INTEGRATING SPACE AND TIME IN AN OBJECT-BASED GIS: A CASE STUDY OF PUBLIC BOUNDARY EVOLUTION

by

Monica Wachowicz

PhD Geographical Information Systems

The University of Edinburgh

1995

INTEGRATING SPACE AND TIME IN AN OBJECT-BASED GIS: A CASE STUDY OF PUBLIC BOUNDARY EVOLUTION

Abstract

Although space and time are inherently related concepts, they have not yet been fully incorporated into GIS. Integration of the domains of space and time in order to represent evolutionary processes in the real-world is still a major issue in spatio-temporal modelling in GIS. In this investigation, a time-geographic (*time-geography*) approach has been chosen as an appropriate framework for elucidating our understanding of the temporal structuring of space within a GIS context.

The concepts developed in time-geography are similar to those acquired through our understanding of the real-world. They are not related to any particular level of abstraction, either geographically (nation-region-centre), temporally (year-month-day), or demographically (population-group-individual). This is particularly important for understanding the dynamics of the human activity involved in a particular problem domain rather than monitoring time as a useful attribute value within a GIS.

Therefore, time-geography provides a *framing tool* for understanding a problem domain in spatial and temporal contexts, as well as a *modelling tool* for representing the passage of time and the mechanisms of change within a GIS.

Embedding the time-geographic framework within a GIS is accomplished by using the object-oriented analysis and design method proposed by Booch. This method assures the object-oriented decomposition in which the complexity of the concepts developed in the time-geographic framework are transposed to a meaningful collection of objects. These interact in different kinds of scenarios within a *spatio-temporal data model.* In addition, inheritance has been adopted as the incremental change mechanism within the spatio-temporal data model because of its usefulness in subclassing the evolutionary processes of incremental change within the time-geographic framework.

A taxonomy of change is proposed on the basis of the spatio-temporal data model, designed for embedding the time-geographic approach. A version management approach plays an important role in managing changes within the spatio-temporal data model. The version management method developed by Ahmed and Navathe has been selected because of its consistency with object-oriented concepts. The main concepts of this method are described and its incorporation into the spatio-temporal data model is presented in the thesis.

The GIS application chosen to evaluate the spatio-temporal model is the study of the evolutionary aspects of public boundaries. Public boundaries represent the line of physical contact between administrative units in Great Britain. The arrangement of public boundaries forms an irregular tessellation of polygons that represent the whole hierarchy of local government and European Constituency areas. Approximately 3,000 changes occur to political boundaries in England every year.

The spatio-temporal data model describes the processes which most of the public boundaries would pass through in their lifespan as well as the different evolutionary states which reveal the changes occurring over a public boundary lifespan.

Finally, the implementation aspects involved in incorporating the spatio-temporal data model into an object-based GIS are discussed. The Smallworld GIS is used for implementing the spatio-temporal data model and the issues arising from this implementation of the theoretical framework are identified and assessed.

Declaration

I declare that this Dissertation represents my own work, and that where the work of others have been used, it has been duly accredited.

Signed

Monica Wachowicz

March, 1995

Acknowledgments

I would like to thank my supervisors, Dr. Richard G. Healey and Dr. Michael F. Worboys, for their unceasing support and valuable advice during my research. My thanks also go to Les Rackham and Alan Hardiman of Ordnance Survey, who have significantly contributed in expanding my view about the evolution of public boundaries.

I would like to thank Steve Dowers and Chris Place for their support and advice about the computing facilities available at the University, and I would like to thank Penny Legg in making my meetings with Richard Healey happen.

The Department of Geography has provided a stimulating environment in which to work and my thanks go to those who helped to make it that way. In particular, Bhaskar Ramachandran, Trevor Rotzien, Charles Stewart, Jonathan Makin and Cristoph Corves for their thought-provoking discussions and interesting ideas. I will always remember Vicen Carrio-Lluesma, Juan Suarez, Monica Decanini, Manoel Claudio da Silva, Marcelo Vieira, Gwo-Jinn Hwang, Xuan Zhu and Bob Hodgart for their help and friendship during my years in Edinburgh. I would like to thank Marianne Broadgate and Ian Alexander for their endless support and care, specially during my last year in Edinburgh.

I am grateful for the support and encouragement I have received from my family, both in Brazil and USA. Distance could never deter them from being a continuous source of love and encouragement. I would like to express my warmest appreciation to my father-in-law Herman for his financial help.

With a risk of under-expressing my gratitude, I would like to thank my husband Victor for his perpetual optimism, continuous source of love and reassurance.

Contents

List of Figures

List of Tables

Chapter 1
Introduction

1.1 Exploring the Main Concepts Involved in Space and Time

The meaning of time has been perceived differently in different cultures and at different times. In some religions, time is thought to run in cycles around the riddle of the past, present, and future. Other major religions teach that time was created and is destined to end one day in a terrifying climax.

In the 5th century BC, Parmenides and Zeno proclaimed that change is logically impossible, reality is motionless and time is an illusion. As an antithesis, Heraclitus considered the flow of time to be the very essence of human reality. Isaac Newton, in the late 17th century, regarded space and time dimensions as vast containers inside which all experiences are situated. Kant also contributed to this thought by assuring his readers that space and time are only real as long as there is a human perspective to experience them.

Most difficulties in understanding the meaning of time are a result of our spatially based conceptions. The notion that time 'moves' is an extension of the common use of the word. Movement, in its usual sense, involves a change of position relative to time. So what is time moving relative to? Is there a time scale relative to which time moves? How fast is that moving, and relative to what?

Albert Einstein's influence on the 20th century has been to demonstrate that time, like space, depends on the motion of the observer. In his theory of relativity, published in 1905, he asserted that the speed of light in a vacuum is an absolute speed limit. One consequence of this is that the flow of time appears to decrease with an increase in velocity. Therefore, events which are simultaneous to one observer may be temporally distinct to another one.

Philosophical and physical attempts at exploring space and time have involved efforts to represent space and time as different scales in social, geographical and physical studies.

> "Geography's study of processes over space and time is neither new or unique. ... in the 1950s, a variety of approaches for studying space-time phenomena has evolved. Andrew Clarks's early work on historical geography demonstrated that changing spatial patterns could be studied as "geographical change" [Clark 1959, 1962]. Cliff and Ord [1981] later examined change through time by scanning a sequence of maps, searching for systematic autocorrelation structures in space-time in order to specify "active" and "interactive" processes. Perhaps the best-known efforts within the field of geography that made explicit use of time as a variable in the study of spatial processes are Hägerstrand's models of diffusion and time geography ..." [Peuquet 1994, p.441].

Torsten Hägerstrand, a Swedish geographer, unfolded the time-geographic approach in the 1960s. He examined space and time within a "general equilibrium model, (in which) it is assumed that every individual performs multiple roles, it is also implicitly admitted that location in space cannot effectively be separated from the flow of time." [Hägerstrand 1970, p.10]. In his model, an individual or an entity follows a space-time path, starting at the point of birth and ending at the point of death. Such a path can be depicted over space and time by collapsing both spatial and temporal dimensions into a space-time path. Time and space are seen as inseparable.

Time geography has provided a foundation for recognising paths of entities through space and time and for uncovering potential spatio-temporal relationships among such paths. Moreover, its application in various areas has produced the concept of a "continuous path" in order to represent the experience occurring during the lifespan of an entity. This experience is in fact conceptualised as a succession of changes of locations of an entity over its space-time path.

Most of the applications using time geography have been devoted to modelling individual activity paths within a period of time, analysing the pattern of activities for any individual path, as well as simulating individual activity paths. This research proposes a new means for applying the time-geographic approach. It intends to employ the concept of a space-time path developed in time geography for representing spatio-temporal data within a Geographical Information System (GIS).

The term representation will be used throughout this dissertation meaning the "... arrangement or organization of data defined within an explicit set of primitive elements, attributes (properties), and relationships. Such an arrangement serves to preserve the information inherent in the data for subsequent use in problem-solving or analytical evaluation. A representation can be purely conceptual, but any representation must be expressed in formalized mathematical or programming-language terms for computer implementation. A formalized representation intended for computer implementation is know as a *data model*." [Peuquet 1994, p. 459].

Representing spatial data in a GIS has been achieved by defining entities in space in an explicit (vector format) or implicit (raster or grid-cell format) manner [Burrough 1986]. Thus, representing temporal data in a GIS has been regarded as implementing an additional dimension or axis in a geometric space. In a vector format, the spatio-temporal data representation has emphasised the (x,y,z,t) coordinate system, in which (x,y,z) coordinates locate an entity and (t) coordinate represents time. In this case, the objective is primarily related to **organising space over time.**

An empty space is organised into partitions (layers) and the entities that inhabit this space are embedded into these partitions. In fact, a partition serves as a skeleton for organising entities located in space. This involves a space-to-entity organisation: first give a region of space, then locate the entities that inhabit this region. The incorporation of time is achieved by creating a series of snapshots showing the changes in the spatial entities which have taken place, usually using oscillating colours, temporal sequences of static maps, and supplementary graphs. However, the fundamental thesis of this research is that this kind of homogeneous four dimensional representation is inadequate for most applications in a GIS, the main reason being that this kind of spatio-temporal representation treats time and space in the same manner, i.e. as referential dimensions.

In contrast, space-time paths in time geography emphasise the need for integrating space and time dimensions in a more abstract conceptual sense. Space and time dimensions are fused into a space-time path in such a way that space is defined on the basis of entities and the spatio-temporal relationships between those entities. In this case, the emphasis is in **representing entities in space over time.** As an entity is supposed to change its state or move in a highly dynamic way, a true 'object representation' is needed in order to provide an appropriate spatio-temporal data representation of this entity.

This object representation emphasises how entities are represented, independently of how space is organised. However, the representation of space and entities must be compatible, and their interrelationship is the key issue in designing spatio-temporal data models in GIS. The challenge is to ensure that relationships between partitions of space and entities that inhabit these partitions can be computed efficiently in both directions: space-to-entities as well as entities-to-space.

Such an object representation will introduce a new capability into time geography for identifying spatio-temporal patterns directly derived from the similar characteristics among the trajectories of a group of space-time paths. Consequently, the overall goal of this research is to integrate space and time within a GIS in such a way that a time-geographic framework is built up to represent entities in space over time. The evolution of political boundaries has been chosen as the case study due to the richness found in the 'boundary-making' processes involved in the history of human civilisation.

1.2 The Evolution of Political Boundaries: A Case Study

Until the end of the 19th century, territories were separated by zones which defined the political frontiers in the landscape. These zones played an important role in reinforcing cultural and social differences between adjacent countries. Gradually, there was an increase of interest in replacing most of these types of zones (political frontiers) by lines (political boundaries). Such a process might have happened due to strategic reasons or conflicts of interest in the territory between neighbouring countries.

Since then, political boundaries have been established to define precise lines in the landscape, which have preferably been attached to geographical features. They have been defined and redefined according to the changes of political and geographical circumstances.

The process of boundary-making and the conflicts involved in boundary disputes have attracted most of the attention from researchers wishing to understand the role of boundaries in the history of human civilisation. Prescott provides an extensive survey illustrating the historical facts and the main characteristics involved during the evolution of political boundaries:

> "A significant part of the history of several countries concerns the struggle for territory, and the identification of previous national boundaries on a single map provides a shorthand account of stages in the progress to the present pattern of states. Often the events which established new boundaries were sufficiently important to mark the division between important periods in the political history of countries or in the diplomatic and military history of continents." [1987, p.1].

Research studies carried out on the evolution of political boundaries [Harley 1989, Paddison 1983, Prescott 1987, Sack 1986] have shown several aspects of the geographical significance of the presence of a boundary, for example, the influence the location of a political boundary has over the whole development of the border landscapes. It has had significant effects upon attitudes of the neighbouring inhabitants and upon the policies of the countries on either side of the boundary. Disputes, historical conflicts, and sometimes even wars have occurred over time due to political boundary definitions having an incomprehensible description in official documents and maps, as well as having a dubious location in the landscape.

Several attempts have been made to generalise some possible rules describing the evolution of political boundaries [Ratzel 1897], the possible categories in the process of boundary-making [Haushofer 1927], and finally the principal stages in the evolution of a boundary [Lapradelle 1928, Jones 1945]. Although the history of each political boundary is unique, some of these efforts have contributed in clarifying the essential elements and procedures by which the dynamic nature of the evolution of political boundaries can be analysed geographically.

In this research, time geography has played an important role as a *framing tool* for understanding the evolution of political boundaries. Four framing categories have been utilised for designing the time-geographic framework: the representation framework, the evolution framework, the mutation framework, and the implementation framework. The overall merit of designing a time-geographic framework can be evaluated by its particular function in serving as an exploratory means of developing space-time paths involved in the evolution of political boundaries. This has entailed an analysis of the evolution of political boundaries on the basis of the following questions:

- How should a political boundary be described: in context, space and time?
- Which are the processes of evolution acting upon political boundaries?
- How might political boundaries be altered: by what actions, where and when?

Having selected the evolution of political boundaries, a real application has been chosen to assist in the design of the framework based on time geography. This application is concerned with the evolution of public boundaries.

1.3 The Ordnance Survey and Public Boundaries

External political boundaries have been drawn to delimit a country's sovereignty over its territory. Internal political boundaries have been defined to separate the social-economical-political activities and development policies within a country.

Having opted to examine the evolution of internal political boundaries, a concrete application has been chosen to assist in the design of the time-geographic framework. This application has played an important role in understanding a number of issues: 1) how data can evolve over time; 2) how evidence of change is manifested; and finally, 3) how semantic constructs for modelling space and time should be devised. The chosen application deals with the evolution of public boundaries in Great Britain.

The Ordnance Survey is the national mapping agency for Great Britain which "... has had a statutory requirement to ascertain, mere and record public boundaries since 1841. As a result, it has become the main depository for, and authority on, public boundaries in Great Britain." [Rackham 1987, p.6]. The Ordnance Survey on the 1 of April 1991 created a spatial data set at 1:10,000 scale containing the digital out lines of the public boundaries in England. In order to support this product, the 'Boundary-Line' system has been defined which produces snapshots of the location of public boundaries at specific dates.

This pioneering initiative has been influential in consolidating the perspective of this research towards the design of a spatio-temporal data model which could contribute in a number of ways to the development of the actual Boundary-Line data management system used by the Ordnance Survey. In particular, the main contribution would be a GIS-based model for handling the evolution of public boundaries in an explanatory and planning mode.

1.4 The Role of Geographical Information Systems

In the last few decades, Geographical Information Systems (GIS) have emerged as distinct tools for analysing spatial data sets pertaining to social, environmental and economic studies. They have permitted the integration of a variety of existing models with spatial data sets. Examples include the innovative GIS-based monitoring model developed by Blom and Löytönen [1993] to monitor current epidemics in Finland, including that of HIV. This model integrates spatial-diffusion, spatial-interaction and environmental modelling into a GIS-based model for monitoring the passing of infectious diseases between individuals. The goal of this model is to provide disease-specific forecasts for the future course of an epidemic.

The European Groundwater Project [Thewessen *et al.* 1992] is one example of the integration of existing, non-spatial simulation models with spatial data sets. The result is the design of a GIS-based environmental model which provides rapid and coherent access to the most significant causes and effects of ground water contamination. In this example, physical and chemical models have been integrated into the GIS-based model resulting in the identification of serious threats to the quality and quantity of ground water resources in the European Union.

Different uses for GIS-based models in non-traditional applications are still being uncovered by researchers and developers. An array of perspectives and new outlooks are expected to flow from a wide spectrum of multidisciplinary studies. Some indications of this include the interest in temporal aspects of GIS-based models and the common problems of managing historical information within a GIS. Ramachandran points out the challenges to be undertaken:

> "Temporal change has enormous philosophical and practical implications when considered within a GIS context, which involves modelling the real-world entities, their structure and behaviour. These implications are sharply accentuated for a temporal GIS, whose capabilities do not fully exist as yet. Presently existing atemporal GIS have no memory of the earlier states of the database, which means they are static snapshot databases which describe only one data state. For capturing a history of changes an object undergoes, an application data structure is required to record the evolving temporal state of that object." [1992, p. 11]

The modelling difficulties involved in attempting to incorporate temporal semantics within a GIS context are apparent from the literature on temporal database design as well as spatio-temporal modelling. Research in temporal database design

[Al-Taha *et al.* 1994, Snodgrass *et al.* 1993, Soo 1991] views time as a record of the evolving changes of entities. The majority of temporal database research has addressed issues related to version management aspects (time stamping, concurrency control, update processing), as well as enhancements required for the logical components (schema evolution, temporal query language syntax) and the physical structure (storage structure, access methods, query optimisation, query language features, underlying data model) of the database management systems [Tansel *et al.* 1993].

The problem of incorporating temporality into spatial databases in GIS has raised further issues due to the application-specific semantics of modelling changes in GIS-based models [Armenakis 1993, Langran 1993, Worboys 1992]. In particular these issues relate to: 1) the support of changes in topological relationships among entities; 2) the dynamic representation and visualisation of historical states of entities; and 3) the development of a unifying structure with both spatial and temporal dimensions. An unconstrained spatio-temporal model for supporting the three temporal states of past, present and future of entities is vital for GIS but unfortunately it has still to be realised in either theoretical or operational terms.

However, the time-geographic framework introduces a robust object representation for conceiving a spatio-temporal data model. In this case, time geography plays an important role as a *modelling tool* for representing the passage of time and the mechanisms of change within a spatio-temporal data model. Such a modelling approach for dealing with time and space within a GIS has not been previously explored, and is an attempt to demonstrate a more encompassing perspective than is currently available for integrating space and time domains within a GIS. The time-geographic spatio-temporal data model being proposed here will be referred to as the spatio-temporal data model throughout the thesis.

1.5 The Design of the Spatio-Temporal Data Model

Developing a spatio-temporal data model based on the time-geographic framework is fraught with a whole assortment of problems. These are essentially related to understanding specifications of the problem domain, the modelling constructs, and the mapping between the model and its implementation in a GIS.

Object-oriented modelling techniques have recently proliferated in terms of their concepts, expressions, extensions, and applications areas. They present a strong support for modularity, reusability, and understandability. Their support for a consistent conceptual model throughout the system analysis and design phases has been an important encouragement to their use in many application areas [Graham 1994].

The use of object-orientation is required in order to obtain the object representation for the intended spatio-temporal data model and the design elements for implementing this model into a GIS. Object-oriented methods offer "a concise methodology (which) allows for a focus of attention on conceptual aspects of the system when necessary, and to concentrate on the details of the design without being overwhelmed." [Rubenstein and Hersh 1984, p.39].

Object-oriented concepts have been employed for modelling the space-time paths developed in the spatio-temporal data model into a meaningful set of classes of objects interacting in different kinds of scenarios. Object-orientation has also provided the mechanism for establishing temporal data management support within the spatio-temporal data model.

Finally, an implementation of such a spatio-temporal data model is undertaken as a 'proof-of-concept'. Implementing the spatio-temporal data model has been the means by which the ideas developed in the model could be empirically tested and refined within an integrated environment such as a GIS. The Smallworld GIS[1] has been used for the implementation, since it offers an object-based environment. In this research, the implementation aspects of the spatio-temporal data model have highlighted the challenges in dealing with time and space in a GIS context.

1.6 Aims of this Research

This dissertation introduces a spatio-temporal data model which integrates space and time domains in a GIS context, based on the concepts developed in the time-geographic and object-oriented approaches. The aims of the research are:

- define the object representation as a new means of representing spatio-temporal data in GIS;
- provide a deeper understanding of the meaning of space-time paths and use this to identify a suitable role for dealing with the passage of time and the mechanisms of change within a spatio-temporal data model in GIS;
- converge both approaches: time geography and object-orientation, by associating space-time paths of a time-geographic framework, with the modelling constructs of an object oriented method;
- contribute to the development of the Boundary-Line data management system of the Ordnance Survey by providing a different perspective about spatio-temporal data modelling in GIS;

[1] Smallworld GIS is trademark of Smallworld Systems Ltd., Cambridge, England.

- undertake the implementation of the spatio-temporal data model into a GIS system as 'proof-of-concept'.

1.7 Summary of the Chapters

Chapter 2 introduces the main concepts involved in the time-geographic approach which have been used for developing the spatio-temporal data model. The feasibility of incorporating this approach into a GIS is discussed on the basis of the previous implementation attempts which have been found in the literature.

Chapter 3 addresses the human activities involved in the process of boundary-making for public boundaries in Great Britain. The time-geographic framework developed for the evolution of public boundaries is described using four framing categories. These are termed the representation framework, the evolution framework, the mutation framework, and the implementation framework.

Chapter 4 provides a historical background to temporal data management by summarising the efforts in the areas of object-oriented methods, temporal databases, and version management approaches. This review illustrates the diverse efforts employed in managing change in spatio-temporal data models.

Chapter 5 presents the spatio-temporal data model using the notation and concepts proposed by the Booch method. A comprehensive set of diagrams expressing the important aspects of the spatio-temporal data model illustrates the fundamental decisions which have been taken in order to handle the time-geographic framework.

Chapter 6 emphasises the importance of having temporal data management within the spatio-temporal data model. A taxonomy of change is defined for the spatio-temporal data model and the versioning method proposed by Ahmed and Navathe [1991] is introduced as a temporal data management mechanism. This method has been implemented in the spatio-temporal data model and its object-oriented features are described.

Chapter 7 provides an overview of the Smallworld GIS (Release 2.0.2) in terms of its proprietary database, its programmatic interface, and its available tools and utilities. The main concepts utilised within the Smallworld GIS are also explained. This chapter explores the issues involved in implementing the spatio-temporal data model.

Chapter 8 presents the results from implementing the spatio-temporal data model. A prototype implementation illustrates the working of the spatio-temporal data model.

Chapter 9 completes the thesis with a summary and the main conclusions of this dissertation.

Chapter 2
The Temporal Structuring of Space in GIS

2.1 Introduction

"Time and the way it is handled has a lot to do with structuring space." [Hall 1966, p.163]. Although space and time are concepts inherently related, they have not yet been fully incorporated into a GIS context. Integrating space and time in order to represent the evolution of the real-world is still an issue in spatio-temporal modelling in GIS. Some fundamental questions arise from examining the temporal structuring of space within a GIS:

- How is human activity simultaneously associated with space and time?

- How can the interaction between time and space be represented in a GIS context?

- How can both space and time be incorporated into a GIS?

This chapter provides the foundation for answering these questions on the basis of the time-geographic concepts. One of the main concepts provided by the time-geographic approach is the depiction of an absolute space-time path in order to represent the passage of time and the mechanisms of change. "This form of representation is unique, not because of its space-time concepts, but its application to societal and environmental problems." [Lenntorp 1976, p.10].

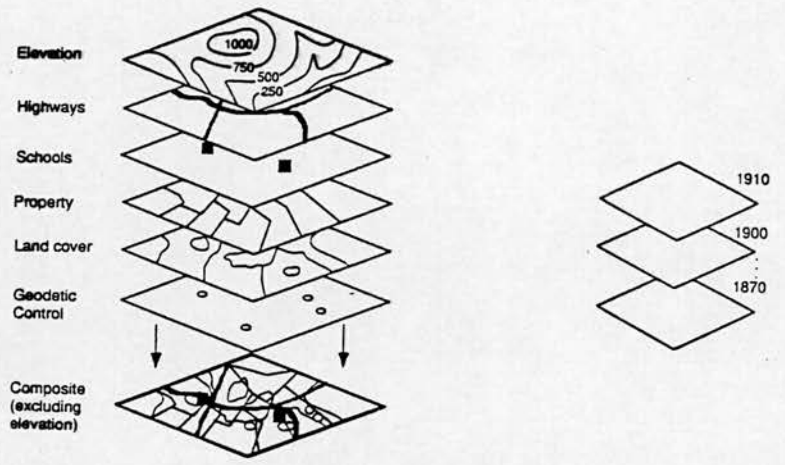
The main concepts involved in the time-geographic approach which have been employed for developing our spatio-temporal data model are described in this chapter. The feasibility of incorporating this approach into a GIS is discussed on the basis of previous implementation attempts. The time geographic framework is presented as a *framing tool* for representing the spatio-temporal data involved in the evolution of the real-world, as well as a *modelling tool* for representing the passage of time and the mechanisms of change within a GIS.

2.2 The Time-Geographic Approach

In a highly correlated manner, social theory combines the compositional and the contextual approaches [Thrift 1983]. Thrift states that:

> "In the compositional approach ... human activity is split up into a set of broad structural categories founded on the property of 'alikeness' and derived via a formal - logical method based on the tool of abstraction. These categories are then recombined as an explanation of society, or at least, of parts of it." [1983, p.27].

Essentially, the compositional approach focuses on form and structure. Real-world phenomena are broken up into sets, and after that, their parts are joined to form the whole. So far, this approach has been used in GIS by making spatially depicted classifications grouped into layers or sets of themes (for example, geology, hydrology, and land cover) between points or periods of time. In other words, space is grouped along the spatial dimension after a categorisation and time is grouped along the time dimension after some sort of periodisation. The historical process is explained based on similarity/dissimilarity between aggregations (layers) as illustrated in Figure 2.1.



(a) (b)
Figure 2.1 - Spatio-temporal layers: the main representation used in GIS
(a) Themes (b) Time Periods
(Source: Laurini and Thompson 1992, p.7)

"In the contextual approach ... human activity is treated as a social event in its immediate spatial and temporal setting and the categories so derived are based on a property of 'togetherness' that must not be split asunder." [Thrift 1983, p.28].

Basically, this approach synthesises on the basis of continuity and connectedness. On continuity, because it searches for events through time that exhibit a certain spatio-temporal pattern, instead of creating layers or time periods. On connectedness, mainly because it is important in this approach to distinguish entities in space that are influenced by human activities over time.

In the 1960s, the pioneering work of Torsten Hägerstrand unfolded the time-geographic approach (time geography) based upon the contextual theory. The research which emerged from the Royal University of Lund, Sweden has served to consolidate the concepts developed in the time-geographic approach, mostly in the work of Hägerstrand and his students and collaborators Lenntorp, Mårtensson and Carlstein.

In time geography, space and time are considered orthogonal dimensions which become fused into a space-time path representing the historical trajectory of a lifespan of an entity (Figure 2.2). Time is viewed as the path which orders events, which separates causes from effects, and which synchronises and integrates human activity. Space is viewed as the path which represents a point-object in space denoting that each entity occupies space. Time and space are considered as the constituents of place. It is the timing component which gives structure to space and thus evokes the notion of *place*. Place is "... a pause in movement..." [Parkes and Thrift 1978, p.120].

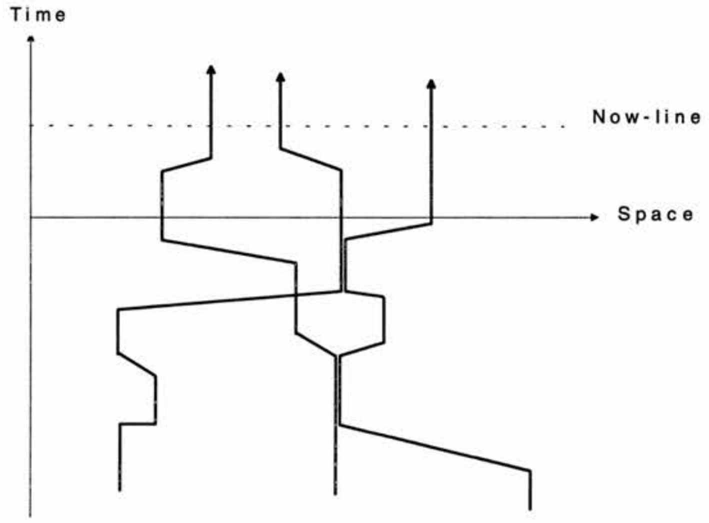


Figure 2.2 - Space-time paths of three entities
(Source: Hägerstrand 1975, p.8)

The lifespan of an entity is represented by a singular and continuous space-time path. The grouping of space-time paths represents the interaction among lifespans of different entities. The sense of past, present, and future depends on where the observer is placed in the space-time path. The observer has a nearness view around every place pertaining to a space-time path [Hägerstrand 1975].

Such underlying concepts in time geography have inspired their application in most of the empirical frameworks and modelling applications which have involved planning and related fields. Pred [1977] provides an overview of the usefulness of time-geographic frameworks in several contexts, among them, those concerned with regional development policies, nation-wide physical planning, and urbanisation and settlement policies. Many of the applications of time-geographic frameworks have been implemented by the Swedish government in order to provide adequate job-market opportunities, and a satisfactory level of social and cultural services. Some examples are the accessibility simulation of daily individual activities in urban environments and regions [Lenntorp 1978], comparative studies of living conditions in different populated regions [Mårtensson 1978], and analysis of various activities in the quaternary sector, mainly concerned with employment distribution [Olander and Carlstein 1978].

> "Owing to the circumstances under which ... (time-geography) has been developed, its contents, and its applications to date, there is a great danger that Hägerstrand's time geographic framework will be mistakenly construed as nothing more than a planning tool. On the contrary, ... the potential usefulness of the framework ... is of much greater range." [Pred 1977, p.213].

In developing these ideas, this research proposes that the potential usefulness of the time-geographic framework should be exploited as a modelling tool in order to capture the role of space and time within a GIS.

2.3 The Main Concepts of Time Geography

The concepts developed in time geography are similar to those acquired through our understanding of the real-world. They are not related to any particular level of abstraction, either geographically (nation-region-centre), temporally (year-month-day), or demographically (population-group-individual) [Lenntorp 1976]. This is particularly important for understanding the human activity involved in a specific application rather than considering time as a useful dimensional value.

The use of a continuous and indivisible space-time path attached to a specific lifespan of an entity is one of the main features of the time-geographic approach. This absolute representation is an important ordering element of **events**, **changes**, and **constraints** which can occur during the lifespan of an entity. It is a descriptive form to reveal the interdependence between events with time and space [Lenntorp 1976]. "Space and time are to be jointly treated ... because when events are seen located together in a block of space-time, they inevitably expose relations which cannot be traced if those events are bunched into classes and drawn out of their place in the block, i.e., conventionally analyzed." [Pred 1977, p.210].

The sense of time is change. "It's man's sense of time which enables him to navigate in physical and social spaces, towards place." [Parkes and Thrift 1978, p.128]. Each place located in the space-time path reveals a particular change occurring over the lifespan of an entity. The continuity of the space-time path and its angularity with the time axis are relevant in classifying types of changes according to their respective duration - the wider the angle, the shorter is the duration of a change. Three main types of change can be characterised in a space-time path [Parkes and Thrift 1978]:

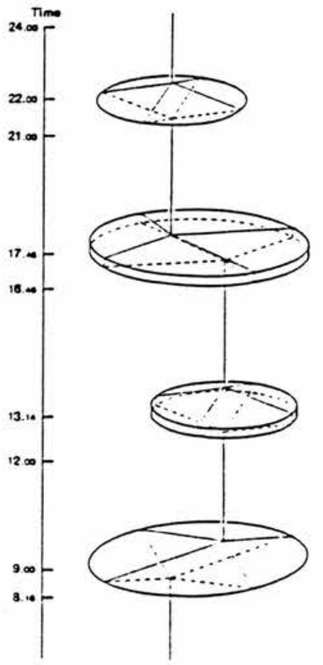
- long-term changes modifying the environment;

- medium-term changes transforming cultures; and

- short-term changes making up the history of day-to-day incidents.

