z.\mathit{mlist}(s),\\
&\qquad\quad \mathit{get\_rep} = \varsigma(s)z.\mathit{get\_rep}(s),\ \mathit{set\_rep} = \varsigma(s)z.\mathit{set\_rep}(s),\\
&\qquad\quad \mathit{test\_res} = \varsigma(s)z.\mathit{test\_res}(s),\ \mathit{test\_met} = \varsigma(s)z.\mathit{test\_met}(s)],\\
&\qquad \mathit{qlist} = \lambda(s)\mathit{List}.\mathit{new},\quad \mathit{mlist} = \lambda(s)\mathit{List}.\mathit{new},\\
&\qquad \mathit{get\_rep} = \lambda(s)\lambda(w)s.\mathit{qlist}.\mathit{get\_match}(w),\\
&\qquad \mathit{set\_rep} = \lambda(s)\lambda(q)((s.\mathit{test\_res}(q)).\mathit{qlist}.\mathit{add\_rep}(q)).\mathit{test\_met}(q),\\
&\qquad \mathit{test\_res} = \lambda(s)\lambda(q)\underline{\mathrm{if}}\ q.\mathit{is\_res}\ \underline{\mathrm{then}}\ s.\mathit{qlist} \Leftarrow s.\mathit{qlist}.\mathit{remove\_match}(q.\mathit{name})\ \underline{\mathrm{else}}\ s,\\
&\qquad \mathit{test\_met} = \lambda(s)\lambda(q)\underline{\mathrm{if}}\ q.\mathit{is\_met}\ \underline{\mathrm{then}}\ (\underline{\mathrm{if}}\ s.\mathit{mlist}.\mathit{get\_match}(q.\mathit{name}).\mathit{get\_size} \equiv 0\\
&\qquad\qquad \underline{\mathrm{then}}\ s.\mathit{mlist} \Leftarrow s.\mathit{mlist}.\mathit{remove\_match}(q.\mathit{name})\ \underline{\mathrm{else}}\ s)\ \underline{\mathrm{else}}\ s]
\end{aligned}$$

$$\begin{aligned}
&\mathit{Rep} \triangleq [\mathit{new} = \varsigma(z)[\mathit{qlist} = \varsigma(s)z.\mathit{qlist}(s),\ \mathit{mlist} = \varsigma(s)z.\mathit{mlist}(s),\\
&\qquad\quad \mathit{get\_rep} = \varsigma(s)z.\mathit{get\_rep}(s),\ \mathit{set\_rep} = \varsigma(s)z.\mathit{set\_rep}(s),\\
&\qquad\quad \mathit{test\_res} = \varsigma(s)z.\mathit{test\_res}(s),\ \mathit{test\_met} = \varsigma(s)z.\mathit{test\_met}(s),\\
&\qquad\quad \mathit{is\_res} = \varsigma(s)z.\mathit{is\_res}(s),\ \mathit{is\_met} = \varsigma(s)z.\mathit{is\_met}(s)],\\
&\qquad \mathit{qlist} = \mathit{Unc}.\mathit{qlist},\ \mathit{mlist} = \mathit{Unc}.\mathit{mlist},\\
&\qquad \mathit{get\_rep} = \mathit{Unc}.\mathit{get\_rep},\ \mathit{set\_rep} = \mathit{Unc}.\mathit{set\_rep},\\
&\qquad \mathit{test\_res} = \mathit{Unc}.\mathit{test\_res},\ \mathit{test\_met} = \mathit{Unc}.\mathit{test\_met},\\
&\qquad \mathit{is\_res} = \mathit{false},\ \mathit{is\_met} = \mathit{false}]
\end{aligned}$$

Appendix B

Core expert system rules

$\forall x, y \in \mathit{Quality} \quad \mathit{SubTerm}(x,y) \Rightarrow \mathit{SubSet}(x,y)$

If x is a sub-term (*SubTerm*) of y, x is also a sub-set (*SubSet*) of y.

(B.1)

$\forall x, y, z \in \mathit{Quality} \quad \mathit{SubSet}(x,y) \land \mathit{SubSet}(y,z) \Rightarrow \mathit{SubSet}(x,z)$

If x is a sub-term of y, and y is a sub-term of z, then x is a sub-set of z.

(B.2)

$\forall x, y \in \mathit{Quality} \quad \mathit{SubTerm}(x,y) \land \mathit{Selected}(x) \Rightarrow \mathit{Selected}(y)$

If the term x is selected (*Selected*) then its super-term y is also selected.

(B.3)

$\forall x, y \in \mathit{Quality} \quad \mathit{SubTerm}(x,y) \land (\mathit{SelectRatio}(x) \geq \mathit{SelectRatio}(y)) \Rightarrow \mathit{PossibleElement}(x)$

where

$$\mathit{SelectRatio}(x) = \frac{\mathit{Card}(\{a|\forall a \in \mathit{Quality} \quad \mathit{SubTerm}(a,x) \land \mathit{Selected}(a)\})}{\mathit{Card}(\{a|\forall a \in \mathit{Quality} \quad \mathit{SubTerm}(a,x)\})}$$

The ratio of the cardinality (*Card*) of the set of selected sub-terms to the cardinality of the set of all sub-terms for a particular term is denoted by *SelectRatio*. For any super-term and sub-term, if the selected ratio of the sub-term is greater than or equal to that of the super-term, the sub-term is possibly a quality element (*PossibleElement*).

(B.4)

$$\forall y \in \mathit{Quality} \quad \mathit{PossibleElement}(y) \land (\neg\exists x \in \mathit{Quality} \quad \mathit{PossibleElement}(x) \land \mathit{SubSet}(x,y)) \\ \Rightarrow \mathit{Element}(y)$$

If there are no sub-set terms x of a possible element y that are also possible elements, then y is definitely an element (*Element*).

(B.5)

$$\forall y \in \mathit{Quality} \quad \mathit{Selected}(y) \land (\neg\exists x \in \mathit{Quality} \quad \mathit{Selected}(x) \land \mathit{SubSet}(x,y)) \Rightarrow \mathit{Attribute}(y)$$

Any selected term y that does not have any selected sub-set terms, is an attribute (*Attribute*).

(B.6)

$$\forall x,y \in \mathit{Quality} \quad \mathit{Attribute}(x) \land \mathit{SubTerm}(x,y) \land (\neg\exists z \in \mathit{Quality} \quad \mathit{SubSet}(x,z) \land \mathit{Element}(z)) \\ \Rightarrow \mathit{Element}(y)$$

The super-term y of any attribute x that does not already have an element super-set term z is an element.

(B.7)

$$\forall x,y \in \mathit{Quality} \quad \mathit{Attribute}(x) \land \mathit{Element}(y) \land \mathit{SubSet}(x,y) \Rightarrow \mathit{AttributeOf}(x,y)$$

An attribute term x can be an attribute of (*AttributeOf*) an element y if it is a sub-set of that term.

(B.8)

$$\text{where } \mathit{Quality} = \{x | \forall x \; \mathit{QTerm}(x)\} \tag{B.9}$$

Appendix C

Program code and documentation

Program source code, compiled code and documentation is contained on the CD-ROM that accompanies this thesis. The directory structure of the CD-ROM is organised as shown below.

```
CD-ROM
  |---> index.html .................... Index to CD-ROM contents
  |---> bin ........................... Executable and compiled code
          |---> error_loader .......... Unix executable
          |---> error_loader.bat ...... Windows executable
  |---> doc ........................... Documentation files
          |---> HTML .................. HTML documentation
                  |----> eaGIS2 ....... Error-aware documentation
                  |----> error ........ Error package documentation
                  |----> servlet ...... Servlet documentation
          |---> images ................ Image files
  |---> lib ........................... Library and class files
          |---> error.jar ............. Error jar file
  |---> src ........................... Source code
          |---> c ..................... C source code
                  |----> api .......... Gothic API C code
                  |----> ep_gothic .... Error propagation C code
                  |----> socket ....... Gothic TCP socket C code
                  |----> vp_gothic .... Variance propagation code
          |---> java .................. Java source code
                  |----> eaGIS2 ....... Error-aware GIS Java code
                  |----> error ........ Error-sensitive prototype
                  |----> kbs2 ......... Induction algorithm code
                  |----> servlets ..... Java servlet code
          |---> lull .................. Lull source code
                  |----> propagation .. Error propagation code
                  |----> server2 ...... Gothic server code
                  |----> uncertainty .. Core Gothic code
```