Another feature of the space-time path is its ability to reveal the structure of constraints within a system [Lenntorp 1976]. Every lifespan of an entity is placed into a 'space-time constraint' which is represented by its space-time path. For example, the space-time path delineates the movement of an entity over space and time. Essentially, an ensemble of constraints can restrain the historical evolution of a set of entities at a macro-level as well as the individual behaviour of each entity at a micro-level. In other words, the time-geographic framework can specify the necessary constraints for most of the interactions involving human activity which can be appropriately described on the basis of individual or group behaviour.

The space-time path is a continuous and indivisible trajectory in which an observer is constantly in motion. An observer always moves his/her position in a space-time path in such a way that his/her position is never the same as the position of any other observer. Consequently, the motion of an observer is limited to a set of circumstances described by the constraints which have been defined for a space-time path. This set is called the Potential Path Space (Figure 2.3), represented as a prism in the time-geographic framework. It comprises space-time positions for which the possibility of being included in the observer's trajectory is greater than zero [Lenntorp 1976]. A general procedure cannot be developed for deriving or calculating Potential Path Spaces from empirical data. Each application domain has to be analysed in order to generate its actual Potential Path Spaces.



An example of Potential Path Spaces in
an individual's daily programme.
Four possible individual paths are followed
over a 24-hour period which start at 08.16 hours.

Figure 2.3 - An example of Potential Path Spaces
(Source: Lenntorp 1976, p.35)

2.4 Time Geography and GIS

There has been significant research investigation into formulating a representation for time at the conceptual level in a GIS context. Langran [1992b] asserts that a GIS application must comprise three sorts of data: states, events, and evidence. In extending this classification, Kraak and MacEachren [1994] point out a further differentiation between events and episodes. Likewise, Peuquet and Wentz [1994] emphasise the need for understanding geographic processes in which the mechanisms of change as well as the patterns of change have to be analysed through time. The sequence of events through time is viewed as the spatio-temporal manifestation of certain processes.

In essence, these temporal elements, developed in a GIS context so far, define a state as a single time slice of data, an event as the moment in time an occurrence takes place, an episode as the length of time during which change occurs, and a piece of evidence as data describing the source of state and event data.

This research contributes to the understanding of the above described notions by distinguishing three main elements of the time-geographic approach. They are: changes, events, and constraints. This research also formulates a new use for these three elements in a way that their interaction in a space-time path can be incorporated into a GIS. Therefore, the ensuing sections describe the new application of these three elements in developing a time-geographic framework.

2.4.1 Change as an Element of the Space-Time Path

In the lifespan of an entity, changes occur on its space-time path describing either the evolution of its inner existence, or the occurrence of a mutation of its state.

In the first case, change is represented by the *evolutionary states* of a space-time path. Evolutionary states are related to the historical evolution which regards change as the adaptation of a system to its environment by the process of differentiation and increasing structural complexity (Figure 2.4). In the second case, change is related to the theory of revolution which emphasises the importance of conflict or struggle as the principal mechanisms of change. It characterises the alteration of the direction of the space-time path. This fact is represented by a *revolutionary state*, of the space-time path (Figure 2.4).

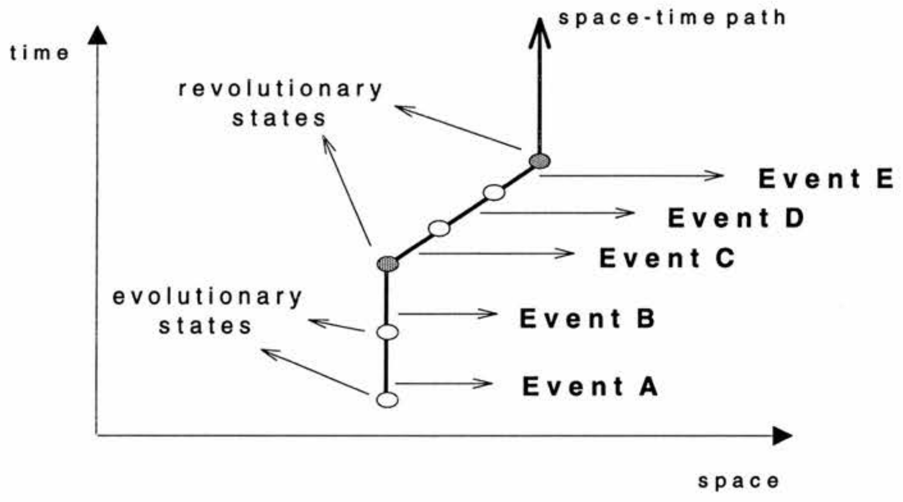


Figure 2.4 - Elements of a space-time path

Revolutionary states tend to represent the short-term changes which can occur during the lifespan of an entity. In the boundary evolution case study, revolutionary states represent changes in the positions of boundaries. Such changes can occur due to man-made alterations in the position of a boundary by open cast mining or erosion of the ground features. Changes can also occur due to natural causes, of which the most common example is the displacement of a boundary position with the displacement of the watercourse by rivers and streams.

In contrast, evolutionary states are related more to the long-term or medium-term changes of boundaries. They are associated with the systematic changes of public boundaries after passing through their historical processes such as the allocation process which refers to the political decision on the distribution of territory, or the demarcation process which is concerned with the marking of the boundary on the ground.

2.4.2 Event as an Element of the Space-Time Path

The event concept in time geography should not be mistaken by those familiar with event-oriented representation, or with update-oriented representation. Both representations have been formerly discussed as pragmatic solutions for representing spatio-temporal data within a GIS. In an event-oriented representation, the principal view of data is described by the moments of change. Only the events responsible for any kind of change over an entity are relevant to an event-oriented representation. Changes over an entity are seen as versions of this entity [Langran 1993].

Conversely, in an update-oriented representation, the view of the data focuses on presenting the currency of the data which belongs to an entity [Langran 1993]. The lifespan of an entity is represented by the occurrence of all kinds of updates, even those updates which are not responsible for any change in a state of this entity. Events are treated as components of a state of an entity. A state represents the occurrence of an update over an entity.

A different perspective in representing events is found in the space-time path described in time geography. Events are not only necessarily related to the creation of changes (versions) or update activities in GIS. In time geography, events embrace the human activity over the real-world that results from a *process study* on the basis of the application domain.

"A process study seeks to identify the rules which govern spatio-temporal sequences, in such a form that the rules are interpretable in terms of the results of the sequence, in terms of the exogeneous variables which influence the sequence, and in terms of the mechanisms by which exogeneous and endogeneous influences give rise to the results which the sequence itself records."
[The Dictionary of Human Geography 1994, p.478].

Therefore, the notion of events being placed in a space-time path gives the meaning of an existence (evolutionary states) and mutation (revolutionary states) for change. The interaction between changes (evolutionary state/revolutionary state) and events of a space-time path is illustrated in Figure 2.4. An event is represented by a line segment of a space-time path of which the length and the angularity indicate the temporal duration of the event according to long-, medium- or short-term changes. A revolutionary state and an evolutionary state are depicted as having a specific place in the space-time path. Hägerstrand [1970] has previously named such places in the space-time path as stations. The space-time path allows the connection, ordering and synchronisation of events and changes occurring in the life-span of an entity.

2.4.3 Constraint as an Element of the Space-Time Path

The possibilities of defining the events and the changes pertaining to a space-time path are immense for any application domain involved in a GIS context. Consequently, a space-time path has to be placed according to an ensemble of constraints which define the circumstances (Potential Path Spaces) where the space-time positions can be or could be placed over space and time. This involves the formulation of general rules of behaviour for a space-time path in terms of its position, scale, and orientation. In fact, these are the properties which are responsible for the density of space-time paths within a time-geographic framework.

Position denotes the location of a space-time path, while the scale property associates a spatio-temporal resolution with a space-time path. The orientation property represents the direction of a space-time path.

Hägerstrand [1970] describes three types of constraints: capability constraints, coupling constraints, and authority constraints. Capability constraints are those which limit the activities of an entity because of its biological construction and/or the tools it can command. Coupling constraints define where, when, and for how long, an entity has to join other space-time paths of other entities. Authority constraints control the space-time path of a given entity or a given group of entities. Whatever the type of constraint being employed in a time-geographic framework, this will require the formulation of rules of a space-time path's behaviour in terms of its position, scale, and/or orientation.

2.4.4 Implementing the Time-Geographic Approach in a GIS

Although time geography is an effective approach to dealing with space and time in an integrated manner within a GIS, it has been neglected so far. After analysing the feasibility of handling space-time concepts of time geography within a GIS, Miller affirms that "Geographic Information Systems, through their ability to manipulate and analyse spatial data, can allow more widespread use of the space-time perspective (of time geography) in spatial modelling and analysis." [1991, p.30].

Few examples are available for illustrating the attempts at applying the time geography approach within a GIS. Miller [1991] has generated Potential Path Areas (PPAs) for a transportation network application on the basis of arcs in the network that are feasible to travel. The mainframe version 5.0 of ARC/INFO was used for the implementation in order to handle a set of nodes and arcs, keep records of locations within the system and handle numerous travel times at both nodes and arcs in the network. Although "... ARC/INFO NETWORK can meet the requirements for standard GIS applications, it is inefficient and unwieldy in meeting the more specialised needs of the network PPA procedure. Whereas ARC/INFO is certainly not representative of all GIS software, it does provide a benchmark which indicates the problems encountered by analysts who wish to use GIS technology in more specialized research and modelling." [Miller 1991, p.299].

Miller also points out the main requirements in applying time geography in GIS:

- the time-geographic approach requires data at a detailed level of spatial resolution in order to obtain an effective analysis using a GIS;
- the GIS must be able to address the behavioural aspects of data to generate a more realistic operational Potential Path Space;
- the favoured GIS to implement a time-geographic framework must be able to store and manipulate topological relationships to avoid adding unnecessary complexity to the framework;
- the derivation and manipulation of space-time prisms (Potential Path Spaces) in a GIS might be accomplished by developing the framework to support space and time constraints. A modular structure with inflexibility of key commands and procedures can render a GIS unable to derive effectively the desired space-time prism framework.

Another example is the application of time geography for simulating an individual's daily shopping behaviour within a GIS [Makin 1992]. The results show how time and space constraints on people's shopping movements affect shops' potential takings and profits. Makin explores the potential of using a GIS to structure spatial relationships according to which routes are accessible to each other, and where the buildings are located on the route network.

He also points out the potential of having implemented his simulation model in an object-based GIS such as Smallworld for simulating the behaviour of people's movements:

- the data are not generalised or aggregated within the GIS;
- the entities are allowed to move and interact with their constraints in space and time in a way that long-term behavioural patterns can be analysed;
- the simulation model is expressed in terms that do not require abstraction into mathematics;
- the whole system can be organised in a modular fashion in which sub-systems are created for reducing the complexity of the model by minimising the amount of data and the number of interactions.

Such examples illustrate the potential perspectives of applying time geography in a GIS context. This investigation proposes the use of the time-geographic framework for formulating a spatio-temporal model. This embraces an approach in dealing with time and space in a GIS which has not been explored thoroughly before.

2.5 Conclusions

The main objective of this chapter was to describe the concepts of the time-geographic approach: events, changes (evolutionary/revolutionary states) and constraints. These concepts play a significant role in the development of a spatio-temporal data model in the way that they have been used for answering the following questions:

- How is human activity simultaneously associated with space and time?

  In time geography, the conceptualisation of a space-time path is an attempt at fusing space and time to represent human activity, which takes place in a continuous sequence of events, changes, and constraints.

- How can the interaction between time and space be represented in a GIS context?

  The temporal structuring of space is consolidated by synchronising and integrating events, changes, and constraints for each space-time path pertaining to each entity rather than treating time as a dimensional value attached to a lifespan of an entity. The semantics of events, changes, and constraints are not related to any particular level of abstraction, either geographically (nation-region-centre), temporally (year-month-day), or demographically (population-group-individual). In fact, the space-time path of the time-geographic framework is the actual 'object representation' of an entity. This representation emphasises how entities are represented rather than how space is organised. It provides an appropriate spatio-temporal data representation of this entity.

Very little previous work is available in the published literature on using time geography with GIS. This investigation proposes the use of the time-geographic approach as a framing tool through which an application domain is framed by ordering a sequence of events and by describing the mechanisms of change. Events are placed in a space-time path in order to give an interpretation of the existence (evolutionary states) and mutation (revolutionary states) circumstances of a lifespan of an entity.

Once the time-geographic framework is created, it becomes an important conceptual representation for revealing different types of constraints in space-time paths which can be attached to specific circumstances. Identifying such circumstances within a time-geographic framework facilitates the development of modelling scenarios within a spatio-temporal data model in a GIS. Each scenario is based on a circumstance (Potential Path Space) which is a set of space-time paths describing the real-world in a particular space-time place.

- Is it possible to incorporate both space and time into a GIS using the time-geographic approach?

  Yes, it is but with some challenges involved. The following chapters explore some of the challenges to be encountered in developing and implementing a spatio-temporal data model based on the time-geographic approach, within a GIS.

In order to explain how a time-geographic framework can be developed, a case study has been selected concerned with the evolution of public boundaries in Great Britain. This is described in detail in the next chapter.

Chapter 3

Framing Public Boundary Evolution in Time and Space

3.1 Introduction

Public boundaries represent the line of physical contact between administrative units in Great Britain. The arrangement of public boundaries forms an irregular tessellation of polygons that represent the whole hierarchy of local government and European constituency areas. Approximately 3,000 changes occur to public boundaries in England every year, increasing the volume of data at the rate of 5 to 10 megabytes a year [Rackham 1992].

Time and space are factors which have produced the history of public boundaries. Records of the history of each public boundary can be found in documents and published material such as parliamentary Acts and Orders retained by the Ordnance Survey. These contain legal information defining the line on the basis of which each public boundary is attached to a physical feature on the ground. This involves the delimitation process which establishes the exact location of the true (legal) public boundary on the ground. All public boundaries have been delimited in Great Britain.

Conversely, demarcation is the process by which the position of the boundary is ratified on the ground by surveyors, on the basis of the information provided by the documentation concerning the delimitation. Maps portraying the public boundaries when they have been delimited and demarcated contain important historical information about identification of the physical features on the ground. In addition, they portray information about different types of border landscapes through which a public boundary is drawn.

Fieldwork records kept by Ordnance Survey are another valuable source of historical information about the evolution of public boundaries. One example is the register of discrepancies between the true (legal) description of a public boundary and its demarcation on the ground. Another example is the collection of perambulation cards noting the occurrence of some change in the location of a physical feature on the ground from its previous public boundary demarcation.

This chapter introduces the time-geographic framework which has been developed for creating the **object representation** of the evolution of public boundaries. Boundaries have been considered as appropriate objects in the context of the case study because of the richness found in the historical processes involved in the change of a boundary's position. This chapter also describes the processes which most of the records of public boundaries would pass through in their lifespan; the different evolutionary states associated with the processes acting upon public boundaries; and finally, the principal revolutionary states revealing the mutations occurring over public boundary lifespans. Four framing categories have been utilised for designing the time-geographic framework: the representation framework, the evolution framework, the mutation framework and the implementation framework. Such frameworks are crucial for an overall understanding of the role of the object representation of the time-geographic framework, and therefore they are described in detail in the next sections.

3.2 The Representation Framework : The Evolution of Public Boundaries in the Context of Time and Space

Consideration of how the evolution of thousands of political boundaries should be represented in the context of space and time has led to the exploration of concepts of the time geography approach, as a means of reconstructing space-time paths related to public boundary lifespans. Public boundaries are linear objects that experience a succession of changes in their positions during their lifespans. The history of each public boundary is unique and shows the geographical significance of a public boundary over the development of landscapes, social-economical policies, and historical conflicts. The semantic richness found in 'boundary-making' processes (allocation, delimitation, demarcation) provides a means for applying the concept of a space-time path of time geography.

The perspective of having the fusion of time and space represented by space-time paths for recreating the history of each public boundary has been the major reason for selecting Hägerstrand's approach. In addition, the perspective of handling this conceptual representation of time and space to design a public boundary evolution model can encourage the evaluation of its use within the Boundary-Line data management system being developed by Ordnance Survey.

3.2.1 The Object Representation of Space and Time

For simplification, as has been suggested by Hägerstrand, the representation of space is along only one dimension to maintain the clarity of the proposed representation framework and to give a better visualisation of the evolution of public boundaries. The same simplification has been adopted for the time dimension.

The representation framework is primarily constructed from the point of view of the fusion of space and time dimensions rather than the dimensions themselves and their orthogonality. The space-time path is treated as the core of the manifestation of space and time within this framework. Therefore, each public boundary has its own space-time path which represents its historical lifespan.

The space-time path is deemed to represent the lifespan of each public boundary and it comprises all or some of the basic circumstances (Potential Path Spaces) defined within the representation framework as being creation, existence, mutation and demise. *Creation* represents the space-time origin over which the space-time path will evolve towards the future or past from its origin.

In the case of evolving in the future direction, the space-time path describes the historical lifespan of each public boundary in order to allow the prediction of events at some future time in relation to the origin of the space-time path. One example of this case is the general reviews of public boundaries carried out by Parliamentary Boundary Commissions and Local Government Commissions [Coombes *et al.* 1993]. These reviews are needed to investigate new positions or arrangements of public boundaries to ensure a uniform representation of the electoral population for every constituency (Ward, Electoral Division, District, Region and Parliamentary Constituency) in Great Britain.

On the other hand, evolution in the past direction represents the historical lifespan of public boundaries and allows the explanation of the events at some earlier time, as accounting for the public boundaries being the way they are at the creation of the space-time path origin. For example, the Parliamentary Boundary Commissions have carried out three general reviews since 1944. If the creation circumstance is set on the space-time path for occurring in 1994, the space-time path will deal with the task of modelling the boundary changes as investigated by the Commissioners since

1944 (Figure 3.5). Once the creation circumstance is dated, it cannot subsequently be modified since this would cause integrity problems within the framework. The creation circumstance applies to the chosen origin for the space-time path rather than to the entity itself, in this case, a public boundary. More details about the creation of an origin for a space-time path are considered in Section 3.5 when the implementation framework is discussed.

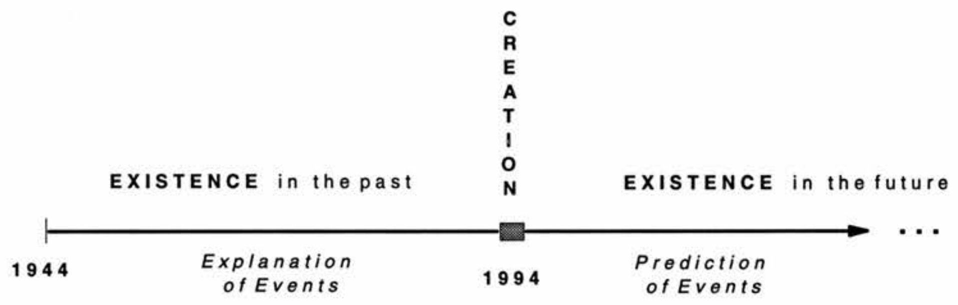


Figure 3.5 - An example of a creation circumstance

Existence encapsulates the space-time path over which the historical lifespan of a public boundary evolves over space and time. The existence of a boundary is constituted by the sequence of changes occurring over its lifespan (Figure 3.5). These changes result from the effects of human activity. They are the *evolutionary states* involved in the evolution in definition of public boundaries. For example, a draft state is assigned to a public boundary which has not yet been confirmed by an Act of Parliament. Once this Act is promulgated, the boundary modifies its draft state to a different state named as new. The evolution framework has been designed to incorporate the existence circumstance into the time-geographic framework, and as such, it is described in the next section.

Mutation can occur at any point on the space-time path of a public boundary (Figure 3.6). In this case, the mutation circumstance represents the occurrence of changes due to an alteration, modification, or transformation of public boundaries over time. One of the main types of such changes is related to the update procedure for relocation of public boundaries. This update procedure produces changes which are the *revolutionary states* occurring in the evolution in public boundary positions. The mutation framework described in Section 3.4 addresses the mutation circumstance in more detail.

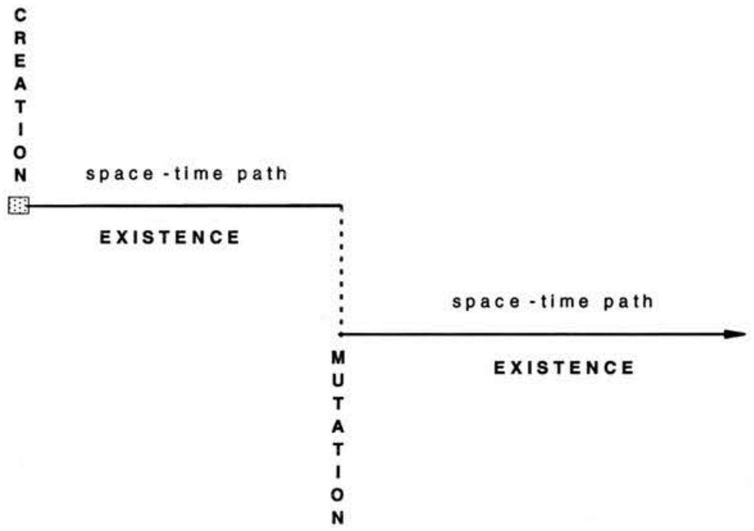


Figure 3.6 - Existence and mutation circumstances on a space-time path

Demise characterises the closure of a space-time path (Figure 3.7). It can occur on the space-time path at any time during the lifespan of a public boundary. This occurs when a public boundary is no longer operative or effective.

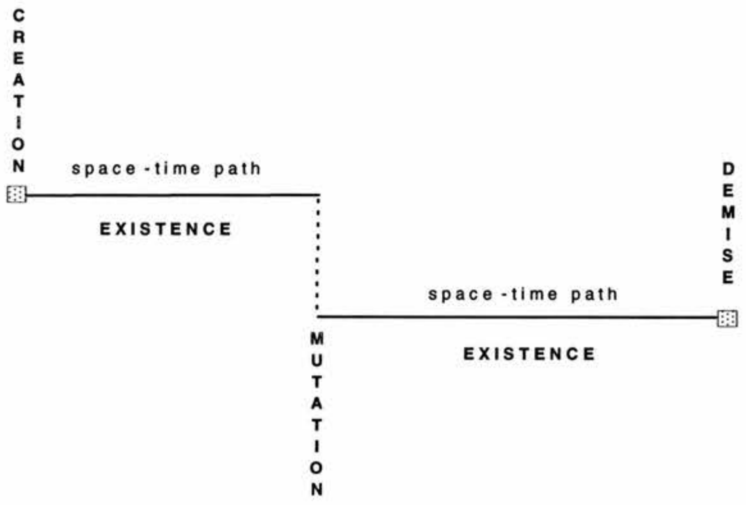


Figure 3.7 - An example of a demise circumstance

3.2.2 The Connectivity and Continuity Aspects of the Space-Time Path

At the next level, in order to be able to define the connectivity and continuity aspects of a space-time path for each public boundary, both *event* and *state* are defined as primary elements pertaining to a space-time path. The event element represents the human activity related to the decision of the location of a public boundary. The state element represents the evolutionary or revolutionary state of a public boundary at any particular time.

The origin of a space-time path can be associated with an event element or a state element. For instance, the creation of a space-time path can occur by a human action or by the explicit choice of a starting point of interest in a boundary state at any particular time. The existence will be a sequence of states and events in which mutations can occur over its states. A space-time path will be an arrangement of these elements which have to be connected to each other effectively to represent the evolution of a public boundary (Figure 3.8). Exploring the temporality and dynamic interactions of events and states may provide an in-depth understanding of our perceptions of reality. Therefore, the existence circumstance is discussed in more detail in Section 3.3.

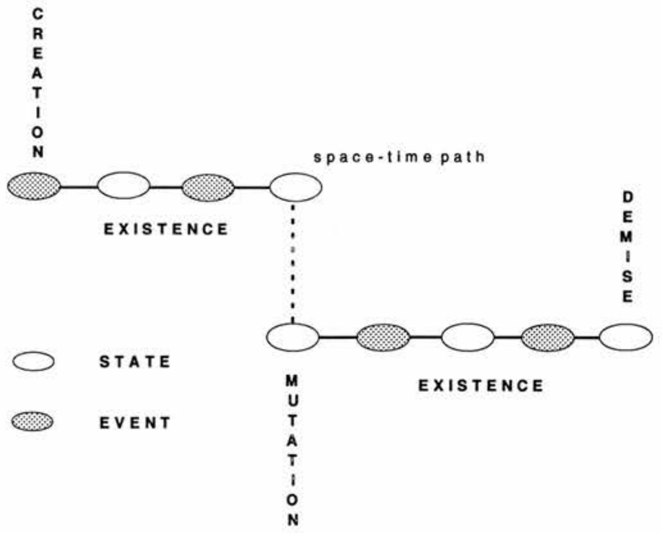


Figure 3.8 - A complete space-time path representation using states and events

The observer's view is a very important factor in determining the object representation. Depending on where the observer is on the space-time path this will determine his present, and consequently, his past and future. With the movement of the observer along the space-time path, his perception and understanding of what is past, present and future is changed. The actual notion of present, past and future in the representation framework relies on the location of the observer on the space-time path, and not on the space-time path itself. The historical view of the observer is reduced to the perception of nearness of states and events on the space-time paths of a lifespan of a public boundary.

The possibility of generating more than one space-time path for each public boundary demonstrates the complexity which one can achieve in the configuration of this representation framework. Having several space-time paths for the same public boundary requires an evolution framework designed to be incorporated into the time-geographic framework. Such a framework can provide a description of the most common states and events which have happened during the lifespans of public boundaries. Therefore, it is discussed in more detail in the following section.

3.3 The Evolution Framework :

The Evolution in Definition of Public Boundaries

Two prevailing lines of thought have been perceived by philosophers and scientists in understanding historical processes based on different perceptions of time. The minority describe a historical process as being nothing more than a disordered collection of random occurrences. In this case, the assumption is that history is an unexplained set of occurrences.

In comparison, the majority of philosophers and scientists assert that there is meaning, purpose or pattern in historical processes. Hegel interpreted a historical process as a process of change caused by action and reaction as well as the synthesis of both. History is not deemed to be a uniform series of transitions but a progress line through which obstacles are overcome. A similar view has been presented by Karl Marx. In his conceptions, history has a direction which is subjected to laws just as nature is.

Attempts to devise historical processes involved in political boundary evolution have been successful in identifying a set of reliable procedures by which human actions can be connected with the evolution of the majority of public boundaries. Lapradelle [1928] identified the three main historical processes which most political boundaries go through as being preparation, decision and execution.

> "The process of preparation precedes true delimitation. The problem of the boundary's location is debated first at the political level then at the technical level. The question is, in general, of determining, without complete territorial debate, the principal alignment which the boundary will follow ... The decision involves the description of the boundary or delimitation ... The execution consists of marking on the ground the boundary which has been described and adopted, an operation which carries the name demarcation." [Lapradelle 1928, p.73].

In adopting these first delineations for historical processes in political boundary evolution, Jones [1945] extended them to allocation, delimitation, demarcation and administration. The administration process would deal with the maintenance of the physical features which have been allocated to be a public boundary. More recently, Prescott pointed out the existence of three main categories in political boundary evolution as being the following:

- evolution in definition: in which the historical processes suggested by Lapradelle and Jones are proposed as boundary-making processes;

"...allocation refers to the political decision on the distribution of territory; delimitation involves the selection of a specific boundary site; demarcation concerns the marking of the boundary on the ground; and administration relates to the provisions for supervising the maintenance of the boundary." [1987, p. 69];

- evolution in position: that means "...how long the boundary has occupied particular sites." [1987, p. 77];

- evolution in the state functions applied at the boundary: in other words, "...the effectiveness with which the boundary marks the limits of sovereignty." [1987, p.80].

Such analysis about the evolutionary aspects of political boundaries represents the study of human activities that have been relevant to the location of a particular boundary. Consequently, the evolution of definition and the evolution of position categories have been utilised in the formulation and construction of our time-geographic framework. This has led to the design of two framing categories which have been designated as the evolution framework and the mutation framework. Both frameworks have been incorporated into the overall time-geographic framework.

The evolution framework is concerned with the historical processes involved in the evolution in definition of public boundaries. It explores the historical events and evolutionary states associated with the space-time path of public boundaries. The mutation framework deals with the scale of change during the evolution in position of public boundaries. This scale of change represents the different revolutionary states which can take place over the space-time path of public boundaries.

3.3.1 The Evolutionary States of the Evolution Framework

Within the time-geographic framework, the evolution framework is characterised by its need to handle systematically the changes associated with the definition of public boundaries. In this case, the changes are related to different *states* acquired by public boundaries after passing through their historical processes (allocation, delimitation, demarcation and administration).

Rackham identified from Booth [1980] the different evolutionary states which most public boundaries can go through as being drawn from the following list:

- "*draft*: ...proposed but not yet confirmed by an Act or Order;
- *proposed works:* ...referred to in an Order related to a physical feature which has not yet been constructed (e.g. new road);
- *new*: ...made in an Act or Order but not yet mered;
- *disputed*: ...mered but not certified because of some disagreement;
- *old*: ...ascertained ... on the ground, certified by the relevant authorities and therefore fixed in alignment ...;
- *obsolete*: ... old boundary no longer used to demarcate administrative areas (obsolete boundary may revert to old boundary if it is reused at a later date to demarcate an administrative area)." [1987, p.32-33].

By having delineated the principal evolutionary states in the evolution in definition of public boundaries, the evolution framework plays an important role in incorporating such states within a space-time path. This is made possible by connecting historical processes and evolutionary states through the occurrence of a sequence of events and states over a space-time path using the creation, existence and demise circumstances (Figure 3.9).

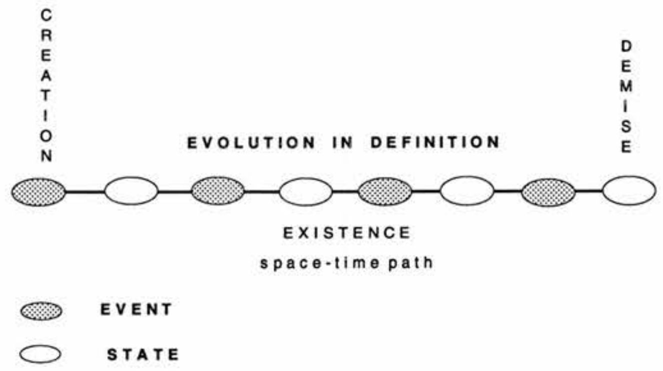


Figure 3.9 - Evolution in definition

3.3.2 Evolution in Definition: The Creation Circumstance

At the creation circumstance, the *allocation* process takes place in selecting a ground feature to be a future public boundary. As a result of this action, the evolutionary state is the *draft* one for a public boundary as illustrated on Figure 3.10. Once a ground feature has been allocated to be part of a public boundary, a spatio-temporal relationship is established between the states and the event involved.

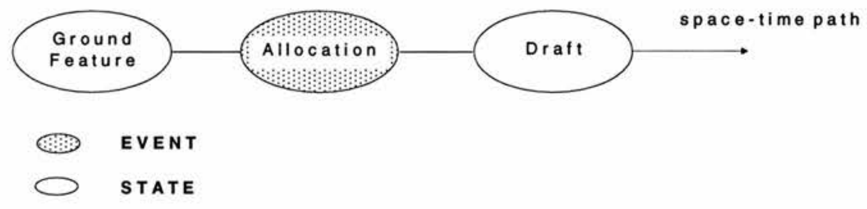


Figure 3.10 - Allocation process representation

There exists a specific spatial relationship between a ground feature and a draft boundary which has to be selected from a set of possibilities depending on the kind of ground feature being utilised. For example, ground features can be paths, ponds, rivers, railways, fences, roads, and hedges. Therefore, some possible spatial relationships would be 'centre of' the road, 'face of' the fence, 'root of' the hedge, '1.00 m from' the railway or '1.83 m from' the path. Draft boundaries are not always necessarily related to ground features since in some areas suitable ground features are not found in the landscape. In this case, a draft boundary is determined by a straight line between two unyielding points on the ground. Otherwise, an engineering work can be carried out to build the necessary ground feature to be a part of the public boundary. All boundary lines representing these spatial relationships are portrayed on Ordnance Survey maps at 1:1,250, 1:2,500 and 1:10,000 scales.

The allocation event is responsible for representing the temporal relationships within the evolution framework. It treats time as a nominal value which indicates the actual date when the allocation took place, for example, on 15th March 1985.

3.3.3 Evolution in Definition: The Existence Circumstance

Following the space-time path into the existence circumstance, the next historical process to occur is the *delimitation* which generates the *new* boundary state (Figure 3.11). This new evolutionary state is the resultant of the action of the delimitation process on the previous draft state.

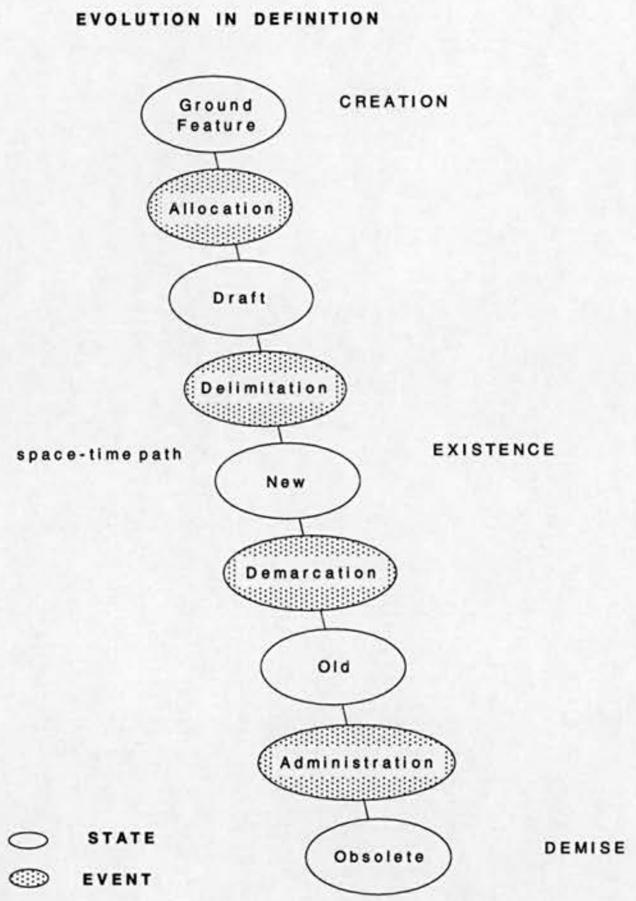


Figure 3.11 - Events and states involved in the evolution in definition

All public boundaries in Great Britain have been delimited by the issuing of an Act of Parliament or an Order of the Boundary Commissions. Extensive archives containing maps at 1:10,000 scale and the Statutory documents, such as Acts and Orders, are held by the Ordnance Survey in order to preserve the legal records of the public boundaries. Thus, the ‘new’ boundary state plays an important role in the evolution framework, which verifies the fact that each public boundary cannot exist without having a ‘new’ boundary state.

The significant characteristic encountered in the spatial relationship between draft and new states is spatial generalisation, which demands procedures for line simplification. The Ordnance Survey utilises different scales for portraying a public boundary having draft and new states. A new boundary is usually portrayed at larger scales than those used for portraying a draft boundary. As a result, some points have to be eliminated from those pertaining to a draft boundary line. However, turning points might have to be preserved as intact points representing the topological junctions, i.e. the line intersections between public boundaries.

The temporal relationship is represented by an interval having as values the operative date and the effective date. An operative date is the actual date of the issuing of a public boundary by an Act or Order, for example, generally 16th of May or the first Thursday in May in Scotland. An effective date assigns the date when an Act or Order has become effectual after the General Election following the operative date [Rackham 1987]. A temporal constraint for each public boundary is essential to guarantee that the date in the allocation process must be prior to both operative and effective dates of the delimitation process.

At this space-time point, a political boundary can be demarcated on the ground, thus, the *demarcation* process occurs and the *old* state is set on the space-time path (Figure 3.11).

All 'old' boundaries are portrayed on basic maps (1:1,250, 1:2,500 and 1:10,000 scales) by boundary lines and by symbols representing their respective demarcation descriptions. The spatial relationship between a new and an old boundary state plays an important role in detecting the occurrence of a controversy between the interpretation of the legal definition of a public boundary and its equivalent geographical position on the landscape. In some cases, this controversy can provoke boundary disputes over the actual location of a public boundary.

This dichotomy can arise for several reasons, such as having more than one interpretation of terms used in the delimitation process, as well as having a contradictory demarcation of the turning points along the boundary line. Generally, the uncertainty of geographical interpretation is more likely to be the culprit.

Considering the temporal relationships, an interval representation is required to date the start and end of the demarcation process for a public boundary. Since most of the disputes concerning the actual location of a public boundary occur during the demarcation process, it is fundamental to have the dates of when the dispute began, as well as the dates on which actions have been taken to rectify the disagreement.

3.3.4 Evolution in Definition: The Demise Circumstance

Finally, demise circumstance can occur and as a result, the *obsolete* state is set, thereby terminating the historical lifespan of a public boundary by an administration process (Figure 3.11).

The following section describes the mutation framework which integrates the evolution in boundary position into the time-geographic framework. This framework represents the changes produced from the update procedures relevant to the evolution in the position of public boundaries. It models the lineage of revolutionary states during the lifespan of a public boundary.

3.4 The Mutation Framework :
The Evolution in the <u>Position</u> of Public Boundaries

The mutation framework addresses the need to provide the support for managing revolutionary states corresponding to changes in the position of public boundaries over time. As the boundary changes its position, a transfer of territory from one authority to another will occur, causing changes in sovereignty and, possibly, changes in the social-economic development of the border landscapes.

Having reviewed most of the available studies on evolution in the position of political boundaries, Prescott suggests the existence of "less severe" effects of changing the position of a political boundary when:

- "the altered boundary has existed for only a short period of time;
- few state functions have been applied at the boundary;
- the groups formerly separated by the boundary have a cultural similarity;
- the economy of the transferred area was formerly oriented across the boundary;
- the economy of the transferred area is of a self-contained subsistence nature." [1987, p.79].

He also affirms that "... the areas transferred by changes in position usually decrease as the definition proceeds from allocation to demarcation." [Prescott 1987, p.77]. Bearing this in mind, the core of the mutation framework is how to connect the immense variety of types of change in the position of boundaries to the configuration of the evolution framework previously described. Hence, the mutation framework is deemed to support the relationship between the evolutionary and revolutionary states of a space-time path. This involves the incorporation of the revolutionary states into the framework, that is to identify the evolutionary states of the space-time path that are likely to be affected by change in their position.

3.4.1 The Revolutionary States of the Mutation Framework

The main reason for creating revolutionary states can be identified as being the need to capture the updates on positions of boundaries. The updates on a position of a boundary can basically occur by three types of change; by natural changes, of which the most common example is the displacement of a boundary position with the displacement of the watercourse by rivers and streams; by man-made alterations, in which updates are due to changes in the position of a boundary by open cast mining, erosion or overthrow of the ground features. And finally, the attachment of new descriptions to an existing boundary can result in an update of the position of a boundary.

New descriptions can occur at any time in the evolution of a public boundary. However, they are more likely to appear during the delimitation process when a boundary line is incorrectly portrayed on the original map in relation to its true position on the ground, and at a much later date, when the position of a boundary has been incorrectly demarcated on the ground.

For the formulation and structuring of the mutation framework, this research will examine how to handle natural changes as well as changes due to a new demarcation description to an existing boundary.

3.4.2 Evolution in Position: The Mutation Circumstance

Natural changes are represented within the mutation framework by considering the change in position of a ground feature which belongs to a public boundary. An update procedure needs to take place in order to create a new position for the ground feature. This involves the creation of a revolutionary state from its precedent evolutionary state. Once the revolutionary state of a ground feature exists within the framework, a space-time path is produced for representing its forthcoming evolution in position (Figure 3.12).

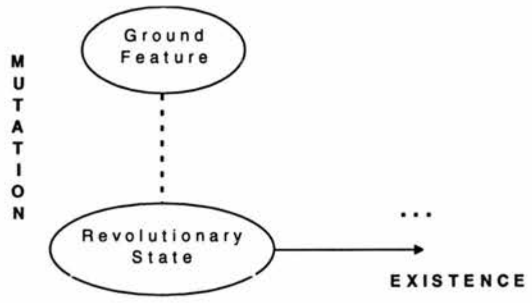


Figure 3.12 - The revolutionary state of a Ground Feature

The evolution in position will be represented by the perambulation process by which the surveyor confirms the displacement of the ground feature on the landscape. This process will affect the previous old boundary state existing in the space-path, in such a way that its revolutionary state will be created in the space-time path (Figure 3.13). The perambulation process plays an important role in the space-time path of the framework since it represents the temporal relationship between the moment when the ground feature is changed on the map and when this change is confirmed in the landscape. Afterwards, the existence of the space-time path follows the historical process and the evolutionary state previously described as being the Administration and Obsolete state.

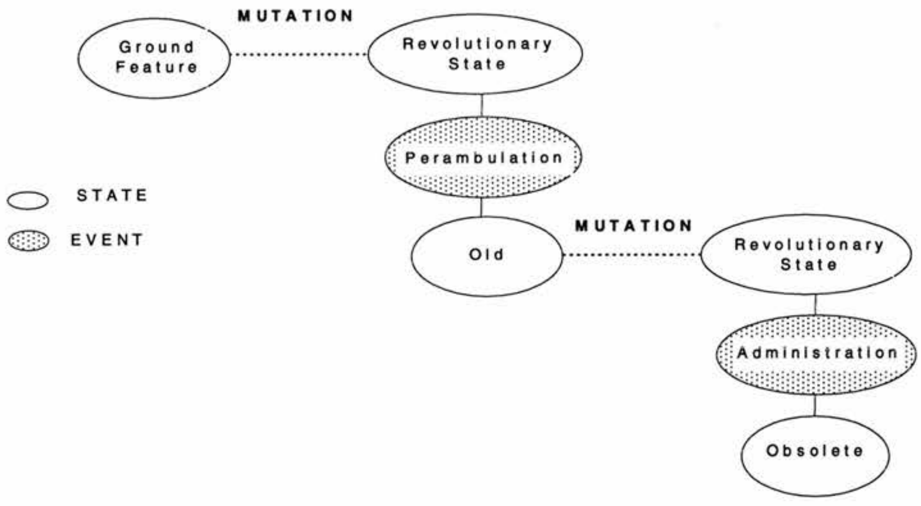


Figure 3.13 - Representation of natural changes

Positional changes caused by new demarcation descriptions have also been chosen for describing the evolution in position of public boundaries. In this case, the revolutionary state is uniquely applied to the old boundary state of a space-time path without producing an additional space-time path within the framework (Figure 3.14). The main reason for not having a space-time path is because the actual change of the position of the boundary line has been performed on the landscape rather than on the map.

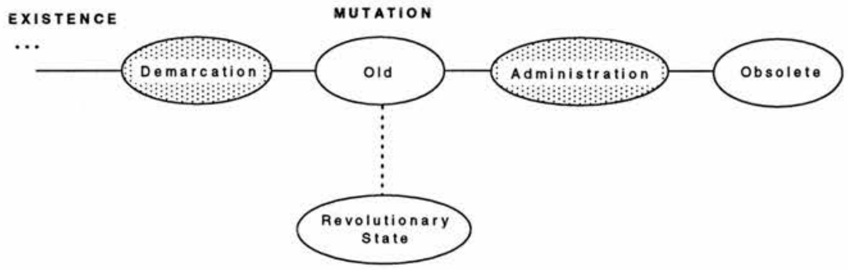


Figure 3.14 - Positional changes due to a new demarcation description

The revolutionary state acts as a depository in charge of portraying on a map where the actual boundary line was when it was wrongly demarcated on the landscape by mistake or misinterpretation of the geographical terms used in the delimitation process. The space-time path representation of the old boundary state is not altered by the existence of this revolutionary state because the necessary update procedures have been carried out on the landscape.

These two examples of evolution in position of public boundaries have been taken to illustrate the variety of changes in a position of a public boundary. However, there is also enormous scope to advance our mutation framework in order to embody the multiplicity of all possibilities of change in the positions of public boundaries.

The next section of this chapter discusses the strategies chosen to incorporate the time-geographic framework into a GIS.

3.5 The Implementation Framework

The Evolution of Public Boundaries within a GIS

Spatio-temporal issues have been explored in different contexts such as in temporal database design, temporal reasoning in Artificial Intelligence, and version management in engineering design applications (CAD, CAM, CASE and VLSI[1]), with the common objective of trying to handle the time element in a way that dynamic and historical information can be managed [Tansel et al. 1993, Chen 1990, Rescher and Urquehart 1971]. The research carried out in incorporating time in GIS benefits from making use of the concepts and achievements attained in these different research domains [Chrisman 1993, Whigham 1993, Beller 1991].

[1] CAD	Computer-Aided Design
CAM	Computer-Aided Manufacturing
CASE	Computer-Aided Software Engineering
VLSI	Very Large Scale Integration circuit

3.5.1 Exploring the Temporal Reasoning Domain

Shoham and Goyal [1988] distinguish four classes characterising the different tasks in temporal reasoning which have been considered in inter-disciplinary research on temporality in GIS. These tasks can be one of the following:

- "prediction: given a description of the world over some period of time, and the set of rules governing change, predict the world at some future time;
- explanation: given a description of the world over some period of time and the rules governing the change, produce a description of the world at some earlier time that accounts for the world being the way it is at a later time;
- planning: given a description of some desired state of the world over some period of time, and the rules governing change, produce a sequence of actions that would result in a world fitting that description;
- learning new rules: given a description of the world at different times, produce the rules governing change which account for the observed regularities in the world." [Shoham and Goyal 1988, p.420]

All these tasks are binding to the GIS applications dealing with dynamic and historical information. In particular, the explanation task is envisaged for the spatio-temporal data model once the time-geographic framework is incorporated into a GIS. The aim is to build a GIS-based model which will be able to explain the evolution in definition and the evolution in position of public boundaries at some earlier time. Chapter 6 illustrates in detail how the spatio-temporal data model incorporates such a reasoning task.

3.5.2 The Object-Oriented Paradigm

Another closely related interdisciplinary research domain in incorporating time in GIS is the research being carried out on object-oriented methods. The fundamental concepts encountered in the object-oriented approach such as object-oriented analysis and design [Booch 1994, Coad and Yourdon 1991a] offer useful improvements in functionality, clarity of data modelling and the potential for simplification of forthcoming application developments in temporal GIS systems.

> "A common difficulty in ... (GIS) application areas is the gulf between the richness of the knowledge structures in the application domains and the relative simplicity of the data model in which these structures can be expressed and manipulated. Object-oriented models have the facilities to express more readily the knowledge structure of the original application." [Worboys *et al.* 1990, p.370].

Therefore, the object-oriented paradigm is viewed as an approach capable of handling the spatio-temporal semantics of the representation framework and the structure of both evolution and mutation frameworks. The time-geographic framework will be embedded within a spatio-temporal model through the modelling capabilities provided by object-orientation. The wide ranging potential of the object-oriented approach and the implications of using it for GIS are discussed in the next chapter.

3.5.3 The Temporal Management Mechanism

Besides incorporating the elements of the time-geographic framework, there is also a requirement for a temporal management mechanism for the organisation of the various states which coexist and interrelate over time and space. Snodgrass [1990] reached the conclusion that object-orientation includes significant support for versioning despite the scarcity of research work on temporal investigations in object-oriented modelling. This research employs the version management approach proposed by Ahmed and Navathe [1991] in which the definition, representation and the management of versions conform with object-oriented concepts. This is further discussed in Chapter 6.

3.5.4 The Smallworld GIS

Integrating a spatio-temporal model based on the time-geographic framework with an existing GIS is fraught with a whole assortment of problems essentially related to the data model structure of the GIS system and its software configuration. The Smallworld GIS has been adopted for the implementation of the GIS-based evolution model of public boundaries. An object-oriented programming language is offered by the Smallworld Magik tool. This provides the constructs for implementing the elements of the representation, evolution and mutation categories of the time-geographic framework. Chapter 7 discusses in detail the integration aspects involved in incorporating the time-geographic framework within the Smallworld GIS.

3.6 Summary
Overview of the Time-Geographic Framework

In this chapter, the concepts involved in the time-geographic framework for framing the public boundary evolution in time and space have been discussed. Table 3.1 summarises these concepts related to each framing category. Framing elements and their main significance within the time-geographic framework are emphasised.

Table 3.1 - Main concepts utilised in the time-geographic framework

Framing Category	*Framing Elements*	*Main Significance*
Representation Framework	1. space-time path	1. fusion of space and time dimensions
	2. Potential Path Spaces	2. circumstances (creation, mutation, existence, demise)
	3. events	3. human activity over a space-time path
	4. states	4. changes
Evolution Framework	5. evolutionary states	5. changes produced by the connectivity and continuity of a sequence of events - evolution in definition
Mutation Framework	6. revolutionary states	6. changes produced by update procedures - evolution in position
Implementation Framework	7. concepts of object-oriented design and analysis	7. spatio-temporal data model
	8. definition, representation and management of states	8. version management

Chapter 4

The Temporal Data Management Component of The Spatio-Temporal Data Model

4.1 Introduction

A vast number of research studies have been carried out in temporal data management in order to address the complex and subtle aspects of space and time domains. This research is engaged in a particular aspect of temporal data management: how to manage changes in the spatio-temporal data model.

Shoham and Goyal [1988] argue that the passage of time is important because changes are only possible with time. In this research, change is significant because it is an element of the time-geographic framework which is responsible for the existence of evolutionary and revolutionary states on a space-time path. Change is also involved in the incremental modification mechanism of the spatio-temporal data model in a way that it is responsible for carrying actions which would create space-time paths within the system. A temporal data management mechanism is the fundamental importance for managing the coexistence of evolutionary and revolutionary states on space-time paths which are generated within the spatio-temporal data model.

In this chapter, a historical background to temporal data management is provided to illustrate the diverse efforts involved in investigating change. Three domains have been selected to illustrate such efforts: object-oriented methods, temporal databases, and version management approaches. First, a historical background on object-orientation summarises the chronological developments from object-oriented programming languages to object-oriented design methods, and finally, to object-oriented analysis methods. The decision as to which of the object-oriented methods to apply in the spatio-temporal data model is shown to be a complex task.

Second, the temporal database research is reviewed on the basis of so many different contexts being researched in the Geographic Information Systems and the Artificial Intelligence fields. Many of the concepts and techniques developed in such fields have been helpful in identifying types of changes which can be involved in the spatio-temporal data model. Identifying change was fundamental to establish the appropriate temporal data management support for the spatio-temporal data model.

Finally, version management approaches are described in this chapter with emphasis on approaches for ordering and updating versions within a model. Deciding on a version management approach to be employed on the spatio-temporal data model involves the finding of an integration solution between the approach and the model. In the spatio-temporal data model, versions are deemed to be distinct from snapshot series because they represent changes that belong to the space-time path of a public boundary.

4.2 Temporal Data Management Historical Background

"As far I have been to ascertain, the first academic treatment of time in database was the 1956 Harvard dissertation by Frederick Brooks, Jr., where the three dimensional view of an historical database was proposed..." [Snodgrass 1990, p.83].

Interest in the temporal component of data management has expanded into areas such as object-oriented methods, temporal databases, and geographic information systems [Soo 1991, Snodgrass *et al.* 1993]. Yet it is since the 1970's that major research projects on temporal aspects have proliferated.

4.2.1 Spatio-Temporal Aspects in the Object-Oriented Paradigm

"Object-oriented methods cover methods for design and methods for analysis. Sometimes there is an overlap, and it is really an idealization to say that they are completely separate activities." [Graham 1994, p.194].

The history of object-oriented methodology starts in the early sixties with the efforts of Dahl, Myrhaug and Nygaard, in creating and implementing new concepts for programming discrete simulation applications. By 1965, an object-oriented programming language Simula [Dahl and Nygaard 1966] had been developed on the basis of the ALGOL-60 language, which was specifically oriented towards discrete event simulation. Later in 1967, the programming language Simula-67 [Dahl *et al.* 1968] was developed by the same Norwegian team, once again an extension of ALGOL-60.

It is with Simula-67 that the basic concepts which characterise existing object-oriented programming languages were first introduced. In particular, Simula-67 introduced the notion of an object class defined by its type and the algorithms necessary to its manipulation. In addition, it also introduced the inheritance mechanisms through which an object class could inherit the data and the algorithms from other object classes.

However, it was only after the mid-seventies that the concepts introduced by Simula-67 were widely recognised. The programming language Smalltalk, a result of the work accomplished by Kay, Goldberg, Ingals and others at Xerox Research Centre at Palo Alto (PARC), has become established as the purest representation of object-oriented concepts. In Smalltalk, everything is perceived as an object and objects communicate to each other by passing messages. Having its origins in Simula and in the doctoral research work of Alan Kay, Smalltalk has evolved by integrating the notion of classes and inheritance from Simula as well as the functional abstractions flavour of LISP[1].

As a result, five releases of Smalltalk have been developed as being Smalltalk-72, -74, -76, -78, and the most recent, Smalltalk-80, in which the last two numbers indicate the year when each release was launched. Smalltalk-V and Smalltalk-AT have also been created as dialects from the former Smalltalk developments [Krasner 1981]. Generally, Smalltalk is a complete programming environment, having features such as editors, a class hierarchy, browsers and many of the features of a 4GL [Graham 1994].

> "Next to Simula, Smalltalk is perhaps the most important object-oriented programming language, because its concepts have influenced not only the design of almost every subsequent object-oriented programming language, but also the look and feel of graphic user interfaces such as the Macintosh user interface and Motif ..." [Booch 1994, p.474].

Several object-oriented or object-based programming languages have been developed, most of them having their conceptual foundations based upon Smalltalk. These attempts have tried to overcome the main inefficiency problems of Smalltalk

[1] LISP stands for LISt Processing which has been originally developed by John McCarthy in 1958 and more recently it has been utilised in Artificial Intelligence developments.

(for example, the lack of support for persistent objects and unfeasibility of having a distributed multi-user environment) but with the pitfall of compromising the purity and consistency of Smalltalk's features. Over 100 object-oriented and object-based programming languages have been developed in the past decade. Some of them are illustrated in Figure 4.15.

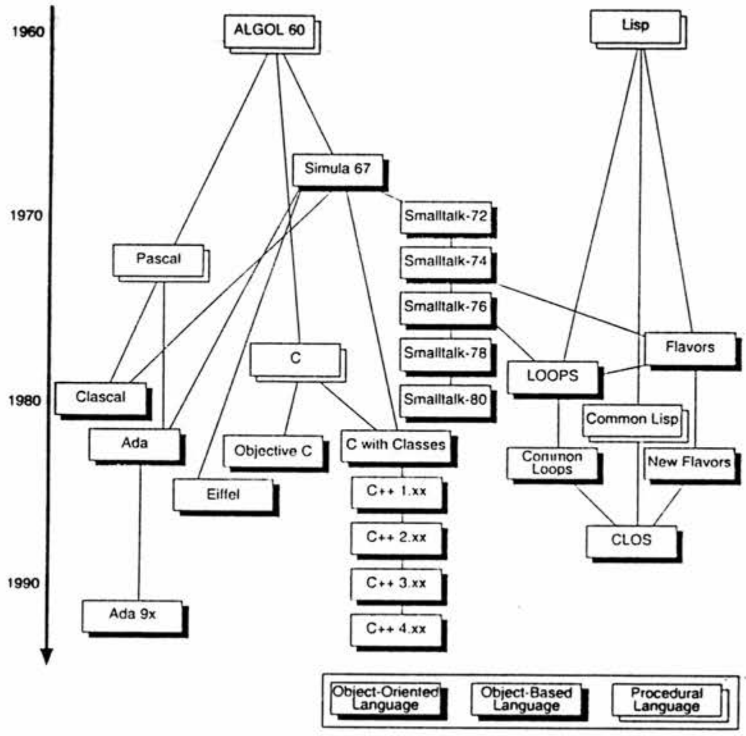


Figure 4.15 - Genealogy of object-based and object-oriented programming languages (Source: Booch 1994, p.474)

However, as Stroustrup points out, "... one language is not necessarily better than another because it has a feature the other does not - there are many examples to the contrary. The important issue is not how many features a language has, but that the features it does have are sufficient to support the desired programming styles in the desired application areas." [1988, p.11]. Object-oriented programming languages are still being developed and it is expected that new languages will emerge, acquiring new features rapidly.

From the mid-seventies and later, the object-oriented paradigm has also influenced research and development in Artificial Intelligence (AI) programming environments. Several object-oriented extensions of LISP, which has been one of the main programming languages utilised in AI Systems, have been created. LOOPS, Common LOOPS, FLAVOURS, KEE, ART and New FLAVOURS are some examples in which a semantically ample form of inheritance is proposed that differs from the one encountered in most object-oriented programming languages such as Smalltalk (Figure 4.18). In this case, values, in particular default values, can be inherited as well as attribute names. Graham assures his readers that, from his point of view, "... AI people have got it right and that this kind of inheritance will gradually penetrate the world of object-oriented programming." [1994, p.78].

With the maturing of the concepts in object-oriented programming languages and their practical use in different application contexts, research interests have diversified, focusing on object-oriented design methods.

> "Object-oriented design is a method of design encompassing the process of object-oriented decomposition and a notation for depicting both logical (class and object structure) and physical (module and process architecture) as well as static and dynamic models of the system under design." [Booch 1994, p.39].

Significant debate has occurred in this research area, mainly concerning whether an object-oriented design method can be, per se, independent of any programming language or whether current design methods are clearly attached to specific object-oriented programming languages. Most object-oriented design methods reveal the influence of Booch's pioneering work [Booch 1986]. In his original proposal, Booch suggested a design method based upon some features of the ADA programming language, using an object-oriented style. GOOD[2] and HOOD[3] are examples of ADA-derived methods which enforce the top-down hierarchical

[2] **G**eneral **O**bject-**O**riented **D**esign Method developed at NASA.
[3] **H**ierarchical **O**bject-**O**riented **D**esign Method developed at the European Space Agency.

decomposition approach among objects but without supporting inheritance and polymorphism.

Also having the influence of Booch's work, OOSD[4] provides a hybrid, low-level notation for logical design of object-oriented methods in general. Although it has been designed to be language independent, OOSD has not been extended to a consistent object-oriented notation due to its inability to deal with complex data structures and large numbers of methods. OODLE[5] is another example of a language independent notation which advocates four interrelated diagrams in order to support the Shlaer/Mellor approach to object-oriented design.

Booch's revised design method [Booch 1991, Booch 1994] gives probably the most incisive and comprehensive prospect of an object-oriented design method. His method improves the concepts of object-orientation and their respective notations as a whole, overlapping with the concepts of object-oriented analysis.

Other research areas have emerged from the synergy between object-oriented programming and database management systems. This has generated a potential mechanism for representing, storing, organising, sharing and recovering objects that include multiple, complex data types and associated methods and functions. Object-Oriented Database Systems (OODBS) have developed capabilities such as persistence, long transactions and versioning, unlike most traditional relational database management systems. Through combining database functionalities with object-oriented programming, OODBS has become an expressive device for multimedia applications, client/server systems as well as GIS, CAD, Engineering and Manufacturing Systems.

[4] **O**bject-**O**riented **S**tructure **D**esign introduced by Wassesman, Pircher and Muller (1990).

[5] **O**bject-**O**riented **D**esign **L**anguag**E** is a design-specific component of Shlaer/Mellor Method (1988).

Object-oriented databases have recently emerged as commercial products. ONTOS[6], O2[7], GemStone[8], ObjectStore[9], and ORION[10] are some examples of object-oriented databases, although their capabilities can differ widely. These object-oriented databases have in common basic characteristics such as methods associated with objects, inheritance of attributes and procedures from supertypes (superclasses), the ability to define the type (class) of objects, their attribute types and relationships. However, they differ substantially in their query languages. The enormous differences probably result from the fact that OODBS have been elaborated based upon programming languages for their data models. Sometimes declarative query languages are only introduced after the initial implementation. The lack of a standard or a formal background for object-oriented query languages has caused differences in query language syntax, completeness, SQL compatibility and treatment of encapsulation [Cattell 1991].

> "The most recent geographic information systems, such as Smallworld, have opted for an object-oriented approach to storing mapping data. The authors of Smallworld chose to create their own persistent version of Smalltalk and object-oriented database because no commercial OODBS existed at the time they started. Vendors starting now have a much better choice." [Graham 1994, p.117].

Object-oriented databases also offer the possibility of storing and manipulating all data pertaining to a GIS application in the same manner. By contrast, in relational databases, spatial data cannot be so readily stored and its integration with other systems is cumbersome. Chance *et al.* [1990] advocate the benefits of object-oriented concepts in developing a seamless environment. In the case of Smallworld GIS, object-oriented database capabilities have been

6 ONTOS is a product of Ontologic of Billerica, Massachusetts which enhances C++ with persistent objects

7 A commercial product of GIP Altair, in Le Chesnay, France. It reveals strong Prolog influences.

8 A product of Servio Corporation of Alameda, California and Beaverton, Oregon. It has been built on top of an extension of Smalltalk-80 denominated OPAL.

9 A product of Object Design of Burlington, Massachusetts, based on C++ programming language.

10 A commercial product of Itasca Systems of Minneapolis, Minnesota, which extends LISP with object-oriented capabilities.

implemented by front ending a version-managed tabular data store with an object-oriented language named Magik. In this environment, system programming, applications development, system integration and customisation are all written using the same object-oriented programming language, i.e. Magik. "Object-orientation does not just mean that there is a database with objects in it, but that the system is organised around the concept of objects which have behaviour (methods)." [Chance *et al.* 1990, p.181].

The question arises as to whether existing relational database products will be superseded or whether they will evolve into some sort of extensions to include object-oriented concepts such as methods, object identity, complex objects and object versions. This has raised a number of issues concerning the relative efficiency of declarative relational query languages and the unfeasibility of storing processing logic at the table level. POSTGRES[11] and Starburst[12] are representative examples of the most advanced implementations of extended relational databases aiming at the main object-oriented features.

> "There is no single 'extended relational approach', in the sense of DBMSs built on the relational model with a common query language and model; indeed, there may be less standardization and consistency than in some other ODMS (Object Data Management Systems). However, the extended relational approach is popular because ... (it) can benefit from much of what has been learned about relational systems, and, more important, it may be possible to migrate users from existing relational database products to ... (extended relational databases) ..." [Cattell 1991, p.83].

Following the proliferation of research on object-oriented programming and database management systems, object-oriented analysis methods have been gradually developed as an approach to improving our understanding of the concepts, activities, rules and assertions of the object-orientation paradigm. "Object-oriented analysis is a

[11] POSTGRES from the University of California at Berkeley is an extension of INGRES with objects, multiple inheritance, versions, historical data, and a powerful extended relational query language (INGRES QUEL).

[12] Developed at the IBM Almaden Research Centre, it extends the relational algebra.

method of analysis that examines requirements from the perspective of the classes and objects found in the vocabulary of the problem domain." [Booch 1994, p.39].

Within the object-orientation paradigm, methods developed so far for object-oriented design are frequently applicable to object-oriented analysis, and vice-versa. The decision about which method of analysis and design should be utilised is laborious. Object-oriented analysis methods have been introduced into several distinct structures and representations, with over 50 published suggestions. "They range from the complex and difficult notations of OMT, Ptech and Shlaer/Mellor to the simpler ones of CRC and Coad/Yourdon, from an emphasis on process to an emphasis on representation and from language dependence to the giddiest heights of abstraction. ... None of these methods is complete in the sense that all issues of the software development life cycle are addressed or that every conceivable system can be easily described." [Graham 1994, p.287].

Computer-Aided Software Engineering (CASE) has become increasingly important as a graphical tool for supporting object-oriented analysis and design methods. CASE tools have been variously regarded with enthusiasm or with disbelief that there is any advantage to be gained through their use.

An increasing number of software products for CASE tools are under development based on the composition of graphical symbols and notations depicting the semantics and features from the object-oriented analysis and design methods. The most important benefits of using CASE tools are their ability to generate code automatically and enhance productivity. However, CASE tools can restrict innovative kinds of application, where the rules and methods provided by CASE tools are inappropriate or even, non-existent.

The main examples of CASE tool systems available in several platforms and operating systems are the ROSE tool supporting Booch's method; Object Maker, with support for a vast range of conventional and object-oriented methods including Booch, Coad/Yourdon, Shlaer/Mellor, Rumbaugh and HOOD; OOATool supporting the Coad/Yourdon method.

The benefits of object-orientation in modelling geographical data within a GIS have been widely recognised as a powerful tool for capturing far more of the meaning of the real-world in a problem domain of an application [Rojas-Vega and Kemp 1994, Milne *et al.* 1993, Worboys *et al.* 1990]. Object-orientation enhances the level of abstraction in a way close to our perception of the real-world, offering the facility for expressing our understanding of the original application.

The current stage of the history of the object-oriented paradigm is characterised by an awareness of the issues related to the necessity of developing open systems and standards. Recently, the Object Management Group (OMG) formed by several representatives from object-oriented industries, has undertaken the task of reaching an industry-wide consensus for a reference architecture and data model for object-oriented database management systems (OODBMS). As being an organisation in charge of creating and promoting a standard for OODBMS, OMG has proposed the Object Database Standard ODMG-93 1.0 which specifies an ODM-Object Data Model, ODL-Object Definition Language, OQL-Object Query Language as well as C++ and Smalltalk language bindings for OODBMS. Conforming to the ODMG-93 standard, an object-oriented database might supply tools for implementing ODM, ODL and OQL features.

The ANSI SPARC OODB Task Group has also been working on a reference object-oriented data model [X3/SPARC/DBSSG/OODBTG 1991, ODMG 1994, Kim 1994]. Although the ANSI Standards Committee has not yet recognised the ODMG-

93 as an official standard, some efforts have been accomplished so far in defining some references for object information systems (ANSI X3H7) and managed objects (ANSI X3T5.4).

With the maturing understanding of object-oriented ideas, this investigation proposes to focus on how the same object-oriented ideas can now be applied to capture the semantics of space-time paths of the time-geographic framework. First, object-oriented analysis and design methods can be viewed as change modelling tools which uphold the meaning of time by incorporating it as part of the semantics of a system. This is due to object-orientation being a key tool for managing updates on the data and the schema, in order to keep track of the changes occurring at the application-level and the system-level.

Second, the effectiveness of object-oriented analysis and design methods in modelling the circumstances (Potential Path Spaces) of a space-time path is fostered by the instantiation of object classes, their states and behaviour. These object-oriented concepts are viewed as the mechanisms to support the temporal semantics and the structure of both evolution and mutation framing categories of the time-geographic framework. An object exists in time and space, whereas a class represents the essence of this object. By having an unique identifier through time and space, an object is efficiently retrieved at any time in its lifespan. The behaviour of an object provides the support for dynamically handling its evolution over space and time.

Ramachandran [1992] elucidates the distinction in version configuration between the system-level and the application-level. At the system-level, version management is supported by object versions which reflect the history of modifications over objects as perceived by the system. At the application-level, version management supports the representation of time or the sequential development of information as defined by the user/application. This level is not yet supported by object-oriented mechanisms in current database systems and it is further discussed in the following section.

4.2.2 Temporal Database Research and GIS

In the temporal research domain, the Legol 2.0 System [Jones *et al.* 1979, Jones and Mason 1980] was the first experiment which incorporated time into a relational model in such a way that time was captured by two mandatory temporal attributes, denominated as start and end attributes, which appeared in each tuple of the relation. It embodied a limited temporal join in extending the meaning of the WHILE operator to identify overlapping periods of time. The update procedures in the system were performed in a non-destructive manner, in which changes were stored in new tuples and were incorporated into the database through the relational union.

Among the major contributions of this prototype, was the motivation for conceptual investigations into how the temporal dimension should be incorporated as an extension of the relational model. One such investigation was the HRDM - the Historical Relational Database Model proposed by Clifford [1982, 1983]. His proposal was that change is best incorporated as a component of the database at the attribute level, rather than at the tuple or relational level (Figure 4.16). "Associating the time-stamp with each attribute provides the user with more control over the semantics of the data, and more flexibility in the kind of queries that can be posed, while at the same time providing the DBMS with greater flexibility in both storage and query evaluation strategies." [Clifford and Ariav 1986, p.173].

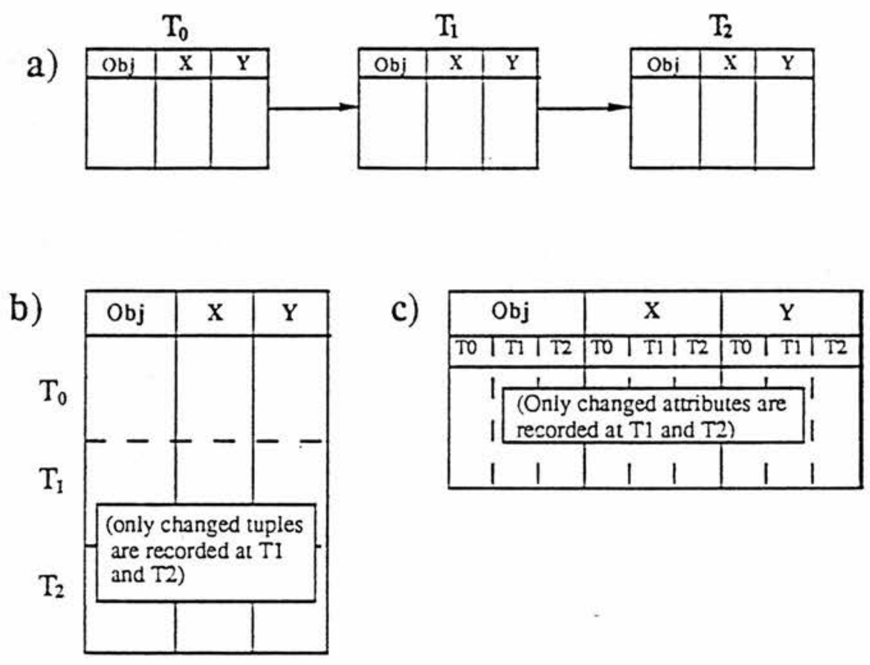

a)
T0
T1
T2
Obj
X
Y
b)
(only changed tuples are recorded at T1 and T2)
c)
T0 T1 T2
(Only changed attributes are recorded at T1 and T2)

Figure 4.16 - Change as a component of the database at the (a) relation level, (b) tuple level, and (c) attribute level (Source: Langran 1992a, p.81)

Langran [1989] has reviewed temporal research within the computer science and information processing domains, evaluating some of the proposed temporal database designs and the problems involved in incorporating them into a GIS. She has concluded that the time dimension should be a component of the database at the attribute level due to its structural organisation which offers the most adequate approach for versioning spatio-temporal data in GIS applications.

In contrast, the Temporal Relational Model [Navathe and Ahmed 1986] introduces a historical relation similar to the one designed in the Legol 2.0 System, but having changes recorded at the tuple level instead. A time-stamp is associated with each tuple using two mandatory time attributes which have been denominated as time-start and time-end. Likewise, the Molecule-Atom Data Model (MAD) [Käfer *et al.* 1990] has also presented the tuple versioning approach as having a time sequence for ordering the tuples. This model has been utilised as a version model for CAD[13] applications used in a VLSI[14] design process.

In a GIS context, Xiao *et al.* [1989] introduced a theoretical model for describing spatial changes in polygon boundaries derived from remotely sensed data. The model incorporates changes at the tuple level in which time is recorded for each pixel change in a tuple using surrogate keys. However, the implementation of such a model has been considered cumbersome, since it creates an uncontrolled volume of spatio-temporal data due to the number of pixels involved. Controlling the volume of data in a GIS is a troublesome technical problem. At the moment, GIS users control the volume of data by selecting an appropriate temporal resolution according to the needs of the application as well as the availability of data. Generally, the volume of data is controlled by decreasing the time resolution.

By the end of the eighties, the need for supporting information varying in time within databases had been recognised [Ackoff 1981, Anderson 1092, Ben-Zvi 1982, Bonczek *et al.* 1981, Clifford and Warren 1983, Snodgrass 1987]. However, certain misconceptions had arisen concerning the terminology and definition of the features supported by temporal databases. It was with the work of Snodgrass and Ahn [1985] that a new taxonomy of time was first presented to unify the concepts developed in the literature.

Transaction time has been proposed as a consensus term for previously defined physical time [Dadam *et al.* 1984, Lum *et al.* 1984], registration time [Ben-Zvi 1982], data-valid-from/to [Mueller and Steinbauer 1983], and start/end time [Reed 1978]. Correspondingly, valid time has been proposed for event time [Copeland and Maier 1984], effective time [Ben-Zvi 1982], logical time [Dadam *et*

[13] Computer-Aided Design
[14] Very Large Scale Integrated Circuit

al. 1984, Lum *et al.* 1984], state [Clifford and Warren 1983], and start/end time [Jones *et al.* 1979, Jones and Mason 1980].

The terms transaction time and valid time have also been proposed to differentiate the distinct types of databases according to their ability to handle temporal information. Four types of databases have been defined according to their ability to support these time concepts and the processing of temporal information [Snodgrass and Ahn 1986]:

- Snapshot Databases which provide a unique snapshot of a database state;
- Rollback Databases which provide a sequence of snapshot states of the database by offering support for transaction time (the time the information has been stored in the database);
- Historical Databases which provide the knowledge about the past by supporting valid time (the time the information has existed in the real-world);
- Temporal Databases which support both transaction time and valid time.

Using the taxonomy of time utilised in the research on database systems, object-oriented databases can be considered as rollback databases due to their ability to represent temporal information. Rollback is the term used for selecting previous database states, therefore a database having this support can be designated a rollback database. Such databases support the history of the transaction results in snapshot states rather than the history of the real-world as we perceive it. Rollback databases "... store all past states, indexed by time, of the snapshot database as it evolves. Such an approach requires a representation of transaction time, the time the information was stored in the database." [Snodgrass and Ahn 1986, p.36]. Rollback databases may eventually advance into the realms of temporal databases.

POSTGRES embodies the first substantive proposal for implementing a rollback database using optical disks with support for transaction management and concurrency control mechanisms [Stonebraker 1987]. In providing these extended database functionalities, the possibility of implementing spatio-temporal constructs as extended database functionality for POSTGRES has also raised expectations in the GIS field. The GEO System [Hoop and Oosterom 1992] is an example of a prototype design in which spatial constructs have been implemented by developing some extensions to the POSTGRES structure. Probably the best-known effort to implement a spatio-temporal data model utilising the POSTGRES database is the Sequoia 2000 Project which has been led by Michael Stonebraker of UC Berkeley [Gardels 1992]. POSTGRES has been integrated with the GRASS[15] GIS to allow temporal manipulation of global change data.

The temporal DBMS prototype developed by the IBM Heidelberg Scientific Centre, in the Advanced Information Management Project, was the first attempt to implement both valid and transaction time with the support of temporal indexing [Dadam *et al.* 1984, Lum *et al.* 1984]. This attempt has been important in consolidating the concepts developed in temporal research in databases, having as a consequence the awareness for the need of spatio-temporal indexing in the GIS field. For example, Langran [1988] pointed out spatio-temporal indexing as a temporal GIS design trade-off, in order to define how to control the volume of spatio-temporal data required in an application within a GIS. Following this concern, Hazelton [Hazelton *et al.* 1990] discusses the possibilities of spatio-temporal indexing for a temporal GIS

[15] Geographic Resources Analysis Support System

as being a natural extension of a quadtree structure, or even, as he proposes, a multidimensional indexing structure which could be the most appropriate.

Snodgrass and Ahn [1985, 1986] have also demonstrated the existence of a true orthogonality between transaction time and valid time which allows each kind of time to be pursued independently. "Time is a universal attribute in most information management applications and deserves special treatment as such." [Snodgrass and Ahn 1986, p.35]. As a result, the concept of a *bitemporal element* was subsequently introduced by Snodgrass [Snodgrass 1992]. This element is deemed to represent the valid-time/transaction-time space (Figure 4.17).

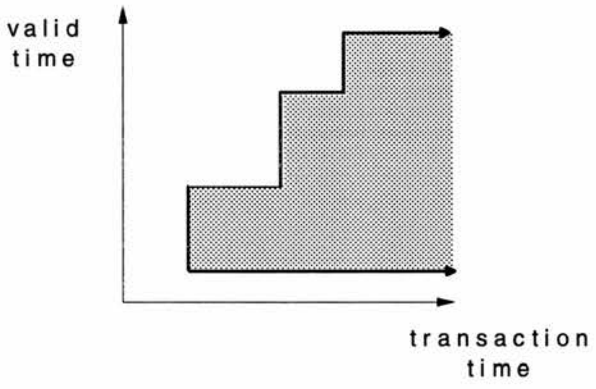


Figure 4.17 - A bitemporal element
(Source: Snodgrass 1992, p. 26)

The spatio-temporal data model proposed by Worboys [1992] was the first investigation employing the concept of a bitemporal element in order to unify the spatial and temporal dimensions of geographical data in a GIS. In his spatio-temporal model, the concept of a bitemporal element "... indicates the extent of an interval event in valid and transaction time. Thus, the T-interval (transaction time interval) gives the insertion and deletion times and the V-interval (valid time interval) describes the interval of temporal existence of the event in the real world." [Worboys 1994, p.510]. Every time a change occurs on an object a bitemporal element is attached to it. The bitemporal element indicates when a change has occurred in the real-world (V-interval) and when the database state has been updated (T-interval).

The link between the object and its bitemporal element is explored using object-oriented concepts.

McKenzie and Snodgrass [1991] provided an evaluation of 12 time-oriented relational algebras against 26 criteria, which had been selected as the specific properties desirable for a temporal extension of the relational database model. Although time is generally perceived as being continuous, the preference for a discrete time model stands out from the evaluation. Time is mostly incorporated into relational databases as discrete subsets of the real numbers ordered linearly. Therefore, changes are supposed to take place in a finite number of times producing a sequence of historical states indexed by time.

By looking for different directions in the temporal research domain, Snodgrass [1990] alludes to the outstanding potential of object-oriented databases which offer significant support for handling time, despite the lack of research work carried out in object-oriented databases about valid time. The important role of object-orientation has also been identified in the proceedings of the first temporal GIS workshop [Barrera *et al.* 1991]. The need for object versions to have an identity is pointed out as a fundamental temporal issue. The conclusions reached in the workshop were that the identity approach supported by object-oriented databases would be better suited for incorporating time in a GIS rather than the value-based approach of the relational database model. The challenge is to identify which are the capabilities provided in object-orientation for the creation, modification and version maintenance of objects.

Kemp and Kowalczyk [1994] presented the first step in incorporating a temporal capability within a GIS using the object-oriented paradigm. They employed object-oriented concepts as the tool for providing a flexible and adaptable temporal capability within a GIS. Johnson and Kemp [1995] have also provided a description of the implementation aspects of temporal capabilities within a GIS such as the use of functions to support spatial and temporal types and operators.

Moreover, significant research work on exploring the dynamic visualisation of data in a GIS context had been initiated by Moellering [1980]. He had proposed a flexible real-time interactive cartographic system having a sequential representation of a changing surface over time, which has been termed a spatio-temporal display. This work is important as much for its motivation of new visualisation techniques as for the specific issue it introduces in visualising changes in GIS. Expectations about dynamic visualisation to be offered by forthcoming GISs have been linked to new technological and methodological developments such as hypermedia coordination [Armenakis 1993], multimedia functions [Fonseca *et al.* 1992], and animated sequences [Gersmehl 1990].

4.2.3 Temporal Reasoning and GIS

In the late eighties, temporal reasoning research from the Artificial Intelligence field brought innovative ideas to temporal data management. Temporal databases are deemed to capture what is known about events and their effects occurring over time. Temporal Reasoning consists of making predictions on the basis of incomplete information [Dean and McDermott 1987].

Temporal databases provide more structure whilst temporal reasoning deals with the modification of information. Allen [1983] asserts that temporal reasoning is useful in domains where the temporal information is imprecise and relative in such a way that dating events is not possible.

Two main approaches have been defined in temporal reasoning. These are the Change-Based Approach and the Time-Based Approach.

> "The **change-based approach** concentrates on the entities that signify a change having taken place; that is, *change-indicators.* Situation calculus in AI, and dynamic logic in theoretical computer science, are prototypes of this approach. *Actions* in situation calculus, and *programs* in dynamic logic, are the basic change indicators." [Shoham and Goyal 1988, p.420].

> "The **time-based approach** recognizes only one fundamental change, the passage of time, which is a constant change unaffected by anything else. There is a time structure (temporal logic), and the assertions are either true or false at various points in the time structure." [Shoham and Goyal 1988, p.425].

The time-based approach has been divided into three main categories which have been proposed by Al-Taha and Barrera [1990] in an attempt to classify the research directions in temporal reasoning. The categories are one of the following:

- Interval-based models: temporality is specified using regular or irregular intervals [Allen 1983];
- Point-based models: temporality is specified using explicit instants of occurrences [Dean and McDermott 1987];
- Mixed-based models: generalised from the previous categories [Shoham and Goyal 1988].

Both approaches have brought up issues concerned with spatio-temporal representation in a GIS context. Reasoning about gradual changes of topological relationships has recently emerged as a requirement in formalising a spatio-temporal representation in a GIS. Egenhofer and Al-Taha [1992] have investigated gradual changes of the location of an object such as translation, scaling, and rotation, by formalising these changes using eight binary topological relationships for two spatial regions. The eight binary topological relations have been depicted in the Closest-

Topological-Relationship-Graph which demonstrates the link among gradual changes in topology. For each gradual change an extensive number of scenarios are possible, for example, the scenario illustrated in Figure 4.18.

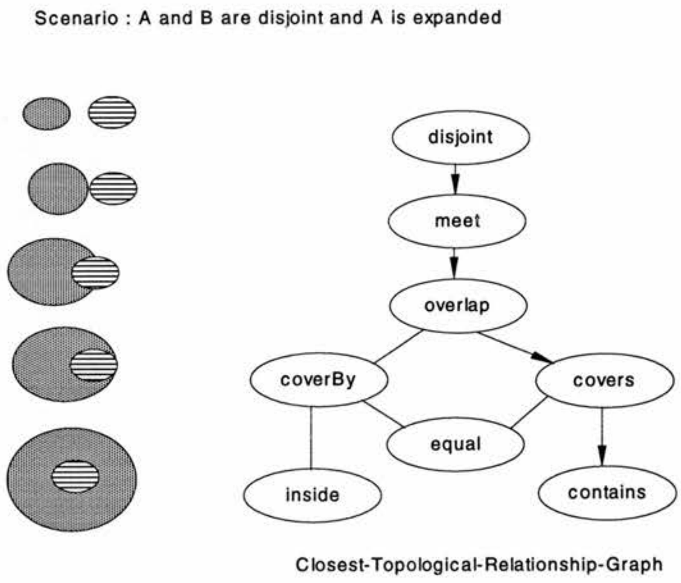

Scenario : A and B are disjoint and A is expanded
disjoint
meet
overlap
coverBy
covers
equal
inside
contains
Closest-Topological-Relationship-Graph

Figure 4.18 - Example of a sequence of gradual topological changes
(Source: Egenhofer and Al-Taha 1992, p.206)

Addressing temporal reasoning in GIS, Frank [1994] suggests an ordinal model of time in which a relative order is expressed between events on a time scale rather than attaching precise dates for events. "Qualitative ordinal information about events is typically encountered in archaeology, urban development, etc. where precise dates for events are not known but the relative order of events can be deduced from observations." [Frank 1994, p.410].

Moreover, TEMPEST, a prototype Temporal Geographic Information System, is an example of implementing the time-based approach in a GIS. "Location in time becomes the primary organizational basis for recording change. The sequence of events through time (Figure 4.19), representing the spatio-temporal manifestation of some process, is noted via a time-line; i.e., a line through the single dimension of time instead of a two-dimensional surface over space. ... Such a time-line, then, represents an ordered progression through time of known changes from some known starting date or moment to some known, later, point in time." [Peuquet and Wentz 1994, p.495].

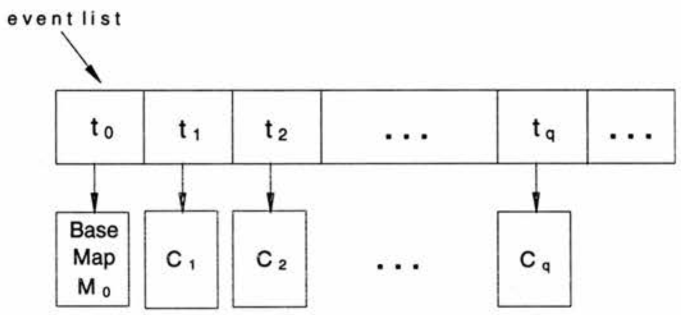


Figure 4.19 - The representation of change organised as a function of time in the TEMPEST prototype (Source Peuquet and Wentz 1994, p.504)

4.2.4 Version Management Approaches

Identifying changes in a consistent configuration is the main issue in defining a version representation within a spatio-temporal data model. Basically, two main strategies are utilised to represent multiple versions of an object. Firstly, version numbers or timestamps may be associated with every attribute or relationship of an object. These attributes or relationships may then be chained, thus having an historical order. The result is a single object and identity (OID) with the versions actually associated with the attributes. This approach could have advantages only in models in which the objects have several attributes as well as changes occurring to only a few attributes.

The second strategy is to track versions at the level of objects rather than attributes. A new version of an object is created by chaining the old object and any older versions to which it has been chained. As a result, a different identifier (OID) is associated with the new version, and thus, each version has its unique identifier within the version configuration.

So far no performance studies have been undertaken on these alternatives. The second strategy has been taken in the spatio-temporal data model because it emerges as a more object-oriented approach to track versions. Considering that the incremental modification approach designed in the spatio-temporal data model is based on an inheritance mechanism of the properties of an object, it would be unfeasible to attach the versions at the attribute level. Versioning needs to be done at the instance level, i.e. at the object level.

Ordering versions within a spatio-temporal data is also an important aspect in temporal data management. Despite the development of storage devices, such as optical disks, offering new capabilities for storing large amounts of data, a significant waste of storage space will still be minimised by ordering versions compactly. The proper choice of a method for ordering temporal data relies on producing the best performance for a given application.

Two general approaches have been developed for aspatial databases [Dadam *et al.* 1984]. The first one is the absolute representation for ordering temporal data in which snapshot series are created for each version. The second is the relative approach in which versions are described in relation to a base state which can be located in the present or past time (Figure 4.20).

The absolute representation is the only one in common use in GIS for storing versions as snapshots. Mainly, it has been used for detecting changes between sequential raster-based images. However, Dadam *et al.* [1984] evaluate both relative and absolute representation approaches in terms of their access time and their integration into update processing.

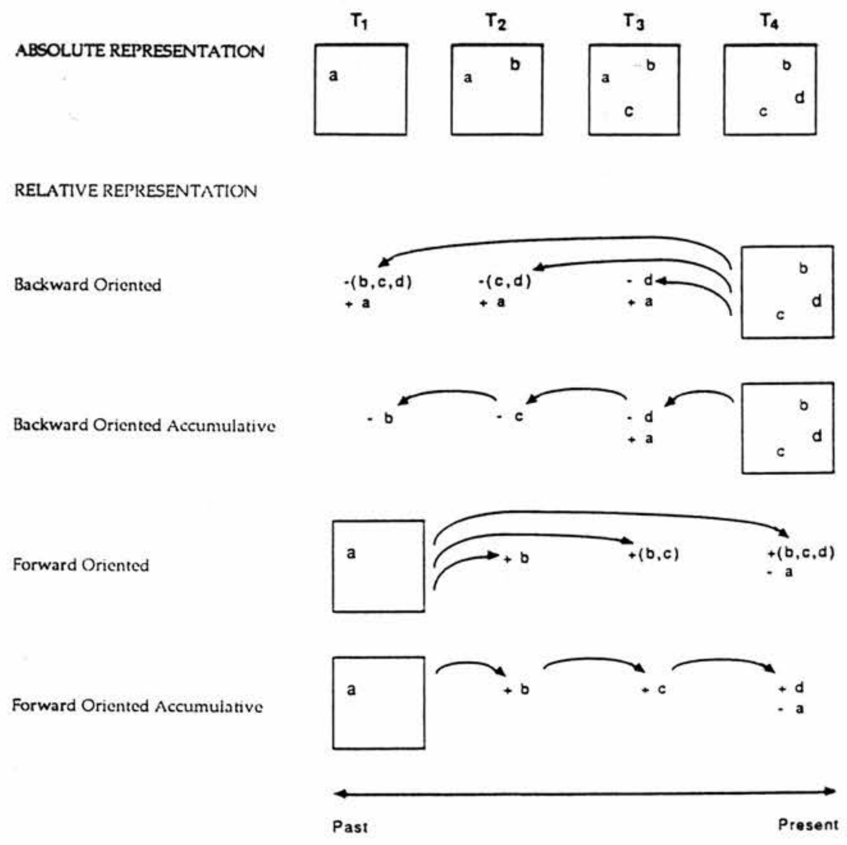


Figure 4.20 - Ordering Time Representations
(Source: Langran 1992a, p.77)

In this evaluation, version numbers have been adopted to address versions, and delta versions to store a version as a delta version instead of a complete version. Therefore, $Cv_x(n)$ is the version number n of the object x, with n_o being the oldest version and CV standing for complete version. The delta version is defined as $\nabla_x(k, k')$ where ∇_x is the delta version between the versions number k and k' of an object x. Some relevant conclusions have been summarised as the following:

- **Absolute Representation**
(based on complete versions)

versions (x) = [$Cv_x(n)$, $Cv_x(n-1)$, ..., $Cv_x(n_o)$]

- storage space requirement is prohibitive.

- **Backward Relative Representation**
  (reconstructs older versions from newer versions)

Backward Oriented
versions (x) = $[Cv_x(n), \nabla_x(n, n\text{-}1), \nabla_x(n, n\text{-}2), \ldots\ldots\ldots., \nabla_x(n, n_o)]$

- it allows a fast access to the current version. Unfortunately, in this versioning strategy, versions have to be recomputed completely whenever a new version is created. This results from the fact that deltas are related to the current version which changes whenever a new version is created.

Backward Oriented Accumulative
versions (x) = $[Cv_x(n), \nabla_x(n, n\text{-}1), \nabla_x(n\text{-}1, n\text{-}2), \ldots\ldots\ldots., \nabla_x(n_{o+1}, n_o)]$

- all versions except the current are expressed as delta to the successor-in-time version;

- access time to versions will increase with their age.

- **Forward Relative Representation**
  (reconstructs newer versions from older versions)

Forward Oriented
versions (x) = $[\nabla_x(n, n_o), \nabla_x(n\text{-}1, n_o), \ldots\ldots\ldots., \nabla_x(n_{o+1}, n_o), Cv_x(n_o)]$

- access time is the same for every version. Each version can be accessed in two steps because all deltas are related to the base version.

Forward Oriented Accumulative
versions (x) = $[\nabla_x(n, n\text{-}1), \nabla_x(n\text{-}1, n\text{-}2), \ldots\ldots\ldots., \nabla_x(n_{o+1}, n_o), Cv_x(n_o)]$

- access time to versions will decrease with their age.

Based on this evaluation by Dadam et al. [1984], the backward oriented accumulative strategy has been adopted in the spatio-temporal data model. The main advantage seen in this strategy is that for each version in the spatio-temporal data model, an accumulative delta is obtained due to a given update. In other words, the delta creation for obtaining the successor-in-time version in the space-time path can be activated by the occurrence of a change within the spatio-temporal data model. Therefore, a successor-in-time version can only be created if an update is triggered within the system. The visualisation of such strategy can be found in Chapter 8 where the results are presented.

4.3 Conclusions

This chapter has shown the significant amount of research in temporal data management which has been carried out on object-orientation, temporal databases, temporal reasoning and version management. Many of the concepts and techniques developed in these areas have been discussed and brought together within the context of GIS. This has played an important role as the basis for designing a temporal data management approach for the spatio-temporal data model. In the following chapter, the spatio-temporal data model is discussed.

Chapter 5

Development of the Spatio-Temporal Data Model Using Booch's Object-Oriented Method

5.1 Introduction

"The act of drawing a diagram does not constitute analysis or design. A diagram simply captures a statement of a system's behaviour (for analysis), or the vision and details of an architecture (for design). If you follow the work of any engineer - software, civil, mechanical, chemical, architectural, or whatever - you will soon realize that the one and only place that a system is conceived is in the mind of the designer. As this design unfolds over time, it is often captured on such high-tech media as white boards, napkins, and the backs of envelopes." [Booch 1994, p.171].

In this chapter, the spatio-temporal data model presented is that based on the object-oriented analysis and design method proposed by Booch [1986, 1991, 1994]. The events and states from the time-geographic framework have both been modelled as classes of objects. However, they play different roles within the spatio-temporal data model. Events as classes of objects are classes for describing what happened, is happening, or will happen during the lifespans of public boundaries. On the other hand, states as classes of objects describing the evolutionary and revolutionary states tell us what has changed, is changing, or will be changed during the lifespans of public boundaries.

The main advantage of this modelling decision is that events can be modelled, exist within the database, and interact with the states of public boundaries without depending on the changes in the states themselves. This has been achieved by utilising the *inheritance* abstraction for developing an incremental mechanism for the space-time path. Changes happen at the instance level in such a way that a public boundary has its space-time path depicted by different instances of different classes of objects representing the events and states, which are connected through a mechanism of incremental modification.

The spatio-temporal data model make use of the object-oriented decomposition in which the concept of Potential Path Spaces developed in the time-geographic framework is transposed to classes of objects interacting in different kinds of scenarios. Four scenarios have been designed for modelling the main abstractions and the required behaviour of the objects pertaining to Potential Path Spaces.

A comprehensive set of diagrams expressing the important aspects of the model has been produced to illustrate the fundamental decisions which have been taken in order to handle the semantics of the time-geographic framework within the spatio-temporal data model. The problem domain is illustrated by a Process Diagram which describes a set of processes for each scenario and their allocation to processors in the physical view of the spatio-temporal data model. Class Diagrams are utilised to describe the relationships among objects as well as the operations and properties associated with them. They are also used to describe the semantic dependency between classes and the ability to walk through the model from one scenario to another. Finally, Interaction Diagrams are used to illustrate the execution of a scenario as well as to visualise the process involved in 'making space-time paths' within the spatio-temporal data model.

ObjectMaker release 2.1 by Mark V Systems Limited has been used as the graphical tool for supporting the proposed diagrams of the spatio-temporal data model. The key for the symbols is found in Appendix A as a convenient reference for interpreting the diagrams which illustrate this chapter.

5.2 The Choice of Booch's Object-Oriented Method

"Is there a 'best' (analysis and) design method? No, there is no absolute answer to this question..." [Booch 1994, p.23].

The decision to apply an object-oriented analysis and design approach to developing the spatio-temporal data model, was motivated by the simplicity as well as the complexity presented in this methodology. Due to its simplicity, the underlying ideas of the time-geographic framework can be described more easily using the primary concepts developed in object-orientation. Also, the complexity of implementing the time-geographic elements of such a model is an interesting challenge from both the conceptual and the implementation points of view.

Some main criteria have been established a priori for choosing a suitable object-oriented method without any particular solution in mind. Such criteria are preconditions that should be provided by the favoured object-oriented method to increase the quality, flexibility and clarity of the system. One of the criteria that inevitably arises as a management issue, is the need to support an extremely rich notation to describe the main concepts of object-orientation which will be employed to model the object representation of the time-geographic framework.

However, such a notation must not be complicated or difficult to employ, to allow an easy learning and understanding of the layers, phases or activities pertaining to the object-oriented method. A good analogy is the Entity Relationship (ER) diagram [Chen 1976] which is the most common approach to relational data modelling due to the clearness and comprehensibility of its graphical notation. This notation can immediately be translated into a database implementation.

Another complementary criterion is the need to represent the dynamics of change and evolution of the objects within the object-oriented methodology. There is a need for handling rules (topological rules), constraints (authority constraints), and methods (trigger methods - update, delete, create) at the levels of both object and class. The graphical notation might show a scenario with a message being passed from one object to another as well as ensuring that the message is in fact part of the protocol of an object.

Finally, an implementation criterion has been also taken into account, namely language independence. Since the implementation will be carried out in Smallworld GIS, the Magik programming language will be used. Therefore, the object-oriented method has to be language independent which means that it can not be bound to a specific object-oriented language such as C++ or Smalltalk. This is of the greatest importance for assuring an overall integration in the system between the stages of analysis, design and programming.

The object-oriented method proposed by Booch [1994] is a major contribution to unifying ideas by incorporating the best from each of the existing object-oriented methods, including the work of Jacobson, Rumbaugh, Coad and Yourdon, Constantine, Shlaer and Mellor, Firesmith, and others. A unified notation has been achieved in which the 'cosmetic differences' between Booch's notation and those of other object-oriented methods have been eliminated: in particular, the notation used by Rumbaugh in the OMT method. Booch asserts that "Rumbaugh's work is particularly interesting, for as he points out, our methods are more similar than they are different." [1994, p.vi].

Four distinct models are defined to produce the analysis and design within an object-oriented development, namely the Logical Model, the Physical Model, the Static Model, and the Dynamic Model. These models are then grouped into two dimensions: the logical/physical view and the static/dynamic view. For each dimension, Booch has defined a number of diagrams which denote a view of the models of the system. The Class Diagram, the Object Diagram, the Module Diagram, and finally, the Process Diagram are used to capture the semantics within the Logical and Physical Models.

In the case of the Static and Dynamic Models, two additional diagrams are proposed: State Transition Diagrams and Interaction Diagrams. Each class may have an associated State Transition Diagram which indicates the event-ordered behaviour of the objects. Similarly, in conjunction with an Object Diagram, an Interaction Diagram can be provided to show the time- or event-ordering of messages. Booch presents a fairly rich notation as well, but one that is easier to understand, and therefore, easier to apply. For example, the notation used in the Interaction Diagram is actually a generalisation of event diagrams of the Dynamic Model of OMT [Rumbaugh *et al.* 1991] combined with the interaction diagrams of Jacobson's method [Jacobson *et al.* 1992].

This method distinguishes two processes in object-oriented development, these being the micro process and the macro process. The micro process is more closely related to the spiral model of Boehm [1986], and serves as the framework for an iterative and incremental approach to development. The macro process is more closely related to the traditional waterfall life cycle, and serves as the controlling framework for the micro process [Booch 1994]. This provide a flexible and legitimate object-oriented model of an application in which analysis and design techniques have been integrated for each process, model, and view of the object-oriented development.

Booch argues that the processes within an object-oriented analysis and design method cannot be described in a 'cookbook'. Basically, his proposal for an object-oriented development embodies purpose, products and activities which are considered as incremental and interactive phases rather than steps. By avoiding presenting his method as a 'cookbook', Booch emphasises processes, models and views within his object-oriented method. This provides a flexible and legitimate object-oriented model of an application in which analysis and design techniques have been integrated for each process, model, and view of the object-oriented development.

Booch introduces two main concepts about objects in his object-oriented method. Firstly, the client/server concept between objects. A client object is an object that uses the operations of another object, either by operating upon it or by referencing its state. Conversely, a server object is the one which provides the operation. Secondly, the existence of state within an object means that the order in which operations are invoked is important. This brings the concept of active and passive objects. Active objects can manifest some state change without being operated upon by another object. They hold their own thread of control. Passive objects, on the other hand, can only undergo a state change when explicitly acted upon. In this manner, the active objects in the system serve as roots of control. If the problem domain involves multiple threads of control, then we will usually have multiple active objects [Booch 1994].

Basically, four kinds of operations can act upon an object [Booch 1994]:

1. Modifier: an operation that changes the state of an object;
2. Selector: an operation that captures the state of an object, without changing the state;

3. Iterator: an operation that allows the access to some properties of a state of an object in a well-defined order;
4. Destructor: an operation that frees the state of an object and/or destroys the object itself.

The definition employed by Booch for behaviour also discerns that the state of an object affects its behaviour. In other words, objects do not have a static state, but each state of an object represents the cumulative results of its behaviour. At any point in time, the state of an object involves all properties of this object (usually static) as well as the current values of these properties (usually dynamic). An object can have spatial properties and non-spatial properties representing its state as illustrated in Table 5.2.

Table 5.2 - Examples of spatial and non-spatial properties of an object

SPATIAL PROPERTIES	*Spatial References*	1. coordinates 2. enclosing rectangles or boxes of minimum size 3. place names or numerical code 4. reference to tiles, blocks of space, either regular or irregular in shape
	Spatial Relationships	1. connectivity 2. orientation 3. adjacency 4. containment
	Spatial Considerations	1. scale 2. resolution 3. units of space (mm, m, km) 4. map projection
	Spatial Dimensions	1. line 2. point 3. area 4. volume 5. grid cell
	Spatial Measurements	1. length, perimeter length 2. surface area 3. volume 4. shape 5. orientation 6. slope
NON-SPATIAL PROPERTIES	1. qualitative and quantitative attribute values assigned to an object 2. descriptive or thematic information about an object	

As the perception of phenomena in the real-world varies from one individual to another, objects can be referred to different classes or the same object can belong to different classes at the same time. Deciding upon the best classification for a given problem domain is a fundamental aspect of object-oriented analysis and design. Burrough asserts that classification "... is essential for human understanding - without classification or generalization our brains become swamped by detail". [1986, p.137].

Booch devotes a chapter to the subject of classification in which three approaches are described: classical categorisation, conceptual clustering and prototype theory. Classical categorisation utilises, as a main criterion in its classification process, the association of common behaviour or properties among objects to categorise the object classes. In contrast, conceptual clustering attempts to explain how knowledge is represented by clustering the conceptual descriptions of classes to which objects may belong. For applications that have neither clearly bounded properties/behaviour nor concepts, prototype theory is considered as an alternative classification approach. In this case, objects are grouped according to their degree of relationship to concrete prototypes. The classical categorisation has been employed for designing the spatio-temporal data model.

5.3 The Problem Domain

Defining the boundaries of the problem domain relies on deciding about the requirements which are relevant to the scope of the spatio-temporal data model. The fundamental requirement is to handle the two conceptual pillars of the time-geographic framework. They are the Space-Time Path and the Potential Path Space.

5.3.1 Designing Space-Time Paths within the Spatio-Temporal Data Model

In investigating the possibilities of designing a space-time path for each public boundary, one of the main findings encountered in the analysis was the need to deal with ***change***. Change is responsible for the existence of evolutionary and revolutionary states on a space-time path. In other words, change is an incremental modification mechanism responsible for carrying out actions which would create space-time paths within the system.

Considering that states and events from the time-geographic framework will be classes of objects within the spatio-temporal data model, the design of an incremental mechanism relies on defining the kind of relationship that should exist among such classes in order to reproduce a space-time path. Precisely, this involves an instance of one class being connected with an instance of another class in such a way that it reproduces the history of a particular public boundary on a space-time path.

Inheritance has been adopted as the incremental modification mechanism for designing space-time paths within the spatio-temporal data model due to its potential for subclassing the evolutionary processes of incremental change of the time-geographic framework. Two types of incremental modification mechanisms can be described in the spatio-temporal data model. They are the Independent Incremental Modification and the Overlapping Incremental Modification.

Basically, an independent incremental modification by inheritance transforms a parent class **P** with a modifier **M** into a resultant class **R** = **P** + **M** [Wegner and Zdonik 1988]. The composition operator + plays an asymmetric role in determining **R** from **P** and **M** (Figure 5.21), since

```
result R
    inherits P;
    modified by M.
```

The parent, modifier, and result classes exhibit a structure with a finite number of properties:

$$P = (p_1, p_2, \dots p_p)$$
$$M = (m_1, m_2, \dots m_m)$$
$$R = (r_1, r_2, \dots r_r)$$

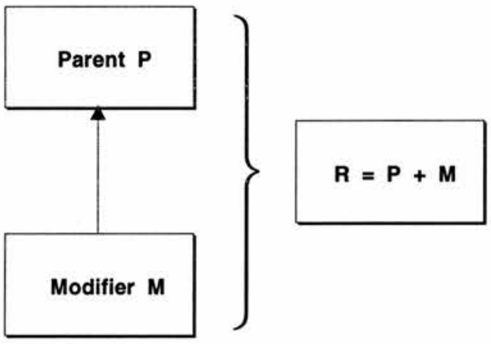


Figure 5.21 - Independent incremental modification by inheritance
(Source: Wegner and Zdonik 1988, p.55)

In the spatio-temporal data model, the independent incremental modification occurs over classes of objects belonging to the creation-existence-demise circumstances of a space-time path. It represents the connection of evolutionary state-event-evolutionary state of a space-time path. In this case, the Parent class is an evolutionary state which is modified by an event (Modifier class) into another evolutionary state (Resultant Class) as illustrated in Figure 5.22.

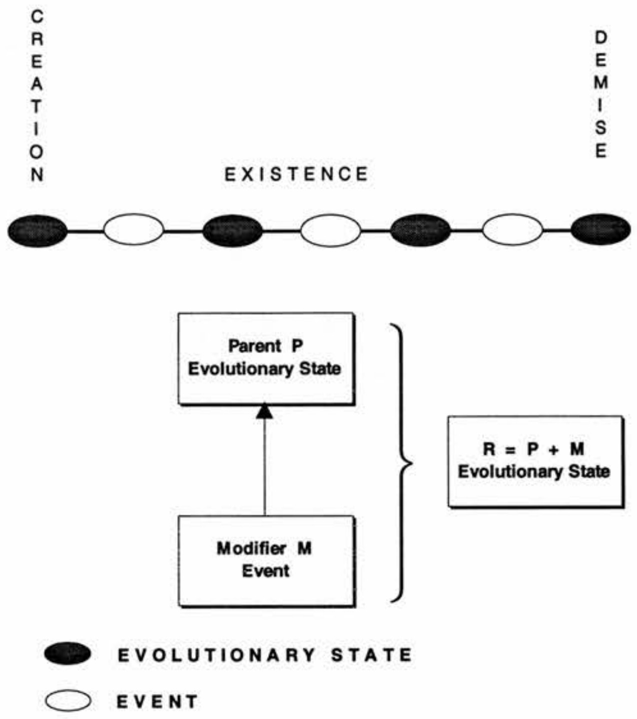


Figure 5.22 - Independent incremental modification associated to a space-time path

The properties of the Resultant class (r_1, r_2, ... r_r) are independent to those properties of the Parent class (p_1, p_2, ... p_p) and the properties of the Modifier class (m_1, m_2, ... m_m). The independent incremental mechanism is responsible for the union of these properties in a way that R has r+p+m properties.

Conversely, an overlapping incremental modification by inheritance transforms a Parent class into a Resultant class without having the presence of a Modifier class. The properties of the Resultant class are inherited from the properties of its Parent class [Wegner and Zdonik 1988]. In the spatio-temporal data model, the overlapping incremental modification occurs over classes of objects involved in the mutation circumstance of a space-time path. In this case, the Resultant class is assigned to an revolutionary state which inherits some of the properties of its Parent class, i.e. an evolutionary state. New properties can be added to the Resultant class whose names do not occur in its corresponding Parent class (Figure 5.23).

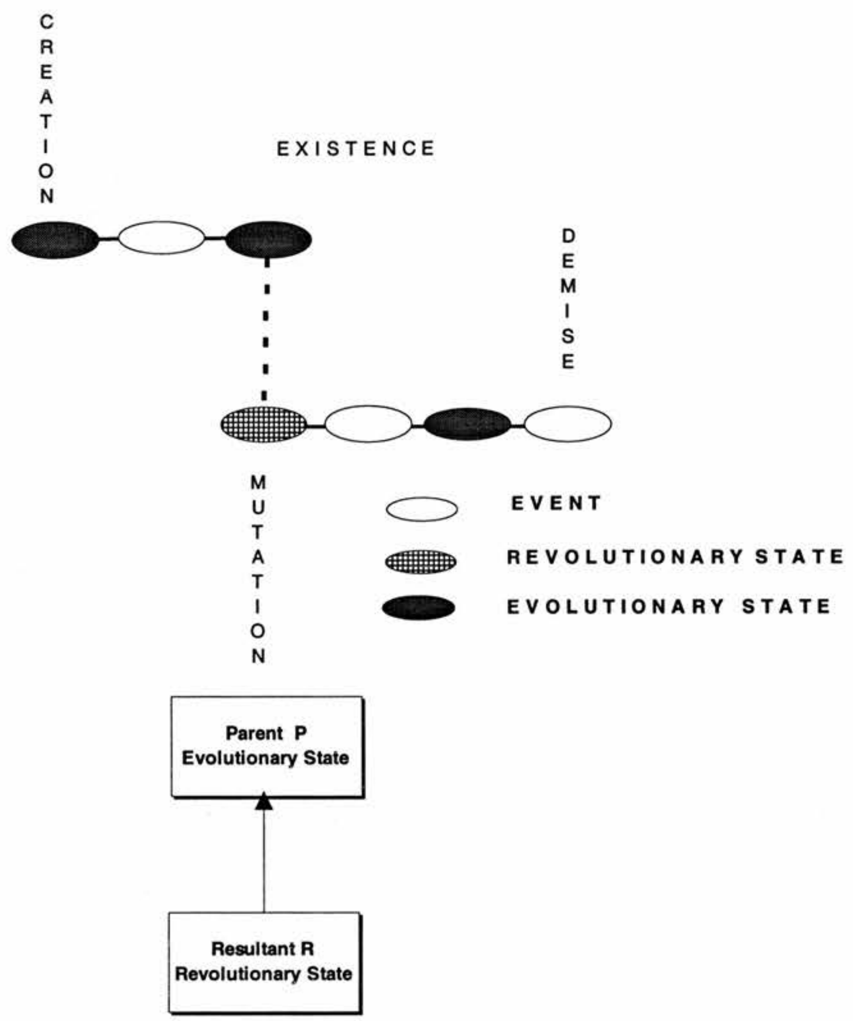


Figure 5.23 - Overlapping incremental modification associated to a space-time path

In conclusion, the space-time path is built by connecting each instance of a class with its corresponding instance of another class according to the incremental modification mechanism involved. This allows us to represent the connection between events and states within the spatio-temporal data model by using the instantiation of classes of objects and the inheritance relationship between these classes. The result is the generation of a new object representation for spatio-temporal data modelling in GIS.

Having such a configuration for space-time paths in the spatio-temporal data model, the need for temporal data management of all instances which can belong to a space-time path is fundamental. This implies the analysis and design of a version management mechanism in order to manage change over time within the spatio-temporal data model. Therefore, the version management mechanism adopted for the spatio-temporal data model is discussed in more detail in Chapter 6.

The next sections of this chapter explore how both incremental modifications have been incorporated into the spatio-temporal data model. A complete analysis and design of the definition of the incremental mechanisms is presented, focusing on the problem of building a resilient architecture.

5.3.2 Designing Potential Path Spaces within the Spatio-Temporal Data Model

A general procedure for deriving or calculating Potential Path Spaces has not been devised within the spatio-temporal data model. Instead of this, Potential Path Spaces are modelled in the form of scenarios which embody the processes previously identified in the time-geographic framework. Four main scenarios have been devised as integrated subsystems within the model for designing Potential Path Spaces.

Fundamentally, the four scenarios are described as follows:

- The *Public Boundary Entry Scenario*: based on the creation circumstance of the time-geographic framework. It is responsible for managing the allocation process in which a ground feature is assigned to be a public boundary;
- The *Evolution Tracking Scenario*: based on the existence circumstance of the time-geographic framework. It is in charge of managing all existing evolutionary states of a public boundary;

- The *Update Scenario*: based on the mutation circumstance of the time-geographical framework. It is responsible for updating public boundaries and managing all existing revolutionary states of a public boundary;

- The *Archiving Scenario*: based on the demise circumstance of the time-geographic framework. It is in charge of storing and retrieving the obsolete public boundaries.

Although these scenarios play an important role in creating the conceptual representation of Potential Path Spaces, they have also been relevant for some design decisions about the implementation of the spatio-temporal data model. Each Potential Path Space (scenario) has been allocated differently in the physical view of the system. The Process Diagram (Figure 5.24) illustrates the different scenarios by allocating the appropriate processes to the processors in the physical view of the spatio-temporal data model. This diagram can also be used to visualise the mapping of the architecture on to its realisation in code.

In addition, the Process Diagram shows the overlap between the analysis and design activities in developing the spatio-temporal model. Each scenario represents a Potential Path Space at the analysis level as well as a client/server architecture of the system at the design level. This suggests a decentralised architecture which implies that most of the processing can be done by the client reducing the bottleneck across the network.

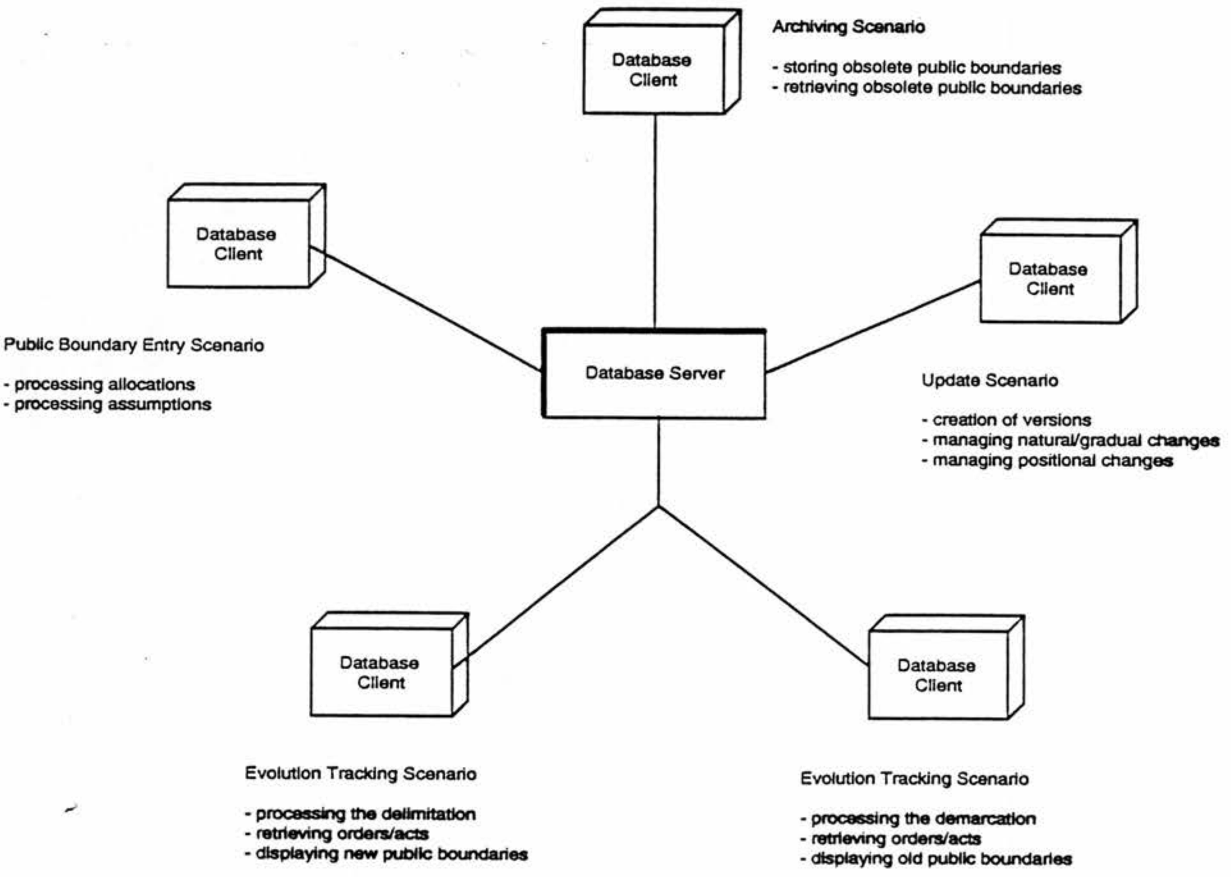

Archiving Scenario
- storing obsolete public boundaries
- retrieving obsolete public boundaries
Database Client
Database Client
Public Boundary Entry Scenario
- processing allocations
- processing assumptions
Database Client
Database Server
Update Scenario
- creation of versions
- managing natural/gradual changes
- managing positional changes
Database Client
Database Client
Evolution Tracking Scenario
- processing the delimitation
- retrieving orders/acts
- displaying new public boundaries
Evolution Tracking Scenario
- processing the demarcation
- retrieving orders/acts
- displaying old public boundaries

Figure 5.24 - The Process Diagram

In order to implement this efficiently, a version management client/server mechanism has to be built in at a fundamental level in the DBMS. The Smallworld Version Managed Data Store (VMDS) utilises a B*-tree standard structure which allows disk blocks to be cached on the client workstation. When an operation is produced by a user of VMDS, it is only seen by that user until a version management operation is carried out to update the database on the server [Newell and Batty 1993]. This approach for version management of database transactions in VMDS is discussed in Chapter 7 where the Smallworld GIS is described in more detail.

5.4 Scenarios

GroundFeature and PublicBoundary have been identified as primary classes within the spatio-temporal data model. Each one of them contains objects which embody some state, exhibit certain behaviour, and are uniquely identifiable. GroundFeature represents every physical feature in the landscape which has been assigned to be a public boundary object. Likewise, the PublicBoundary class denotes the political boundary itself.

The PublicBoundary class involves several evolutionary states which were previously described in Chapter 3 as being Draft, New, Old, and Obsolete. Since each state represents a different level of abstraction for a PublicBoundary, the object-oriented concept of metaclass has been adopted for PublicBoundary. Therefore, PublicBoundary characterises a metaclass whose instances are themselves classes which represent the different evolutionary states of a public boundary (Figure 5.25).

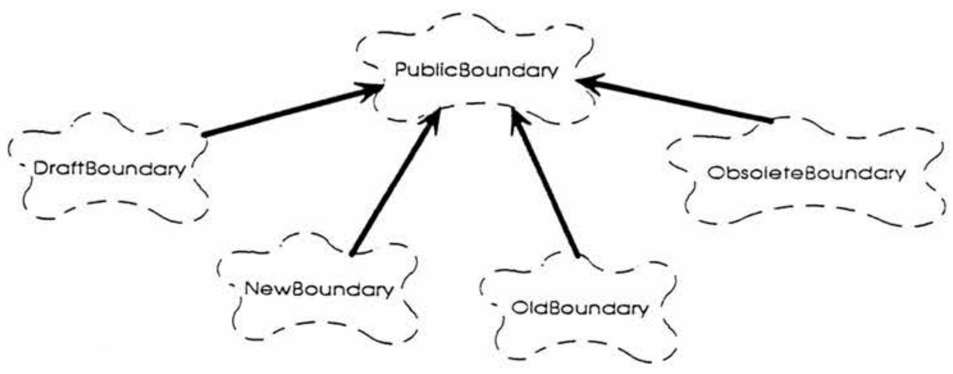


Figure 5.25 - PublicBoundary metaclass diagram

5.4.1 Public Boundary Entry Scenario

The Public Boundary Entry Scenario represents the creation circumstance of the space-time path within the spatio-temporal data model. Therefore, it manages the evolutionary states and the events which are concerned with the allocation process in the spatio-temporal data model. Besides, it involves the creation of the space-time path itself within the model.

Four classes have been designed for the Public Boundary Entry Scenario as being the DraftBoundary Class, the GroundFeature Class, the Assumption Class and the Allocation class. The GroundFeature class represents the ground features on the landscape about which statements might be made. The DraftBoundary class comprises many GroundFeature objects, each of which might have different statements about them.

The statements play an important role in the Public Boundary Entry Scenario. A statement can be related to an assumption which states that a ground feature can be regarded as a possible feature to be a public boundary. This denotes the mapping between an instance of the GroundFeature class and its corresponding instance of the Assumption class.

Once a ground feature has been selected to be a public boundary, this denotes the mapping between an instance of the GroundFeature class and its corresponding instance of the Allocation class. In this case, the mapping is definite and therefore, unchangeable, because it depicts the creation of a space-time path within the system. As a result, an instance of the DraftBoundary class can be created. The Class Diagram in Figure 5.26 illustrates the design decisions regarding these classes of objects (See Appendix C for the entire Class Diagram of the spatio-temporal data model).

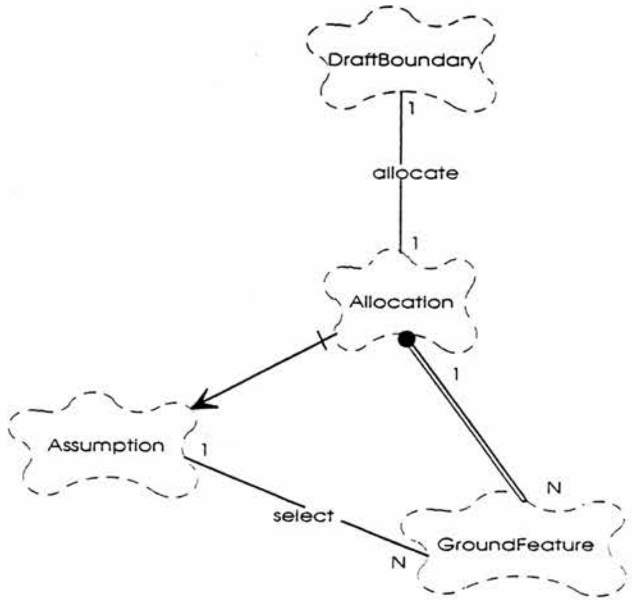


Figure 5.26 - The Class Diagram for the Public Boundary Entry Scenario

An inheritance relationship exists between the Allocation and Assumption classes. The Assumption class represents a superclass having the generalised statements about the allocation process. The Allocation class is a subclass representing a specialisation in which properties and methods from the superclass Assumption are added.

The aggregation relationship is assigned between Allocation and GroundFeature classes. Such abstraction permits different instances of the GroundFeature class to be allocated as a possible public boundary. Allocating ground features to be a public boundary is deemed to happen in such a way that selecting a ground feature to be an instance of the GroundFeature class does not interfere the properties of its corresponding instance of the Allocation class as a whole. In addition, removing an instance of the GroundFeature class does not necessarily delete all of its corresponding instances of the Allocation class.

The allocate and select associations in the Class Diagram denote a semantic dependency and suggest a bi-directional navigation between classes. For example, given an instance of GroundFeature, we should be able to locate the respective object denoting the DraftBoundary, and vice versa. Besides, these associations represent the independent incremental modifications in the spatio-temporal data model.

In this case, GroundFeature is the parent class having Allocation and Assumption as modifiers into a resultant DraftBoundary class. Table 5.3 shows the properties defined for each of the classes involved in the Public Boundary Entry Scenario. Although these classes are interrelated through an incremental mechanism, each one of these classes has its own properties.

This independent incremental mechanism of the spatio-temporal data model is represented by the HistoricalView class in the Public Boundary Entry Scenario. Such a class denotes the union of all properties which belong to the Parent, Modifier and Resultant classes in the scenario. As a design decision, the HistoricalView class has been defined for visualising the space-time path involved in the Public Boundary Entry Scenario. The historical view provides a snapshot of all the properties within the system, which have been involved in the assumption and allocation processes of a specific public boundary selected by the user up to a certain point in time. HistoricalView classes are illustrated in Chapter 8.

Table 5.3 - Properties of DraftBoundary, GroundFeature, Allocation and Assumption Classes

classes	spatial properties	non-spatial properties
DraftBoundary	line length mereing_point*	type *{Draft Creation Not Confirmed, Draft Creation Confirmed, Proposed Works}* description *{base of, centre of, edge of, foot of, face of, root of, side of, top of, track of, 1.00m from, 2.00m from, not applicable}*
GroundFeature	point line area	type *{stream, fence, hedge, kerb, loch, pavement, railway, road, wall, pond, undefined, defaced}*
Allocation	none	map_scale_used_for description date *{year, month, day}*
Assumption	none	statement validated valid *{from, to}*

*mereing points are points portraying the change of a ground feature description (See Appendix B)

During the execution of the Public Boundary Entry Scenario, many statements mapped through the Assumption class will be made about ground features' feasibility of becoming a public boundary. As the scenario moves closer to a final decision, these statements eventually become Allocation objects. The execution of this scenario is illustrated on the Interaction Diagram in Figure 5.27 which provides a global perspective of the various operations involved in the Public Boundary Entry Scenario. This diagram illustrates the behaviour of the system in terms of the interaction between instances of classes.

These are the operations designated for the following classes:

- GroundFeature
    - draftBoundary
- DraftBoundary
    - groundFeature
- Assumption
    - make
    - value
    - mostRecent
    - select
    - become
- Allocation
    - select
    - assign

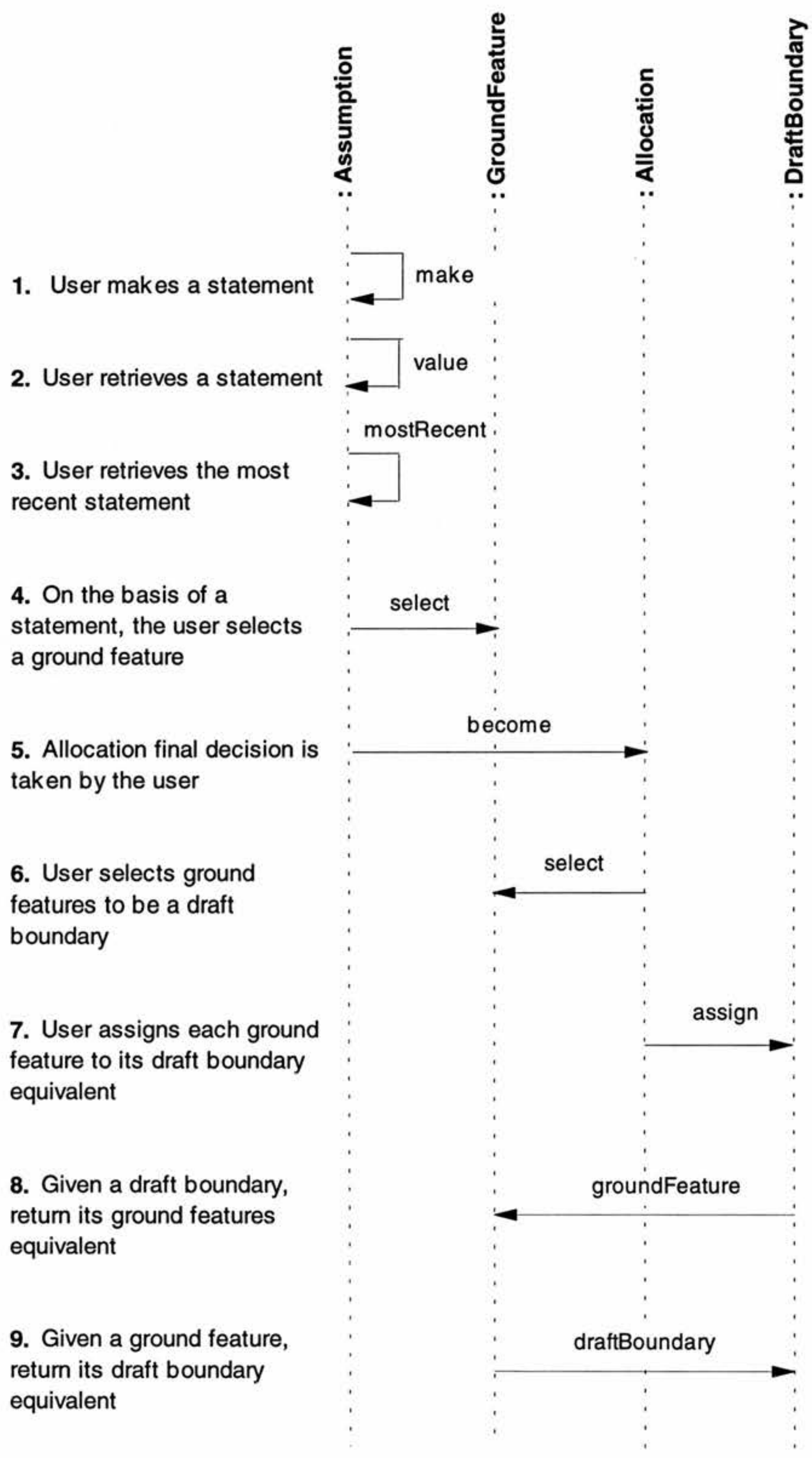

: Assumption
: GroundFeature
: Allocation
: DraftBoundary
1. User makes a statement
make
2. User retrieves a statement
value
mostRecent
3. User retrieves the most recent statement
4. On the basis of a statement, the user selects a ground feature
select
become
5. Allocation final decision is taken by the user
select
6. User selects ground features to be a draft boundary
assign
7. User assigns each ground feature to its draft boundary equivalent
groundFeature
8. Given a draft boundary, return its ground features equivalent
draftBoundary
9. Given a ground feature, return its draft boundary equivalent

Figure 5.27 - The Interaction Diagram for the Public Boundary Entry Scenario

5.4.2 Evolution Tracking Scenario

The main role of the Evolution Tracking Scenario is to assign an order among events rather than precise dates measured over time. Such qualitative modelling, where precise dates for events are not essential, can be deduced from the 'boundary-making' process as described in the time-geographic framework. Therefore, both delimitation and demarcation processes from the time-geographic framework are described in the Evolution Tracking Scenario (See Appendix C for the Class Diagram of the Evolution Tracking Scenario).

In the delimitation process, a draft boundary is confirmed by an Act or Order as a public boundary and therefore, it assumes a new evolutionary state in the time-geographic framework which is called a new boundary state. For the spatio-temporal data model, the focus is on the design decisions regarding the structure and behaviour presented by the instances of the DraftBoundary, Delimitation, and NewBoundary classes.

For the demarcation process, the position of a new boundary is ratified on the ground by surveyors, on the basis of the information provided by the documentation concerning the delimitation. In the spatio-temporal data model, the emphasis is on modelling the controversial aspects between the delimitation and demarcation of a public boundary over time. This involves the NewBoundary, Demarcation, and OldBoundary classes.

The Delimitation and Demarcation classes characterise the delimitation and demarcation events which have been defined in the time-geographic framework. NewBoundary and OldBoundary are classes defined to represent respectively the new and old evolutionary states of a public boundary.

On the basis of the analysis carried out for modelling the delimitation process, a prerequisite arises to carry out a line generalisation of a public boundary. This imposes different scales for portraying the same line which belongs to the DraftBoundary and later on when it has been assigned to be a NewBoundary. For example, consider displaying a specific line of a public boundary having a draft status at 1:2,500 scale. Once this line has been elected to be a new boundary, its display might appear at one unique scale, that is, at 1:10,000 scale. In addition, some previous points belonging to the line in its draft state have to be eliminated for reasons of clarity.

Two possibilities are acceptable for representing visually the different states of a public boundary. Firstly, an external class could be defined in order to query each object in terms of each kind of graphical display to be employed. Conversely, the other possibility is that each object can encapsulate the knowledge of how to display itself. The former possibility has been chosen as the better solution due to its closeness to the object-oriented concepts.

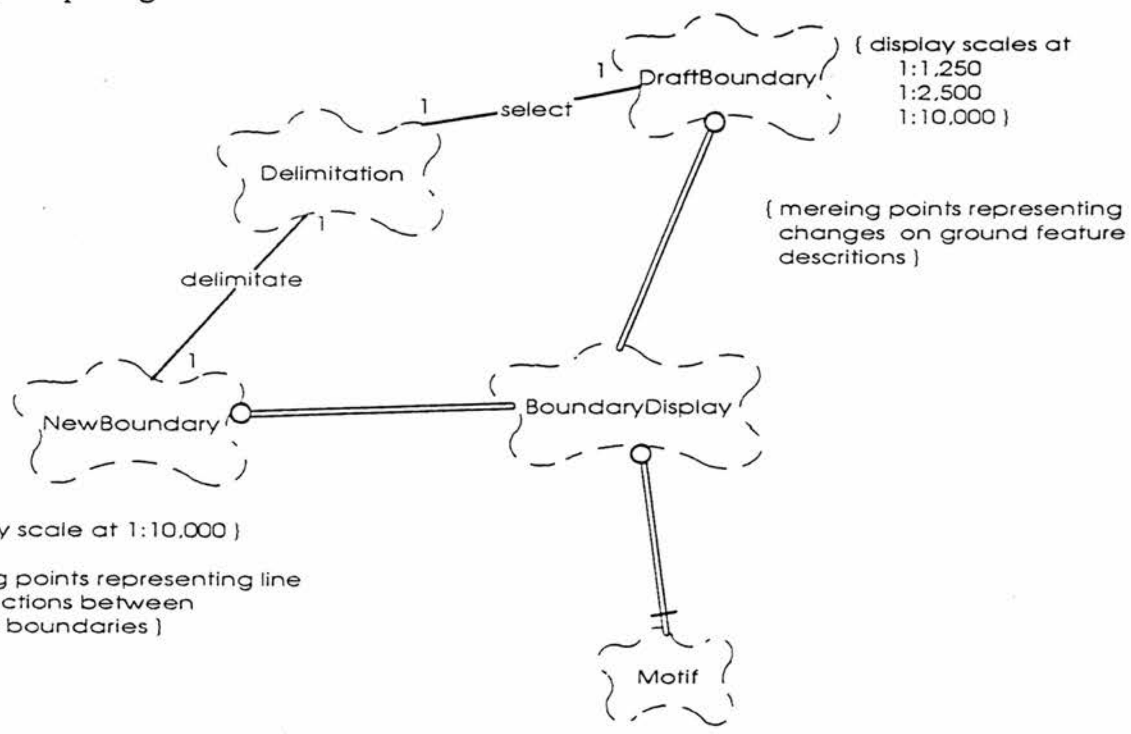
For example, an object will be a line representing the public boundary. Having a draft state, such a line will be displayed at 1:1,250, 1:2,500, or 1:10,000 scale with the following graphical elements (See Appendix B):

- points portraying the change of a ground feature description (mereing points)
- text properties portraying the description of the line, for example, CR (centre of road), CS (centre of stream), or FF (face of fence).

In contrast, by having a new state, the same line will be displayed at 1:10,000 scale with the following graphical elements (See Appendix B):

- turning points portraying the line intersections between public boundaries.

Figure 5.28 illustrates the DraftBoundary and NewBoundary classes sharing a common class BoundaryDisplay through a **using** relationship within the model. Both DraftBoundary and NewBoundary are the clients of the display interface of the supplier BoundaryDisplay. The BoundaryDisplay provides the display routines for the graphical elements that each client requires. In implementation terms, the BoundaryDisplay is built upon a Motif class which deals with the off-the-shelf graphics package of the Smallworld GIS.



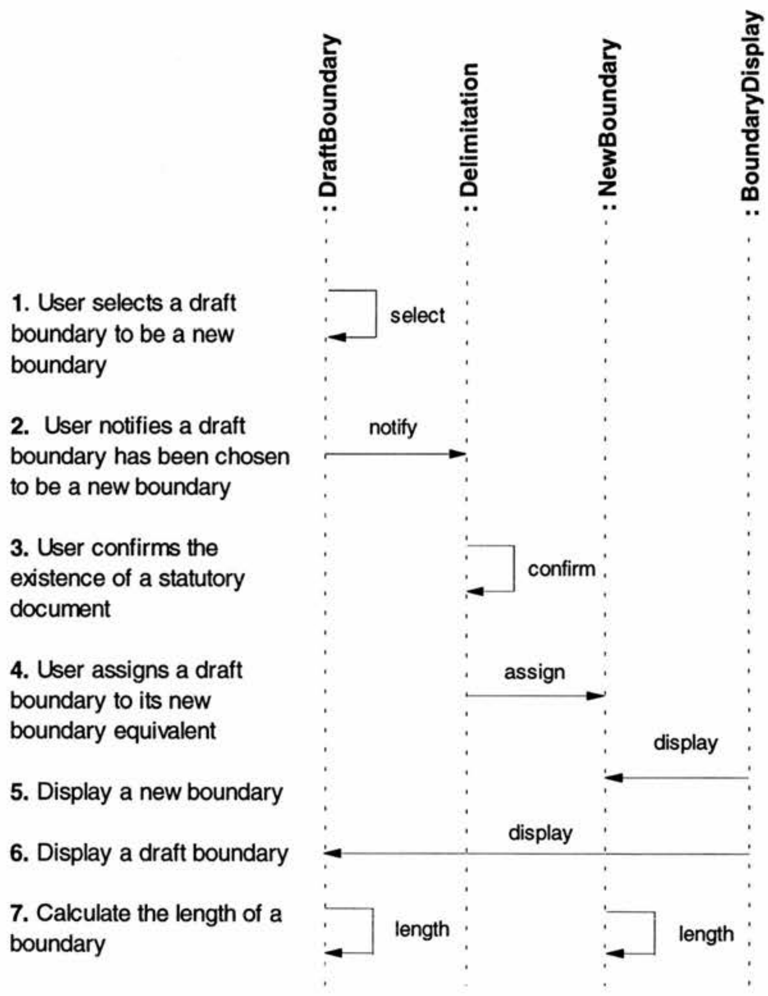
DraftBoundary
{ display scales at 1:1.250 1:2.500 1:10.000 }
1
select
1
Delimitation
1
delimitate
1
{ mereing points representing changes on ground feature descritions }
NewBoundary
BoundaryDisplay
{ display scale at 1:10.000 }
{ turning points representing line intersections between public boundaries }
Motif

Figure 5.28 - The Class Diagram for the Evolution Tracking Scenario - Delimitation Process

The advantage of this design is that it allows a future replacement of the available display software with, for example, a hypermedia coordination which would display the objects in a dynamic manner. This would require the replacement of the display routines in the BoundaryDisplay class without the need to modify the implementation of every displayable object of DraftBoundary and NewBoundary.

The execution of such a scenario is illustrated in Figure 5.29. As previously mentioned, the Interaction Diagram illustrates the behaviour of the system in terms of the interaction between instances of classes. The following operations have been identified for the respective classes given below:

- DraftBoundary
    - select
    - notify
    - length
- Delimitation
    - confirm
    - assign
- NewBoundary
    - length
- BoundaryDisplay
    - display



: DraftBoundary
: Delimitation
: NewBoundary
: BoundaryDisplay
1. User selects a draft boundary to be a new boundary
select
2. User notifies a draft boundary has been chosen to be a new boundary
notify
3. User confirms the existence of a statutory document
confirm
4. User assigns a draft boundary to its new boundary equivalent
assign
display
5. Display a new boundary
display
6. Display a draft boundary
7. Calculate the length of a boundary
length
length

Figure 5.29 - The Interaction Diagram for the Evolution Tracking Scenario - Delimitation Process

In the Evolution Tracking Scenario, the demarcation process involves the NewBoundary class, the Demarcation class, and the OldBoundary class (Figure 5.30). NewBoundary represents the legal location of a public boundary. OldBoundary represents the old status of a public boundary. In other words, it characterises the actual location of a public boundary after the demarcation. The Demarcation class plays an important role within the spatio-temporal data model. It controls the flow of operations between NewBoundary and OldBoundary.

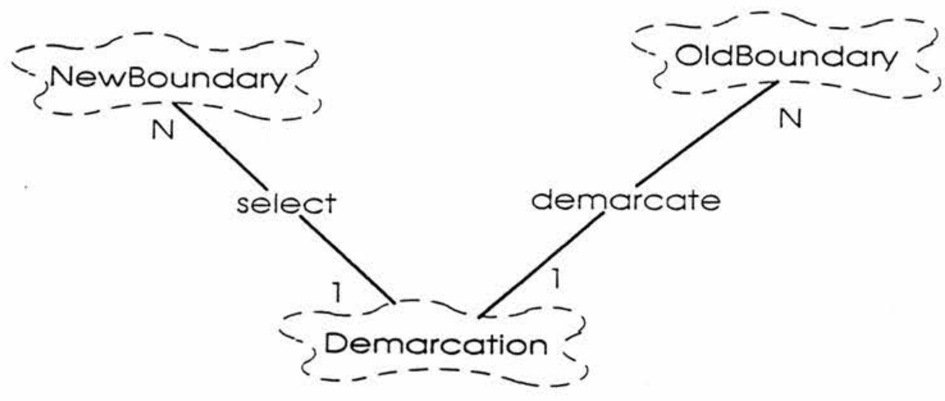


Figure 5.30 - Class Diagram for the Evolution Tracking Scenario - Demarcation Process

The interaction diagram in Figure 5.31 illustrates with a box representing the relative time that the flow of control is focused in a Demarcation object. Each instance of the Demarcation is the ultimate focus of control, and its behaviour of carrying out a demarcation process invokes different methods over the instances of NewBoundary and OldBoundary. This is achieved by defining the following operations for the Demarcation class:

- Demarcation
    - select
    - confirm
    - assign

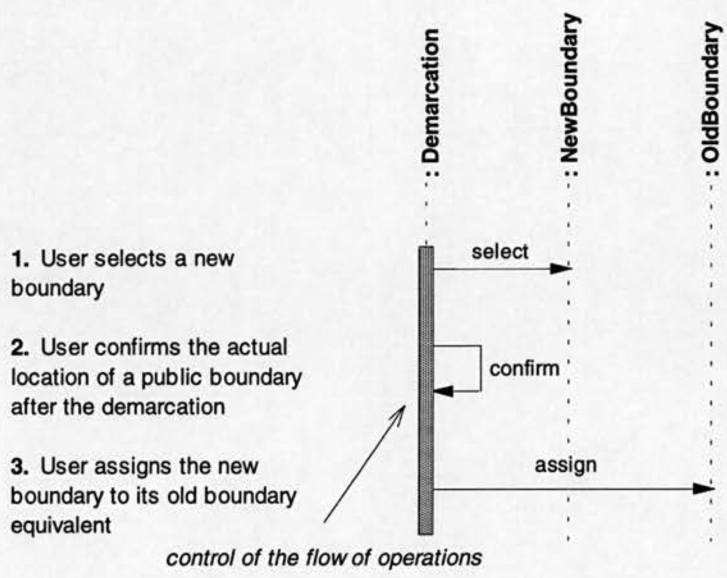

: Demarcation
: NewBoundary
: OldBoundary
1. User selects a new boundary
2. User confirms the actual location of a public boundary after the demarcation
3. User assigns the new boundary to its old boundary equivalent
select
confirm
assign
control of the flow of operations

Figure 5.31 - The Interaction Diagram for the Evolution Tracking Scenario - Demarcation Process

The delimitate, demarcate and select associations in the Class Diagrams (Figure 5.28 and Figure 5.30) denote a semantic dependency and suggest a bi-directional navigation between classes. They also represent the independent incremental modifications in the spatio-temporal data model. In the delimitation process, DraftBoundary is the parent class having Delimitation as the modifier into a resultant NewBoundary class. Moreover, in the demarcation process, NewBoundary becomes the parent class having Demarcation as the modifier into a resultant OldBoundary class.

Consequently, a user will be able to select a draft boundary to be a new boundary, and afterwards, to be an old boundary using the independent incremental mechanism. This allows the user to perform data entry associated with the properties of the classes which are involved in the Evolution Tracking Scenario in such a way that the evolutionary aspects of the scenario is taken into consideration. Classes are interrelated through the incremental mechanism in order to support space-time paths. Their properties are independent of the existence of such a liaison between the classes.

As mentioned in the previous section, the independent incremental modification assures that the lifespan of the instances of each of these classes are independent. HistoricalView classes have been defined for visualising the space-time paths in the Evolution Tracking Scenario and they are presented in Chapter 8.

The following Table 5.4 shows the properties attached to the classes designed for the Evolution Tracking Scenario.

Table 5.4 - Properties of Delimitation, Demarcation, NewBoundary and OldBoundary Classes

classes	**spatial properties**	**non-spatial properties**

Delimitation	none	statutory_document *{Act of Parliament, Boundary Commission Order}*
		map_scale_used_for
		operative_date *{year, month, day}*
		effective_date *{year, month, day}*
		actual_date *{year, month, day}*

Demarcation	none	surveyor_name map_scale_used_for valid *{from, to}*
NewBoundary	line turning_poin t length	description *{operative*, operative not effective**, operative effective**}*
OldBoundary	line	type *{old current, disputed}*

*assigned by an Act or Order
**assigned by an Act or Order which has not yet become effective
***assigned by an Act or Order already effective.

5.4.3 Update Scenario

In the Update Scenario, every change is due to an update procedure, but not every update causes a change. When an update procedure generates changes, such changes are according to the spatial data representation being used in the GIS context. Among the spatial data representations available within GIS, the vector based representation is the most complete because the geometry, topology, and thematic properties of a class of objects can all be employed to describe changes. In contrast, a grid cell representation allows the description of changes by employing only thematic characteristics.

For a vector representation, Armstrong [1988] defines eight possible combinations of changes which can occur in an update procedure. Table 5.5 illustrates these changes associated with their respective update procedures.

Table 5.5 - Main update procedures related to changes

Update Procedure	***Change***
1. creation of a new object	none
2. creation of a new object from a previously existing one	geometry topology thematic
3. status updating of an object	thematic
4. relocation of an existing object	geometry
5. alteration of the spatial relationships among objects	topology
6. relocation of an existing object with an alteration of its spatial relationship with other objects	geometry topology
7. status updating of an object with alteration of its spatial relationship with other objects	thematic topology
8. status updating and relocation of an existing object	thematic geometry

The Update Scenario is deemed to handle the revolutionary states due to natural changes and new demarcation descriptions to public boundaries. Natural changes can be linked to the update procedures 2, 4, 5, and 6 which have been described in the table above. On the other hand, new demarcation descriptions in the Update Scenario are associated with update procedures 1, 2, and 4. Therefore, update procedures 1, 2 , and 4 have been taken to illustrate the changes of a public boundary within the spatio-temporal data model.

These update procedures have been designed to operate over the GroundFeature and OldBoundary classes in such a way that revolutionary states are created, as previously described in the time-geographic framework. The Class Diagram in Figure 5.32 illustrates the design decisions regarding these classes of objects (See Appendix C for the entire Class Diagram of the spatio-temporal data model).

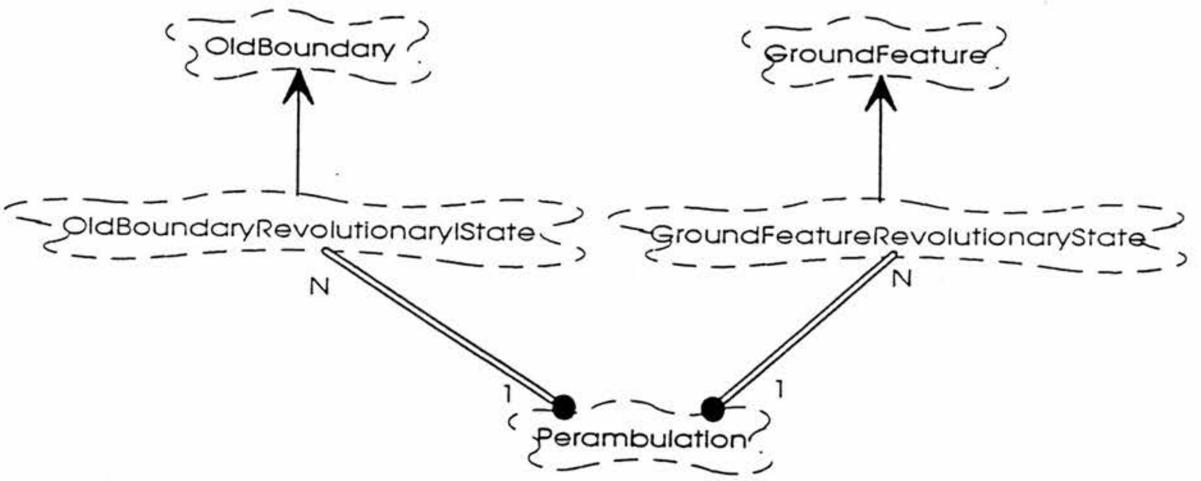

OldBoundary
GroundFeature
OldBoundaryRevolutionaryIState
GroundFeatureRevolutionaryState
N
N
1
1
Perambulation

Figure 5.32 - Class Diagram for the Update Scenario

Both revolutionary states on the diagram present an inheritance relationship with their corresponding evolutionary states. For example, an instance of the OldBoundaryRevolutionaryState class is created by inheriting some properties from its corresponding OldBoundary class. In this case, the spatio-temporal data model is deemed to manage the overlapping incremental modification. New properties can be added to an instance of OldBoundaryRevolutionaryState class whose names do not occur in its corresponding instance of the OldBoundary class. The Table 5.6 illustrates the properties of the GroundFeatureRevolutionaryState class and the OldBoundaryRevolutionaryState class.

Table 5.6 - Properties of GroundFeatureRevolutionaryState and OldBoundaryRevolutionaryState classes

classes	**spatial properties**	**non-spatial properties**
GroundFeature RevolutionaryState	updated_point updated_line updated_area	type*
OldBoundary RevolutionaryState	updated_line	type*

*inherited properties from their respective evolutionary states

The exploratory nature of this design embraces two main aspects in creating discrete revolutionary states using the overlapping incremental modification. Firstly, the instances of the GroundFeature and OldBoundary classes are subject to the previously mentioned update procedures. Secondly, there can exist several revolutionary states for the same GroundFeature and OldBoundary classes.

The update procedures play an important role in the Update Scenario. Figure 5.33 illustrates an example for the system behaviour bearing in mind the three update procedures: the creation of a new object, the creation of a new object from a previous one, and finally, the relocation of a new object.

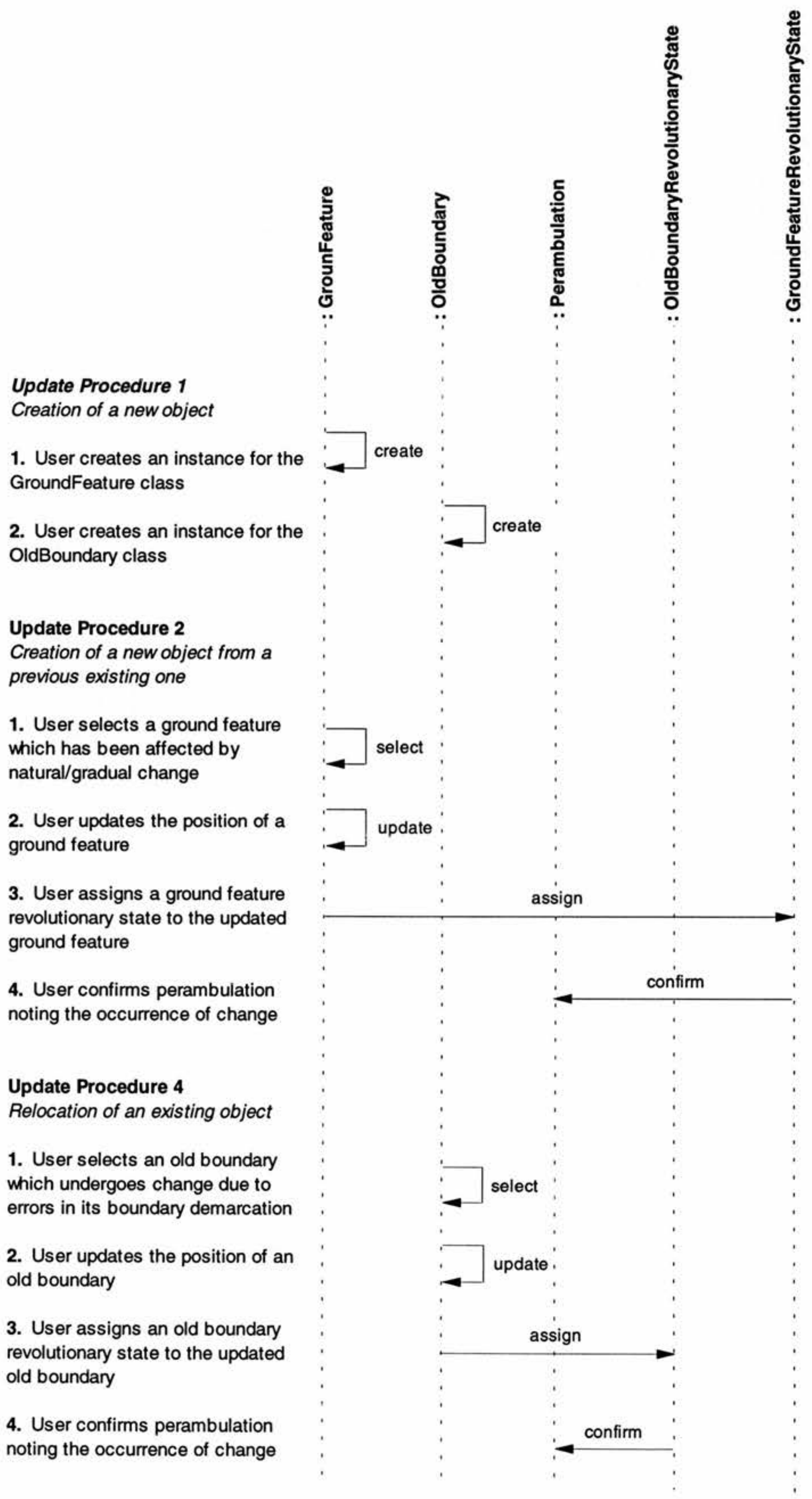

: GrounFeature
: OldBoundary
: Perambulation
: OldBoundaryRevolutionaryState
: GroundFeatureRevolutionaryState
Update Procedure 1
Creation of a new object
1. User creates an instance for the GroundFeature class
create
2. User creates an instance for the OldBoundary class
create
Update Procedure 2
Creation of a new object from a previous existing one
1. User selects a ground feature which has been affected by natural/gradual change
select
2. User updates the position of a ground feature
update
3. User assigns a ground feature revolutionary state to the updated ground feature
assign
4. User confirms perambulation noting the occurrence of change
confirm
Update Procedure 4
Relocation of an existing object
1. User selects an old boundary which undergoes change due to errors in its boundary demarcation
select
2. User updates the position of an old boundary
update
3. User assigns an old boundary revolutionary state to the updated old boundary
assign
4. User confirms perambulation noting the occurrence of change
confirm

Figure 5.33 -The Interaction Diagram for the Update Scenario

Managing such a structure of revolutionary states implies the design of a version management mechanism. Consequently, the approach chosen for the definition, representation, and manipulation of revolutionary states within the spatio-temporal data model is consistent with object-oriented concepts. It was originally proposed by Ahmed and Navathe [1991] for CAD systems. Their approach deals with versioning at the instance level and its versioning scheme can be implemented over any object-oriented platform [Ahmed and Navathe 1991]. This approach is discussed in more detail in Chapter 6.

5.4.4 Archiving Scenario

Two classes have been defined in this scenario which are the OldBoundary and ObsoleteBoundary, representing respectively the old state and obsolete state of a public boundary (Figure 5.34). This assumes that current data are associated to the OldBoundary class. Meanwhile ObsoleteBoundary represents the historical data. Current data are likely to have a higher query access frequency. Historical data archives will tend to become larger and therefore are most likely to be stored on lower cost optical disk.

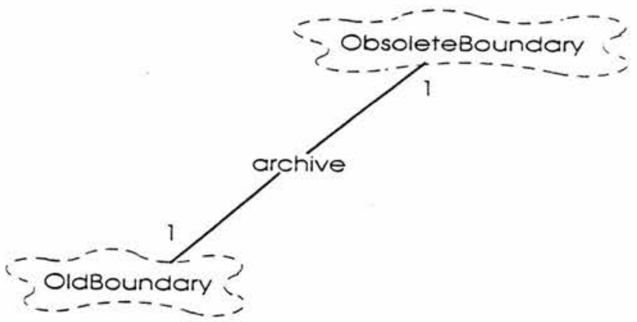


Figure 5.34 - Class Diagram for the Archiving Scenario

The execution of this scenario can be illustrated on the Interaction Diagram in Figure 5.35.

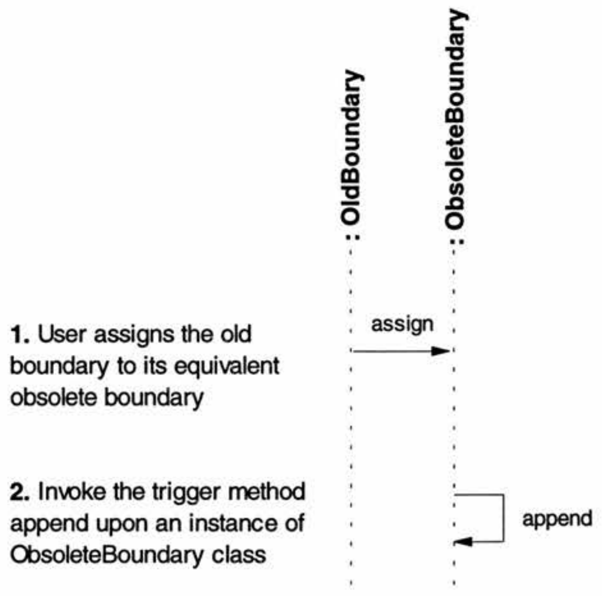

: OldBoundary
: ObsoleteBoundary
1. User assigns the old boundary to its equivalent obsolete boundary
assign
2. Invoke the trigger method append upon an instance of ObsoleteBoundary class
append

Figure 5.35 - The Interaction Diagram of the Archiving Scenario

The trigger method append assumes that current data are stored separately from historical data, each with their own access methods and possibly on different storage media.

5.5 Conclusions

Developing the spatio-temporal data model (See Appendix C for an overview of the spatio-temporal data model) using the Object-Oriented Method proposed by Booch has elucidated several aspects concerning the time-geographic framework. First of all, the design decision of modelling events and states as classes of objects could be taken differently for another application. Perhaps the modelling requirements for a distinct application would require events as methods rather than classes of objects. This would certainly affect the design of the space-time paths within the model. Having decided on modelling events and states as classes of objects, the Class Diagrams utilised in this chapter have proved to be a valuable instrumental tool for the analysis and design of classes of objects and their relationships within the spatio-temporal data model.

Secondly, the design of a space-time path has been defined at the instance level. Different evolutionary or revolutionary states, as well as events, have been defined as classes of objects having their respective relationships. However, the space-time path is only generated by connecting the instances of these classes of objects. Such a connection has been designed using an incremental modification mechanism. Unfortunately, Booch's method does not provide a diagram for modelling different scenarios at the instance level of classes of objects. However, the Interaction Diagrams have been used to illustrate the execution of these scenarios, as well as to visualise the process involved in 'making space-time paths' within the spatio-temporal data model.

On the other hand, the analysis and design of the spatio-temporal data model using an object-oriented method has raised issues in temporal data management. Having a system with several instances of several classes of objects connected to space-time paths has definitely raised a question mark about versioning. Are these instances versions of a public boundary? How should they be managed in order to avoid inconsistency in the representation of space-time paths within the spatio-temporal data model?

In investigating the possibilities of designing a temporal data management mechanism in the spatio-temporal data model, one of the main findings encountered in the analysis was the need to understand the meaning of ***change***. Change is responsible for the existence of evolutionary and revolutionary states on a space-time path. In other words, change carries out actions which create different instances in the space-time paths within the system. However, change has to be deeply understood as an element of the temporal data management of the spatio-temporal data model. This is discussed in more detail in the following chapter.

Chapter 6

The Version Management Approach Employed on the Spatio-Temporal Data Model

6.1 Introduction

"Versioning is the tracking of the evolution of an object's state through time." [Loomis 1992, p.40].

In the spatio-temporal data model, versioning is the tracking of changes of both evolutionary and revolutionary states of a public boundary. States have been modelled as classes of objects and therefore, a version is any instance of these classes which belongs to the space-time path of a public boundary. Moreover, the relationships between events and states have been designed using inheritance as the incremental modification mechanism for the space-time path of a public boundary. Consequently, versioning in the spatio-temporal data model is concerned with handling a predecessor-successor relationship, also termed a parent-child relationship.

Deciding on a version management approach to be employed on the spatio-temporal data model involves the finding of an integration solution between the approach and the model. The version management approach developed by Ahmed and Navathe [1991] has been chosen because of its consistency with object-oriented concepts. It is conceived to be distinct from a snapshot time series because it manages valid changes rather than merely showing them. This chapter describes the main concepts developed in this approach and their application in the spatio-temporal data model.

6.2 What is Still Required in the Spatio-Temporal Data Model ?

On the basis of so many different contexts being researched in temporal data management as shown in Chapter 4, the universality of time can be recognised as the principal element that brings research fields together. In this research, time is important because changes occur on public boundaries. Such changes have been previously denominated as long-, medium-, or short-term changes in the time-geographic framework. Likewise, in the spatio-temporal data model, changes have been viewed as events affecting instances of objects in such a way that evolutionary states are created. Changes have also been associated with update procedures which can occur on an evolutionary state in a way that its respective revolutionary state is created.

In developing a spatio-temporal data model based on time-geography and object-orientation, the temporal data management of change within such a model has certain implications. They are:

- The spatio-temporal data model emphasises the **interaction** between events and states rather than a linear connection among them over a space-time path.
- The space-time path offers an appropriate semantic abstraction for handling **only** valid-time in a spatio-temporal data model.
- The independent incremental modification requires a different version management approach for incorporating change into a database. Most of

the approaches extend the relational model by creating new versions of tables, tuples or attributes to reflect changes occurring over time. A new version management approach is necessary to deal with changes which are represented by the **synchronisation** of events and evolutionary states.

- The overlapping incremental modification mechanism provides a new means for **managing update procedures** within a spatio-temporal data model. It avoids the duplication/redundancy of data in a database because a revolutionary state always inherits the values of the properties of its previous evolutionary state. Besides, it allows update procedures on the properties of an evolutionary state in a non-destructive manner.

However, there is still an essential matter which needs to be clarified, and it is concerned with what actually characterises a change in the spatio-temporal data model.

Three specific properties of a space-time path have been defined in Chapter 2, these being *position*, *orientation*, and *scale*. In fact, these properties are responsible for the density of the space-time paths in the spatio-temporal data model. Position is an important property because it denotes the location of a space-time path. It tells us where and when a change took place. The orientation property reproduces the direction of a space-time path, and finally, the scale property associates a spatio-temporal resolution with a space-time path. In fact, different space-time paths can present different scales, as well as each path can be required to have a single scale or different scales associated with it. This is of fundamental importance in analysing spatio-temporal patterns since different scales can generate different patterns.

A proper understanding of these properties (position, orientation, and scale) is fundamental for effectively managing changes in the spatio-temporal data model.

They have been considered as the core elements involved in temporal data management of change in the spatio-temporal data model. In the following section, a classification of change is presented on the basis of these properties in order to find the most appropriate temporal data management mechanism for the spatio-temporal data model.

6.3 Classifying Change in the Spatio-Temporal Data Model

Transaction time and valid time have been employed to classify three types of change in the spatio-temporal data model. Change is classified as one of the following:

- *data model change*, when change is related to the transaction time in such a way that change represents the schema evolution of a GIS;
- *valid change*, which is related to the valid time when an event occurs creating a state for the lifespan of an entity;
- *bitemporal change*, which relates to transaction time and valid time by creating a bitemporal element for a lifespan of an entity.

6.3.1 Data Model Changes

Banerjee *et al.* [1987] illustrate some changes occurring in the evolution of a schema. Such changes can be considered as *data model changes* in the spatio-temporal data model. They are one of the following:

1. Changes to the components of a class
   - Changes to properties
     - Add a new property
     - Drop a property
     - Change the name of a property

        - Change the type of a property
        - Inherit a different property definition

    - Changes to methods
        - Add a new method
        - Drop a method
        - Change the name of a method
        - Change the implementation of a method
        - Inherit a different method definition

2. Changes to relationships of classes
    - Add a new class relationship
    - Remove a class relationship
    - Change the class relationship

3. Changes to classes themselves
    - Add a new class
    - Drop an existing class
    - Change the name of a class

Data model changes describe the evolution of a spatio-temporal data model which involve updates to the schema. They are not related to the update procedures on the data available in the system. They can be considered as changes which need to be performed in order to rectify errors encountered in the schema of a spatio-temporal data model. Otherwise, they can also be related to the changes which are necessary due to the development of our understanding of the application domain within a spatio-temporal data model.

Data model changes play an important role in a networked GIS with a large number of users. A version management mechanism has to be provided so that a designer can test any data model change to the data model in his/her own version of the whole database.

Some commercial GISs present a mechanism for handling data model changes, for example, Smallworld GIS. The CASE tool[1] in the Smallworld GIS has been specifically developed for handling data model changes. The version management implemented in the CASE tool has the ability to define and modify

[1]CASE tool is an interactive, visually-oriented tool for creating and managing the database schema using graphical notations.

classes of objects. "This uses version management techniques extensively to help functions like the ability to create alternative versions of the data model, apply these to alternative versions of a populated database for testing and development, and finally apply the changes to the master version of the database with a mechanism for propagating the schema changes down to other versions in the database." [Newell and Batty 1993, p.3.2.3]. In Chapter 7, this version management approach for data model changes is discussed in more detail.

The main finding in this investigation is that although data model changes are primarily concerned with updates in the schema of the spatio-temporal data model, they also affect the lifespans of space-time paths. Data model changes affect the position, orientation, and scale of a space-time path in the spatio-temporal data model. Such an influence has been observed during the practical implementation of the spatio-temporal data model in the Smallworld GIS. A general overview has evolved from these empirical observations as an attempt to demonstrate how data model changes can affect a space-time path.

In summary, the following components of Banerjee *et al.*'s classification have been found to be important in the present context:

1. changes to the component of a class
   - changes to properties: they affect the position of a space-time path;
   - changes to methods: they affect the orientation of a space-time path.
2. changes to relationships of classes: they affect the orientation of a space-time path
3. change to classes themselves: they affect the position and the orientation of a space-time path.

A more detailed investigation of data model changes and their effects in the position, orientation, and scale properties of a space-time path is required. However, it has been considered to be beyond the scope of the present study.

6.3.2 Valid Changes

Valid changes are related to space-time paths of the spatio-temporal data model. They play an important role in creating both evolutionary and revolutionary states in the way that the orientation property of a space-time path is handled. In the spatio-temporal data model, valid changes can present a linear or a branching structure. A space-time path having a linear structure, has a unique orientation. Conversely, several directions can be defined for a space-time path in a branching structure (Figure 6.36).

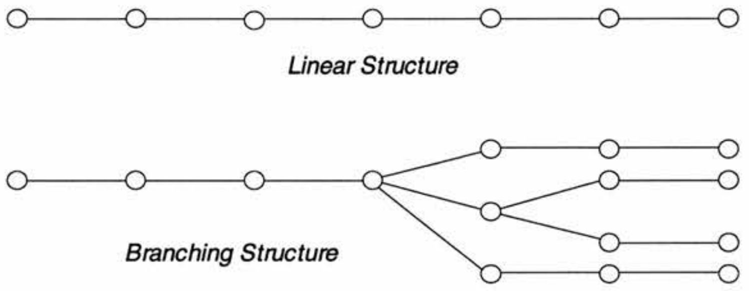


Figure 6.36 - Possible structures for a space-time path

As mentioned in Chapter 3, the spatio-temporal data model is deemed to deal with explanatory reasoning in the temporal domain. This involves the production of a description of the evolution of public boundaries at an earlier time that accounts for the public boundaries being the way they are at a later time. Considering the evolution of public boundaries, such a reasoning task implies that the space-time path is linear. There is no possibility of having two space-time paths for the same public boundary over its past. Consequently, the linear structure has been adopted as the orientation structure for each space-time path within the system.

Each space-time path has a linear structure in which states and events are consecutively and chronologically connected in order to explain the evolution of public boundaries. The states reveal the valid changes while the events disclose when these changes have occurred (Figure 6.37).

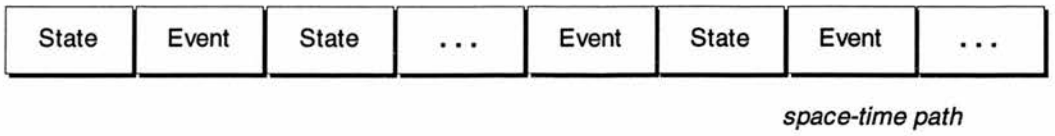


Figure 6.37 - The space-time path

Further investigation is required in order to explore the incorporation of a branching structure within the spatio-temporal data model. This will be very important in predicting both evolutionary and revolutionary states of a public boundary. However, this has been considered to be beyond the scope of this investigation.

Considering the position property of a space-time path, three kinds of space-time paths can be defined as being one of the following:

- the discrete space-time path: each position in the space-time path has a single successor;
- the dense space-time path: between any two positions of a space-time path another position exists ;
- the continuous space-time path: contains no gaps.

Snodgrass [1992] argues that any implementation of any data model with a temporal dimension will of necessity have to hold some discrete encoding of time. Bearing this in mind, the discrete space-time path has been adopted for the spatio-temporal data model. Such a path corresponds to a nondecomposable line of space and time with an arbitrary measurement unit. In terms of time, this unit is termed a chronon [Snodgrass 1992] which is the smallest duration of time that can be represented in a space-time path.

Five fundamental types of discrete encoding of time can be identified as follows [Snodgrass 1992]:

- Nominal, for example: today, May 13;

- Binary: earlier than, later than;

- Ordinal: time ago, long ago;

- Interval: event A is 1 year;

- Ratio: event A starts at 8 and ends at 6.

In the spatio-temporal data model, both nominal and ratio types of encoding time are necessary for timing events in the space-time path. In this case, the time-stamp attributes are attached to the events of the space-time path as required, representing nominal or ratio types. For example, the nominal time-stamp has been defined for the allocation event because this event occurs at a specific position in time. However, the ratio time-stamp has been defined for the demarcation event because a demarcation of a public boundary on the ground occurs over a period of time.

Scale is another property of a space-time path which is related to the spatial-temporal resolution. Although a spatio-temporal resolution can be defined for space-time paths, the implementation of the spatio-temporal data model in a GIS gives us the possibility of analysing the spatio-temporal patterns of space-time paths at any preferred scale. This has been regarded as an advantage in implementing the spatio-temporal data model in a GIS.

6.3.3 Bitemporal Changes

The concept of the bitemporal element is needed in order to represent the transaction time/valid time association, i.e. when the state of an object has changed in the real-world and when it has been changed in the system.

Several applications in a GIS context require information on the state of an object when it has been modified in the system in comparison to its change in the real-world. Worboys [1994] is investigating the possibility of creating bitemporal elements for spatial chains using object-oriented concepts. The implementation has been using the Smallworld GIS which has a discrete model for linear time. So far his findings show that the critical areas for research are: the development of an appropriate data model, the indexing mechanism for this model and related performance questions. Further discussion of these aspects of bitemporal changes is beyond the scope of this investigation.

6.4 The Version Management Approach

Data model and bitemporal changes have been considered to be of vital importance in developing spatio-temporal data models in GIS. This is mainly because they can offer different perspectives on the meaning of change in spatio-temporal data modelling. One main perspective is that the essence of a lifespan of an entity is enforced by many different states over the past, present, and future times of the real-world and a GIS.

This research attempts to manage the *valid changes* which occur in the spatio-temporal data model. The overall goal involves the management of space-time paths of public boundaries in such a way that 1) the orientation (the linear structure of a space-time path); 2) the position (the discrete encoding of time); and finally, 3) the scale (the spatio-temporal resolution of a space-time path), are handled by the temporal data management mechanism in the spatio-temporal data model.

This introduces the need for a versioning mechanism which is a vital tool for implementing temporal data management in the spatio-temporal data model. In order to manage valid changes in the spatio-temporal data model, the version management approach suggested by Ahmed and Navathe [1991] has been selected. This approach is discussed in more detail in the following section.

6.4.1 Modelling Abstractions for Supporting Versioning

In the spatio-temporal data model, the evolutionary aspects of public boundaries have been characterised by events, evolutionary and revolutionary states of a space-time path. These elements have been modelled as classes of objects within the spatio-temporal data model. Ahmed and Navathe [1991] propose, as a basic modelling construct for supporting a version management mechanism, the classification of different types of classes of objects. This classification is based on three system-defined types of classes of objects. Therefore, each class in the model can pertain to one of the following system-defined types:

- Generic Class identifies the class which represents the essence of a set of objects and contains a high level of abstraction of its functionality;
- Versioned Class is a subclass of the Generic Class;
- Unversioned Class represents the unversioned objects.

Considering the classes of objects already designed in the spatio-temporal data model, they can now be associated with their respective system-defined types in the following manner:

- Generic Classes: PublicBoundary and GroundFeature;

- Versioned Classes: DraftBoundary, NewBoundary, OldBoundary, ObsoleteBoundary, GroundFeatureRevolutionaryState, and OldBoundaryRevolutionaryState;

- Unversioned Classes: Assumption, Allocation, Delimitation, Demarcation, and Perambulation.

Both PublicBoundary and GroundFeature classes have previously been defined in Chapter 5 as primary classes within the spatio-temporal data model. Consequently, they have been identified as Generic Classes.

Versioned classes represent the different evolutionary and revolutionary states within the spatio-temporal data model. In essence, the lifespan of a public boundary can be represented by different versioned classes which are its different evolutionary and revolutionary states over time.

Unversioned classes represent the events over the space-time path. Therefore, they hold a timestamp which is the valid time corresponding to the lifespan of the state of a public boundary. In the spatio-temporal data model, the timestamp is an attribute value associated with the valid time which can be nominal type (for example, May 27) or ratio type (for example, event demarcation starts in June 17 and ends in July 29).

Ahmed and Navathe [1991] also propose the use of a version graph (Figure 6.38) to manage the historical evolution of versioned classes. Such a graph is devised as being a disconnected, directed, acyclic graph in which the nodes represent versions and the edges illustrate the successor/predecessor relationship. All versions constitute a version set. In this example, the D.g is the versioned class which presents a number of D.v versions belonging to its version set.

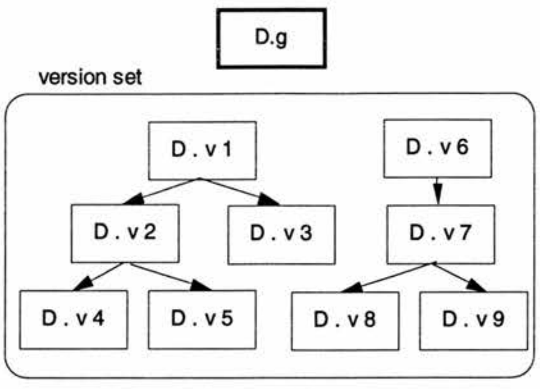


Figure 6.38 - An example of a Version Graph
(Source: Ahmed and Navathe 1991, p.223)

Such a version graph concept has been adapted in order to be incorporated into the spatio-temporal data model. In a version graph, different versions are deemed to belong to the same versioned class. This does not occur in the spatio-temporal data model. In this case, each version of a public boundary belongs to a different versioned class in order to create space-time paths within the system. Therefore, the version graph is deemed to represent the link between versioned classes which are, in fact, the versions themselves in the spatio-temporal data model. Figure 6.39 illustrates the version graph obtained for the versioned class which has previously been defined as DraftBoundary, NewBoundary, OldBoundary, ObsoleteBoundary, OldBoundaryRevolutionaryState, and GroundFeatureRevolutionaryState classes.

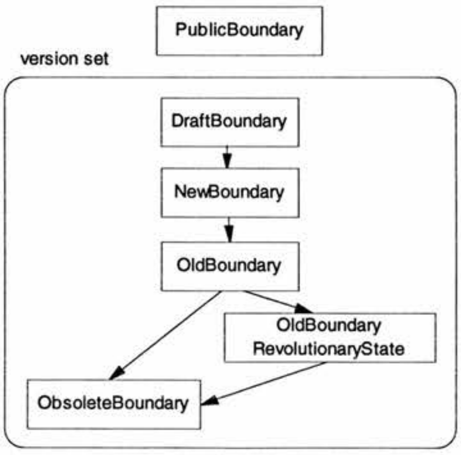


Figure 6.39 - The version graph of the spatio-temporal data model

A version graph plays an important role in the temporal data management of the spatio-temporal data model because it helps to visualise the space-time path without depicting the events. Therefore, it can be used as a modelling tool for designing the versioning mechanism within the system. In defining what versions are in the spatio-temporal data model, the next step is related to how these versions should be identified and distinguished within the spatio-temporal data model. This is discussed in the following section.

6.4.2 Version Representation

Identifying different versions in a consistent configuration is the main issue in defining a version representation within the spatio-temporal data model. Basically, two main strategies are utilised to represent multiple versions of an object. Firstly, version numbers or timestamps may be associated with every attribute or relationship of an object. These attributes or relationships may then be chained, thus having an historical order. The result is a single object and identity (OID) with the versions actually associated with the attributes. This approach could have advantages only in models in which the objects have several attributes as well as changes occurring to only a few attributes.

The second strategy is to track versions at the level of objects rather than attributes. A new version of an object is created by chaining the old object and any older versions to which it has been chained. As a result, a different identifier (OID) is associated with the new version, and thus, each version has its unique identifier within the version configuration.

So far no performance studies have been undertaken on these alternatives. The second strategy has been taken in the spatio-temporal data model because it emerges as a more object-oriented approach to track versions. Considering that the incremental modification approach designed in the spatio-temporal data model is based on an inheritance mechanism of the properties of an object, it would be unfeasible to attach the versions at the attribute level. Versioning needs to be done at the instance level, i.e. at the object level.

6.4.3 The Dynamic Behaviour of Versions

The version configuration is a complex structure in which the dynamic behaviour of versions needs to be managed. The primary issue involves the supervising decision of what should be versioned in the spatio-temporal data model. In handling the dynamic behaviour of versions, a mechanism can be provided for controlling unnecessary proliferation of versions.

Classifying the properties of classes of objects has been proposed by Ahmed and Navathe [1991] as an approach in order to provide a mechanism for controlling update propagation in versions. This is achieved by defining every property of a class as being one of the following attributes:

- invariant attributes which cannot be modified or deleted at any time;
- version significant attributes which can be updated only in a structured manner. An update in one of these attributes creates versions of an object bearing this change;
- non-version significant attributes can always be updated without creating a new version.

For the spatio-temporal data model, this classification can be applied to the attributes of the versioned classes as well as the generic classes. Therefore, a classification has been developed for the attributes of these classes. It is important to emphasise that such a classification is not deemed to embrace the infinite range of possible valid changes which can occur over public boundaries. The proposed classification is based on the valid changes which have been associated with the update procedures defined for the spatio-temporal data model. They are deemed to handle the valid changes due to natural changes and changes due to new demarcation descriptions of public boundaries as described in Chapter 3 and Chapter 5.

Table 6.7 illustrates the classification carried out, bearing in mind the update procedures previously defined in the spatio-temporal data model as: 1) creation of a new object, 2) creation of a new object from a previous existing one, and 3) relocation of an existing object. These update procedures are due to valid changes which have occurred over a public boundary. In the spatio-temporal data model, they are applied to the GroundFeature and OldBoundary classes. Consequently, the classification shows both classes having version-significant attributes.

Moreover, such version significant attributes must be updated in a structured manner. An update in one of these attributes pertaining to the GroundFeature and OldBoundary classes creates versions which are respectively the instances of the GroundFeatureRevolutionaryState and OldBoundaryRevolutionaryState classes. This is illustrated in more detail in Chapter 8.

Table 6.7 - Classification of the attributes of the versioned and generic classes

Versioned Classes	Version Significant	Non-Version Significant	Invariant
DraftBoundary	none	line length mereing_point type description	none
NewBoundary	none	line turning_point length description	none
OldBoundary	line	none	type
ObsoleteBoundary	none	none	line turning_point length type
OldBoundary RevolutionaryState	none	none	updated_line type
GroundFeature	point line area	none	type
GroundFeature RevolutionaryState	none	none	updated_point updated_line updated_area type

6.4.4 Version Management

Based on the evaluation by Dadam *et al.* [1984] as described in Chapter 4 (section 4.2.3), the backward oriented accumulative strategy has been adopted in the spatio-temporal data model. The main advantage seen in this strategy is that for each version in the spatio-temporal data model, an accumulative delta is obtained due to a given update. In other words, the delta creation for obtaining the successor-in-time version in the space-time path can be activated by the occurrence of a valid change within the spatio-temporal data model. Therefore, a successor-in-time version can only be created if an update is triggered within the system This involves the design of update constraints for handling the dynamic behaviour of versions within the spatio-temporal data model. Ahmed and Navathe [1991] propose two types of updates within a system, termed nonatomic and atomic updates.

A nonatomic update is characterised by the update of more than one attribute of an object at the same time. Conversely, an atomic update refers to the modification of only one attribute of an object at a time. Atomic updates are rarely used in GIS contexts.

In order to provide nonatomic updates and to maintain the classification of version significant, non-version significant, and invariant attributes, Ahmed and Navathe [1991] have designated three different kind of states for a versioned object. These being validated, stable, and transient states. The main characteristics of these states have been described as follows [Ahmed and Navathe 1991, p.224-225]:

- Validated Versions

  1. No modification is allowed on validated versions.

  2. All attributes are invariant.

  3. New versions can not be derived from them.

- Stable Versions

  1. Their version-significant attributes cannot be modified.

  2. Non-version significant attributes can be modified.

  3. Stable versions can be promoted to transient versions.

- Transient Versions

  1. All versions begin in the transient state.

  2. Stable versions can be derived from them.

  3. Transient versions can be promoted to validate versions.

In addition, Ahmed and Navathe [1991] describe several possible version operations between these three main version states. For the spatio-temporal data model, all three version states have been incorporated and two operations have been selected to define the transition between version states (Figure 6.40).

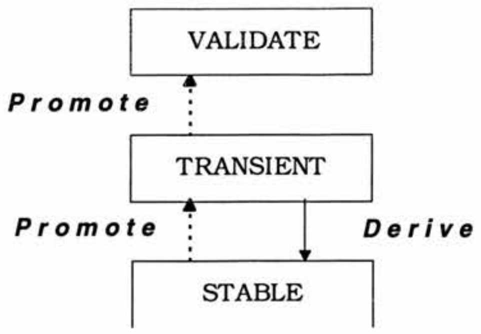


Figure 6.40 - Transition of version states

The dashed arrows signify that valid changes have occurred between the version states. The operation involved is named Promote by Ahmed and Navathe [1991] and it is the operation defined on stable and transient version states. It promotes a stable version state to a transient state, as well as a transient state to a validate state. The solid arrow signifies that no valid changes have occurred over a version state. In this case, the Derive is the operation involved which creates a copy of the operand version. It derives a version state of the same type as the given transient or validate version state.

Three update procedures have been selected in the spatio-temporal data model (See Chapter 5). They are:

1) creation of a new object;
2) creation of a new object from a previous existing one;
3) relocation of an existing object.

In Figure 6.41, a transition diagram illustrates the effect of applying the Promote and Derive operations in order to create version states for the above update procedures. The GroundFeature, GroundFeatureRevolutionaryState, OldBoundary and OldBoundaryRevolutionaryState classes have been used as examples for the creation of a new version, creation of a version from a previous existing one, and finally for the execution of an update procedure over a version state.

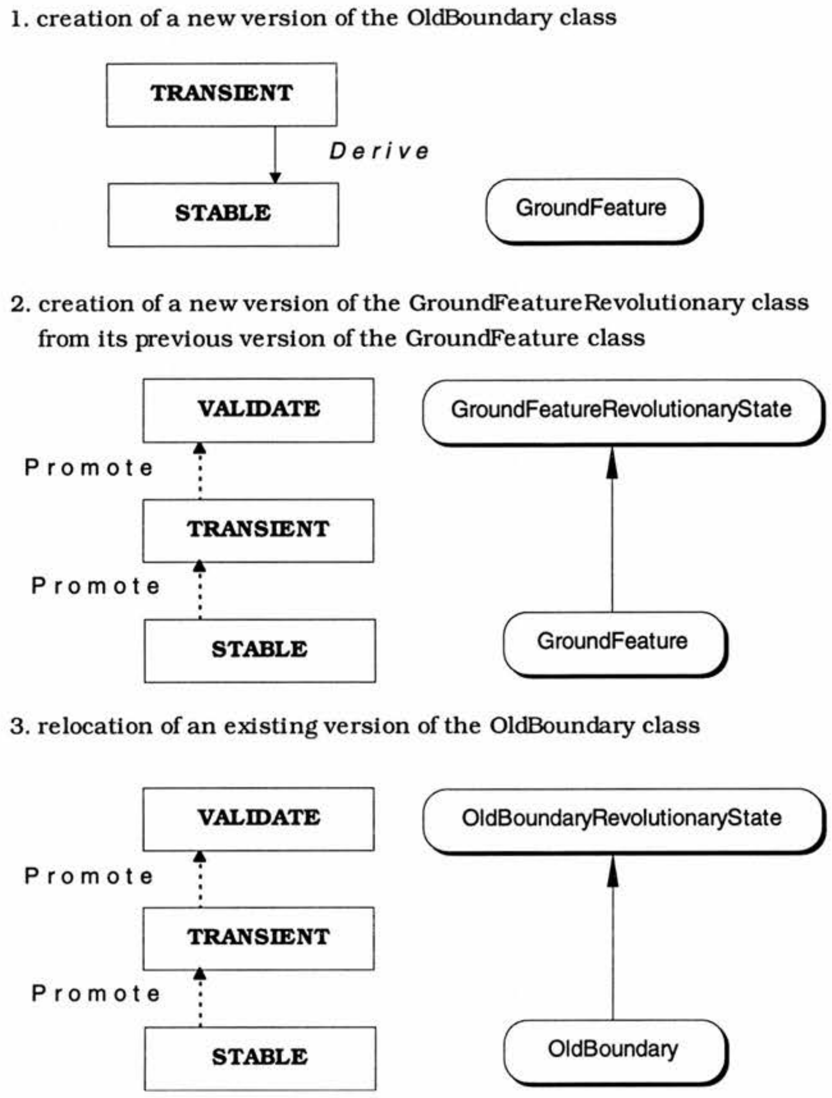

1. creation of a new version of the OldBoundary class
TRANSIENT
Derive
STABLE
GroundFeature
2. creation of a new version of the GroundFeatureRevolutionary class from its previous version of the GroundFeature class
VALIDATE
GroundFeatureRevolutionaryState
Promote
TRANSIENT
Promote
STABLE
GroundFeature
3. relocation of an existing version of the OldBoundary class
VALIDATE
OldBoundaryRevolutionaryState
Promote
TRANSIENT
Promote
STABLE
OldBoundary

Figure 6.41 - Example of a transition diagram in the spatio-temporal data model

6.5 Conclusions

The versioning approach proposed by Ahmed and Navathe has provided a unifying solution for the effective management of versions within the spatio-temporal data model. One result of this is that the maintenance of versions within the spatio-temporal data model can be implemented in a GIS system. That means, the abstractions developed in the spatio-temporal data model, such as space-time paths involving events and states, can be implemented into a GIS system. Such an implementation will provide a mechanism for maintaining a consistent update propagation that prevents an unnecessary version proliferation within the system.

This attempt at applying this version management approach to a fairly realistic application such as the evolution of public boundaries, indicates that it may have much to offer to other applications, both in conceptual and practical terms. However, the approach has only been presented in a single publication up until now [Ahmed and Navathe 1991], and therefore, accurate testing and refining in other realistic settings is required.

The next chapter introduces the main features of the Smallworld GIS which have been relevant for the implementation of the spatio-temporal data model. Implementation issues are identified for characterising how the spatio-temporal data model may be adapted to the features of the system's configuration of the Smallworld GIS.

Chapter 7
The Main Features of the Smallworld GIS for Implementing the Spatio-Temporal Data Model

7.1 Introduction

In this chapter, the focus is on the implementation aspects for incorporating the spatio-temporal data model into a GIS. Firstly, an overview of the Smallworld GIS (Release 2.0.2) is provided for summarising the structure involved in the system in terms of its proprietary database, its programmatic interface, and its available tools and utilities. The main concepts found in the data model of the Smallworld GIS are also explained. Several modelling abstractions of this model are examined in order to provide a basis for an evaluation of the implementation decisions which have been taken.

Secondly, the object-oriented features required by the spatio-temporal data model are presented and the Smallworld support for providing them is discussed. The outcome provides an understanding of the main concepts in object-orientation which are available in the Smallworld GIS.

The overall goal of this chapter is to explore the issues in implementing the spatio-temporal data model as well as choosing which object-oriented features of Smallworld to use in order to satisfy the model needs. Therefore, the implementation aspects involved in incorporating the spatio-temporal data model into the Smallworld

GIS are presented. The main issues encountered during the implementation process are described. The wide disparity in the degree to which the Smallworld GIS actually satisfies the application needs becomes apparent when the implementation aspects are discussed.

7.2 Smallworld GIS Overview

The Smallworld GIS is a multi-user database management system with a distributed client-server architecture. It basically consists of a *proprietary database*, a *programmatic interface*, and *a set of tools and utilities*.

The proprietary database of the Smallworld GIS is called the Version Managed Data Store (VMDS). The only type of data structure found in the VMDS is the table, except for one block which forms the hierarchy of indices mapping different versions of the database.

> "The implementation ...(of the Smallworld GIS) is based on a tabular datastore constructed from conventional B*-trees, with the fundamental versioning mechanism built into the underlying data structure itself. ... This level of the implementation permits multiple versions of the data to co-exist, with blocks which are common to two or more versions being shared between them. Note that the approach does not demand that all blocks reside in the same file. In particular, benefit can be gained in the long term by storing a version on a mass storage medium such as CD ROM and maintaining the changes on conventional disk." [Newell *et al.* 1994, p.6/8].

Having a B*-tree structure, the VMDS uses the hierarchical nature of the tree to permit multiple versions to be supported economically. When a block is updated, it is put back somewhere else ('rehoused'), which means the reference to it must be updated, and that block is also rehoused. This happens all the way up to the top of the tree, at which point there are now two roots, one seeing the old version, the other seeing the new one [Easterfield 1993]. In fact, several roots can coexist within the VMDS, so that each user is always presented with data from a consistent version of the database (Figure 7.42).

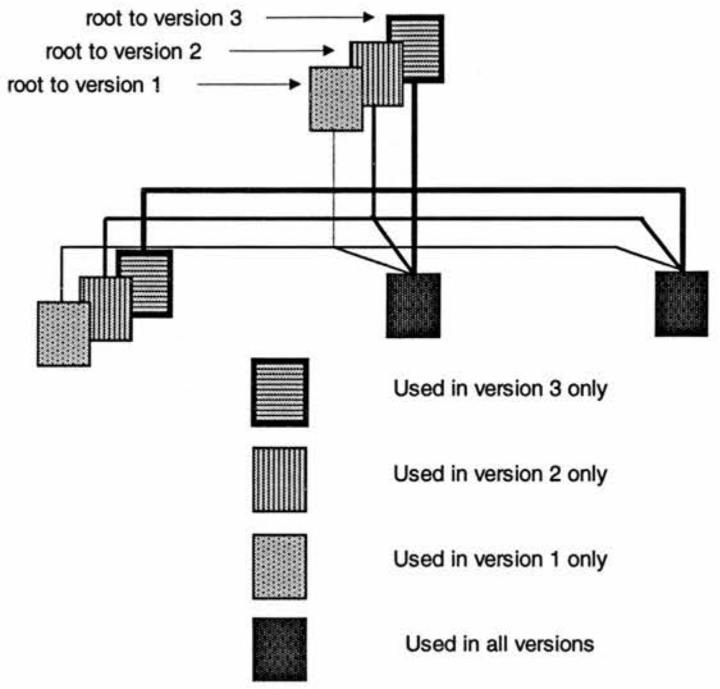


Figure 7.42 - The block sharing in a version managed B*-Tree
(Source: Newell et al. 1994, p.6/8)

Within the VMDS, tables are stored in files as following:

- the *rwo.ds* file which holds tables with data about the system, indexes, and tables created by the user;

- the *gdb.ds* file holding the tables with the geometry and topological data;

- the *raster.ds* file containing the raster background map data;

- the *style.ds* file holding the tables with data for graphical rendering of objects in the system;

- the *point_sym.ds* file which contains tables with the definitions of geometry;

- the *message.ds* file holding the message data tables;

- the *overlays.ds* file consisting of temporary tables for overlay analysis.

These files are allocated to *views* in a way that the files represent the 'disk database' state while a view represents the 'virtual database' state (Figure 7.43). A view allows multiple users to access the VMDS simultaneously although no knowledge of a view is held by the VMDS itself. Several views can be created for the same VMDS. Each view is assigned to a different user, but the same concurrency control mechanism applies in each case.

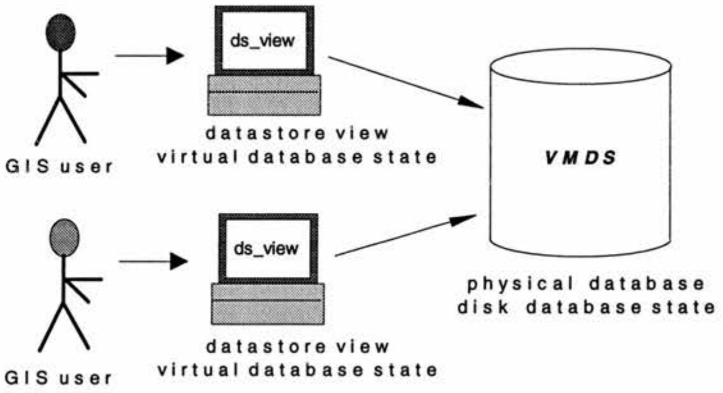


Figure 7.43 - The disk and virtual database views

Each data store view (ds_view) is always attached to an alternative database version of the VMDS. It contains a whole mini-database which consists of a set of tables containing sufficient information for the VMDS engine to bootstrap itself together. Such a mini-database is called a physical alternative. It contains tables which describe things about the alternative as a user sees it (the logical alternative)[Easterfield 1993]. Figure 7.44 illustrates these tables that include information about each alternative and their named versions (checkpoints).

```
Magik>2print(ds_version_view.sys!all_slot_names())
$
simple_vector(1,10):
1        :ds_version_view!files
2        :ds_version_view!frozen_files
3        :ds_version_view!view_name
4        :ds_version_view!mode
5        :ds_version_view!searchpath
6        :ds_version_view!alternative_path
7        :ds_version_view!checkpoint
8        :ds_version_view!alternative_history
9        :ds_version_view!params
10       :ds_version_view!auth_view

Magik>2print(database_view.sys!all_slot_names())
$
simple_vector(1,18):
1        :ds_version_view!files
2        :ds_version_view!frozen_files
3        :ds_version_view!view_name
4        :ds_version_view!mode
5        :ds_version_view!searchpath
6        :ds_version_view!alternative_path
7        :ds_version_view!checkpoint
8        :ds_version_view!alternative_history
9        :ds_version_view!params
10       :ds_version_view!auth_view
11       :database_view!collections
12       :database_view!view_collections
13       :database_view!uvas
14       :database_view!external_databases
15       :database_view!dd_types
16       :database_view!dd_joins
17       :database_view!dd_domain_tests
18       :database_view!undo_info
```

Figure 7.44 - An example of two tables containing information about an alternative

Because the VMDS is designed primarily for long transactions, with gross locking at the alternative level, blocks can be usefully cached at the client. The server merely acts as a funnel for block transfer, and for coordinating what locking there is. This effectively gives distributed processing, and enables operations such as redraw to take place directly out of the database since the data are being cached on the client's workstation [Easterfield 1993].

Figure 7.44 also illustrates the programmatic interface in Smallworld which is called the Magik programming environment. This environment consists of methods and classes which are independent of those existing in the VMDS. It also comprises a Magik compiler which compiles Magik code into Magik virtual machine instructions. In this environment, system programming, applications development, system integration and customisation are all written using the Magik programming language.

The Magik programming environment can be described as following:

- it is an object-based environment written in C in which all persistent data are related to a virtual database. It is deemed to handle the graphics and user interaction, the database access, and external database systems, such as Oracle and Ingres.

1. - the Magik programming language has a procedural style and it has been primarily inspired by Smalltalk, Lisp, and CLU programming languages. It is weakly typed in order to gain flexibility during run-time messages in its polymorphic structure.

The Magik programming environment is always saved in a Magik image which represents the state of the environment rather than the state of the VMDS. For example, methods stored in the Magik image are independent of those stored in the VMDS. Generally, the user should avoid dependence between code in the image and code in the VMDS, mainly because the database code may change between alternatives and cannot necessarily be relied on (Figure 7.45).

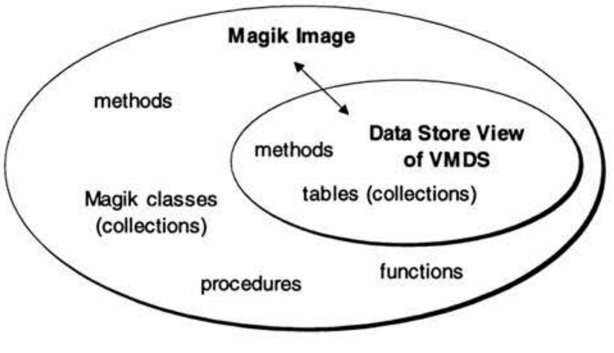


Figure 7.45 - The relationship between a Magik image and VMDS

In fact, Magik as a programming environment, manages *all* the features of the Smallworld GIS, and therefore, it is the way a developer, designer, and programmer operate the whole system: its data store views, images, the VMDS and the GIS functions.

Most people who use the Smallworld GIS fit into one or more of these categories: developer of an application, designer of the database or programmer of computing resources. Each kind of user has a different role in relation to the Smallworld GIS, and a somewhat different way of looking at the system.

A set of tools for database design and application development is provided by the Smallworld GIS. These tools let a user construct products and applications by using the VMDS as a foundation.

Seven database partitions are found in the VMDS of the Smallworld GIS. They are different parts of the VMDS holding data for specific purposes. Each partition within the VMDS is managed in its own view in such a way that any manipulation of data in the Smallworld GIS has to be performed through a view of one or more partitions. These partitions comprise the following (Figure 7.46):

- the Version Management **ACE** Database (Application Configuration Environment) Partition: holding the data about the user environment, such as top-level menus, default settings and saved views.

- the Version Management **GIS** Database Partition: holding the objects which build up the data model for an application.

- the Version Management **Style** Database Partition: holding all information about the graphical display of geometry in views, drawing, and plots.

- the Version Management **Case** Database Partition: holding the data model schema of an application.

- the **NTF** (National Transfer Format) Database Partition: holding the data in the National Transfer Format.

- the **Message** Database Partition: holding the message information.

- the **Authorisation** Database Partition: holding data for user-groups, user-names, rights and access to data.

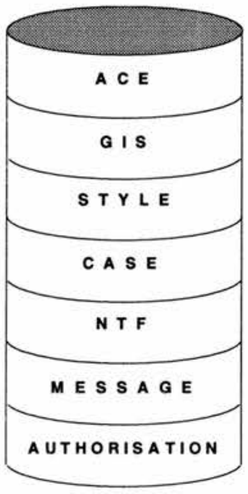


Figure 7.46 - The Database Partitions of the VMDS

The following tools are executed by the database partitions:

- the **ACE Tool** is a user-defined interactive interface for providing different environments for different users in terms of accelerator keys, rendering functions, and display scales.

- the **GIS Tool** is an interactive, visually-oriented tool for accessing the VMDS and performing GIS functions, such as shortest path visualisation. It represents the GIS application within the system in which the geographical world is defined. This involves the definition of the extent of the application area (X_{min}, X_{max}, Y_{min}, Y_{max}).

- the **Style Tool** is also an interactive, visually-oriented tool, but it is used for creating symbols, patterns, and styles for the visualisation of objects which exist in the VMDS.

- the **Case Tool** is an interactive, visually-oriented tool for creating and managing the structure, relationships, and contents of the VMDS database. The user can design the database schema using graphical tools, and then generate the corresponding Magik code from the schema. This can save a good deal of coding time during the process of applying the definitions created in the schema into the GIS application.

- the **NTF Tool** provides an interactive interface for inserting data having a National Transfer Format into the system.

- the **Message Tool** is an interactive interface for supporting the registration and administration of messages within the system.

- the **Authorisation Tool** provides an interactive interface for creating and managing user-groups, user-names, rights and access to data.

7.3 Object-Oriented Concepts in Smallworld GIS

Having discussed the main concepts in object-orientation in the previous chapters, this section explores these concepts in the context of the Smallworld GIS. Therefore, some object data management features have been selected as primary ones in order to implement the spatio-temporal data model in the Smallworld GIS.

This selection has been based on both the static and dynamic configurations which have been designed for the spatio-temporal data model. Basically, the static configuration is related to the object properties (attributes) and the relationship amongst objects. In Smallworld, the definition of object properties and their relationships for a particular application are loaded into the VMDS using the Case Tool.

On the other hand, the dynamic configuration is related to the operations on objects and properties. In Smallworld, the methods for a particular application are loaded into the VMDS using the exemplars file, which is a Magik source code file defining the method.

Bearing in mind both static and dynamic configurations, the 'desirable' object-oriented features for implementing the spatio-temporal data model in the Smallworld GIS have been defined as being the following:

- *Object Identifiers* which allow an object to be referenced via a unique internally generator number.

- *References Attributes* which are used to represent relationships between objects. They are analogous to pointers in a programming language or to foreign keys in a relational system. But references attributes cannot be corrupted, whereas pointers can be. Also references attributes are not associable with a user-visible value, whereas foreign keys are.

- *Collection Attributes*, such as LIST, SET or ARRAY of values.

- *Derived Attributes*, which are defined procedurally rather than stored explicitly. A procedure is specified to be executed when the value is retrieved or assigned.

- the *Referential Integrity* Level which defines the correctness of references when an object is deleted or a relationship is changed. There are five possible levels of referential integrity [Cattel 1991]:

    1. No integrity checks

    2. The system may delete objects automatically when they are no longer accessible by the user, for example, garbage collection algorithms in the GemStone system.

    3. The system may require that objects be deleted explicitly when they are no longer used, but may detect invalid references automatically, for example the IRIS system.

    4. The system allows explicit deletion and modification of objects and relationships, and may maintain automatically the correctness of relationships as seen through all objects, for example, the ONTOS and Probe systems.

    5. The system allow the database designer to specify custom-tailored referential integrity semantics for each object or relationship.

- the *Aggregation Relationship* grouping parts into a whole.

- the *Inheritance Relationship* between objects.

- *Methods* which are associated with objects as well as 'free methods'(independently defined methods).

In investigating the support provided by the Smallworld GIS for these features, an important distinction can be made between a Magik image and a data store view. This is because some features are available only in a Magik image, and others only in a data store view, and very few are available in both image and view. Table 7.8 summarises the main findings about the object-oriented features available in the Smallworld GIS.

Table 7.8 - Object-oriented features available in the Smallworld GIS

Feature required by the spatio-temporal data model	**Magik Image**	**Data Store View**
1. Allow direct access to object identifiers	Yes	Yes
2. Allow the object to have meaningful object keys (which would be the equivalent to primary keys in the relational model)	Yes (object keys)	Yes (primary keys)
3. Allow the use of references attributes	No	No
4. Allow collection attributes (list, set, arrays)	Yes	No
5. Allow derived attributes	No	Yes
6. Allow update of derived attributes	No	No
7. Allow referential integrity	Yes (level 2)	Yes (level 4)
8. Allow aggregation relationships	No	No
9. Allow inheritance	Yes (Multiple Inheritance)	No
10. Allow polymorphism	Yes	No
11. Allow methods associated with objects	Yes	Yes
12. Allow free methods	Yes (Procedures)	No

Although a Magik image is associated with a data store view within the Smallworld GIS, the available object-oriented features are distinct between them. The Magik image holds more object-oriented capabilities than a data store view of the VMDS. This indicates the existence of a VMDS being managed by an object-oriented environment rather than having a fully object-oriented database system. The main consequences of having such a configuration for implementing the spatio-temporal data model is discussed in more detail in Section 7.5.

7.4 The Data Model of the Smallworld GIS

The Real World Object (RWO) is the fundamental element of the data model in the Smallworld GIS. It represents a real world phenomenon within the system, which has been created by a user. An RWO has the basic behaviour of a slotted object of a Magik image. However, for a RWO, its structure is defined in the data store view of the VMDS using the define_rwo_table() method, and also, its behaviour is declared as an exemplar in the Magik image using the declare_record_exemplar() method. Associating the exemplar declaration with the structure definition is carried out when initialising the data store view of the VMDS, by invoking the method gis_init() in the Magik image.

Basically, a RWO consists of a record (or slotted object), each record being composed of fields (or slots). An ensemble of RWOs which belong to a class is called a collection (Figure 7.47).

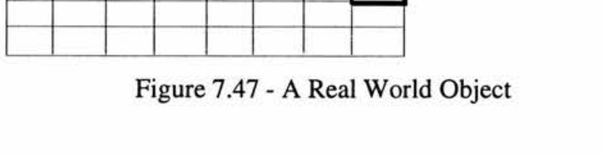


Figure 7.47 - A Real World Object

In the data model of Smallworld GIS, there are three types of collections of RWOs. They are:

- ds_collection: a VMDS collection

- view_collection: a collection built from other collections out of select, join, and project operations.

- extdb_collection: a collection from an external database such as Oracle or Ingres.

A vast protocol is available in a Magik image for operating these collections. For example, methods for accessing records of a collection, checking the validity of a record before an insert or update operation, as well as methods required for defining a collection's behaviour.

Each collection must have at least one instance within the data store view. Such an instance is called an exemplar for a record of a collection. The exemplar used for records in any collection is by default a subclass of a standard Magik record class for the type collection. Hence, the Magik record classes involved in creating a record exemplar for a collection are: ds_record, view_record, extdb_record, rwo_record, and dataless_rwo_record. The hierarchical structure of these classes in a Magik image is illustrated in Figure 7.48.

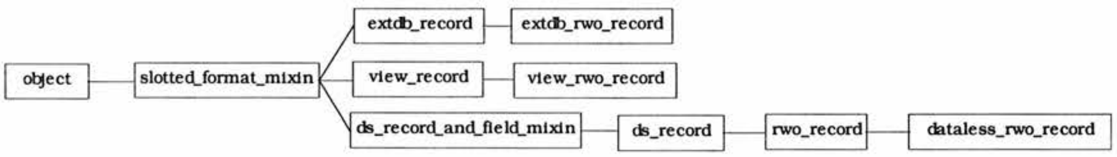


Figure 7.48 - Record Classes used for creating an exemplar for a collection

Methods can be invoked on a record of a collection in order to perform some operations. In other words, methods can be attached to a RWO to carry out operations over the RWO. They are named Trigger Methods which invoke operations for inserting a new record, updating (modifying) or deleting a record.

At the next abstraction level, a record consists of a set of different fields. According to the kind of the field presented in a record, different storage parameters, constraints, and special behaviour of editors and reports, are attached for each field. All kinds of a field inherit from the dd_basic_field class in a Magik image (Figure 7.49). However, this class is an abstract one and only its subclasses are actually used.

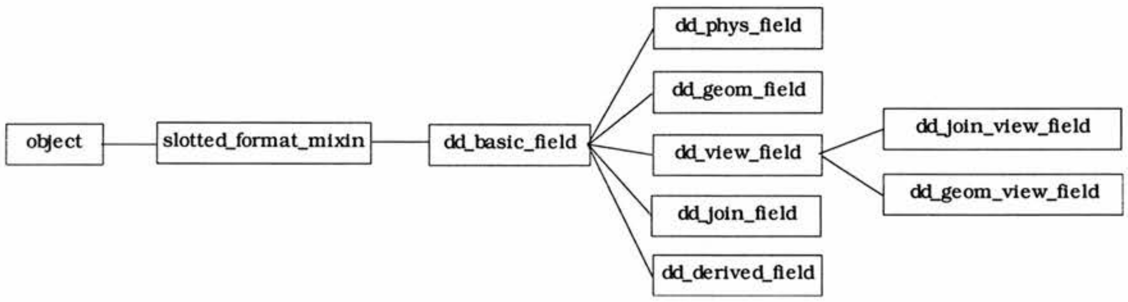


Figure 7.49 - Class hierarchy of kinds of fields

Essentially, the kinds of fields available in the system are:

- dd_phys_field, for physical fields in a collection;

- dd_geom_field, for representing a geometry attribute of a RWO;

- dd_derived_field, for logical fields derived by a special method;

- dd_view_field, for deriving physical or derived fields in a view_collection;

- dd_join_field, for logical fields where the method is constructed in a special way from a join;

- dd_join_view_field, for fields in a view_collection which are derived from join fields;

- dd_geom_view_field, for referencing a geometry field in a collection.

Moreover, a field can be distinguished by its type. The type of a field identifies how it is stored in the VMDS. Some example of the basic types for a field are:

:ds_byte	:ds_byte_vec
:ds_short	:ds_short_vec
:ds_int	:ds_int_vec
:ds_float	:ds_float_vec
:ds_coord	:ds_coord_vec
:gis_id	(ds_uint_vec for geometry keys)
:gis_aid	(ds_uint_vec for spatial keys)
:rwo_id	(ds_uint_vec for rwo keys)
:sys_id	(ds_uint for system generated keys on rwos)

Methods are invoked on a field in the record. For alphanumeric fields, the only possibility is a trigger method for updating the value of the field. On the other hand, for a geometry field, different trigger methods can be invoked. Attach trigger methods can be invoked after geometry is associated with a record, meanwhile update trigger methods are invoked after a geometry field is updated. In addition, detach trigger methods can be invoked when a geometry is removed from a record in such a way that the field value becomes unset.

The geometry for a particular RWO is actually stored in a separate table within the VMDS. The connection from a RWO record to its geometry is obtained through a join, which provides transparent access between tables. The relationship is defined as having a 0:0 cardinality in the data model. This signifies that there can be at most one geometry of a particular type for a particular RWO record. In other words, at most one RWO can have a particular type of geometry (Figure 7.50).

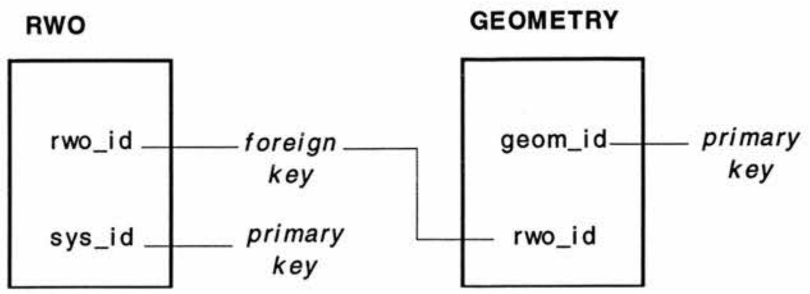


Figure 7.50 - The relationship between a RWO and its geometry

The geom_id is a primary key which is physically encoded to contain information about the geometry space in which an RWO is found. It contains five components as follows:

1. the universe_id
2. the world_id in the universe
3. the ordered priority in the world
4. the spatial tag
5. the unique key

The maximum number of universes which can be supported by the VMDS is defined at the creation of the database. The number of universes determines how many bits on the spatial tag are used to store the universe_id. A spatial tag is based on an overlapping quadtree structure.

Worlds are subsets of a universe, each world representing a different drawing representation of a universe. The number of worlds must be a power of two. The number of bits reserved for world_id, priority, and spatial tag can be completely different within the VMDS. There is a gross control over the order in which geometry will be retrieved during a session. The ordered priority in world gives the magnitude of the priority number that controls the order in which RWOs are drawn. The choice of priority is determined internally on the basis of the rwo_code (the number which identifies the owning rwo table) and app_code (the number which identifies which sort of geometry it is) of the geometry.

All geometry acquires a record class which is inherited from the top_level_geometry class in a Magik image (Figure 7.51). There are different descriptions for defining a geometry for a RWO. These are:

- topologically structured: area, chain, point

- non-topologically structured: simple_area, simple_chain, simple_point

- text and dimensions

- grid_tile (for rasters)

- grids and tins.

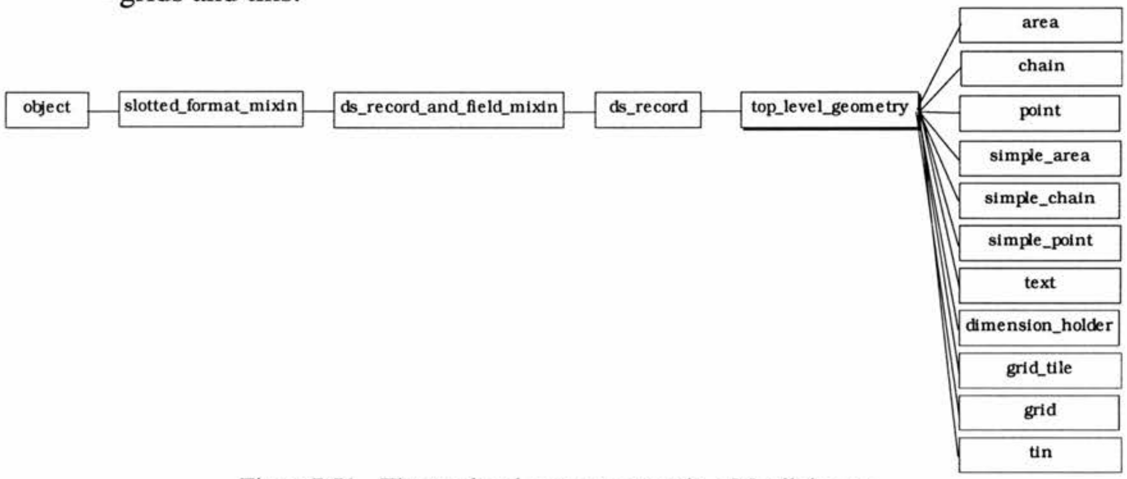


Figure7.51 - The top level geometry root in a Magik image

The geometry model in Smallworld GIS is based on the exemplar names for the geometry classes as illustrated in Figure 7.52. It presents a top level structure from which a hierarchy of lower objects are built up for creating the geometry at the coordinate level.

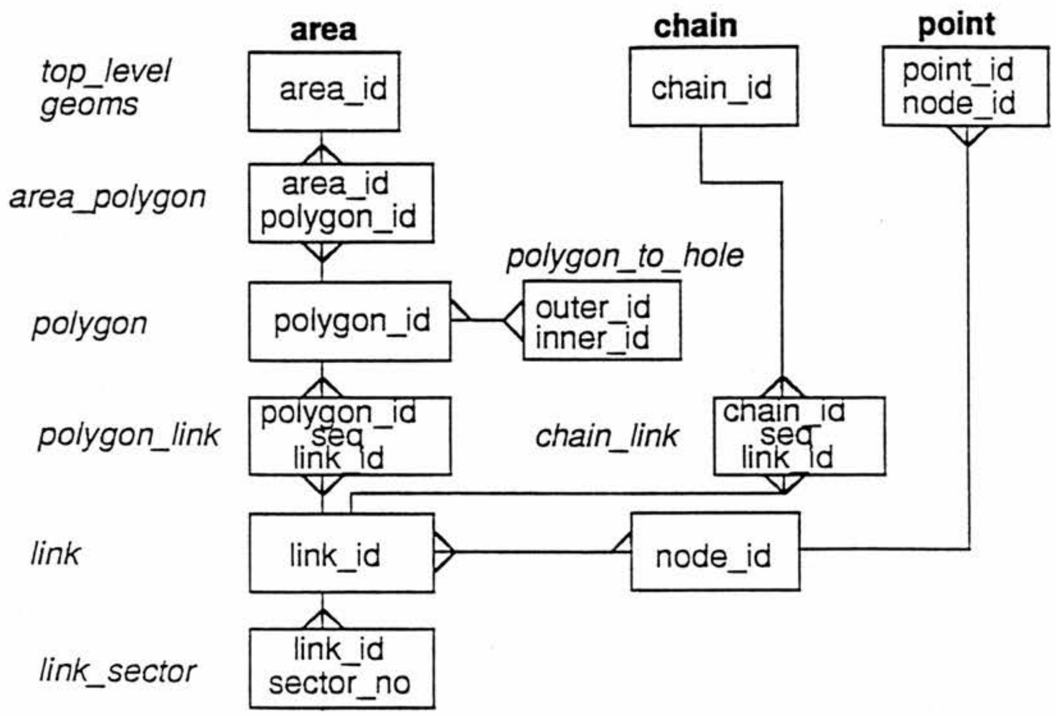


Figure 7.52 - The Geometry Model of Smallworld GIS
(Source: Smallworld GIS 2 Customisation Manual, p.12)

Having described the main features of the Smallworld GIS, the implementation aspects of incorporating the spatio-temporal data model into the Smallworld GIS will be discussed in the next section.

7.5 Implementing the Spatio-Temporal Data Model in the Smallworld GIS

How are applications developed in Smallworld GIS? There are probably as many answers to this question as there are Smallworld users. However, the implementation of the spatio-temporal data model required a number of critical steps to be followed in developing the application. In particular, these are related to configuring the VMDS, specifying the schema for implementing the spatio-temporal data model, and managing the versions within the system.

7.5.1 The VMDS Configuration

In the spatio-temporal data model, states (evolutionary and revolutionary) and events have been modelled as classes of objects. Considering the Smallworld structural abstractions, states have been implemented as Real World Objects (RWOs) within the system. They contain information about the different states of a public boundary by holding its spatial and non-spatial properties. The type of collection chosen for the RWOs has been a ds_collection.

The main adaptation is the definition of a Real World Event (RWE) within the system. It is itself a ds_collection as a RWO but it holds temporal properties about events instead of spatial and non-spatial properties (Figure 7.53).

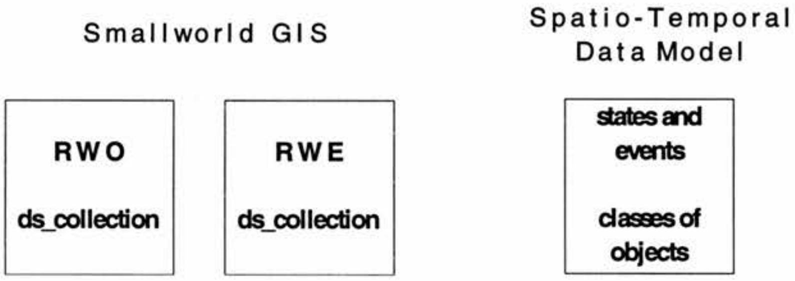


Figure 7.53 - RWO and RWE representations

Each RWO and RWE has an object identifier (sys_id) in order to identify uniquely any state or event independently of the values that they contain (Figure 7.54). Although the Smallworld GIS allows direct access to identifiers, they are not visible to the end user for purposes of clarity, in this implementation.

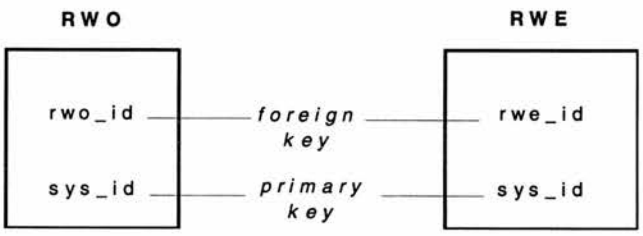


Figure 7.54- The object identifiers in a RWO and RWE

Both rwo_id and rwe_id are foreign keys which are physically encoded to respectively locate a RWO in space and a RWE in time. The Magik image and the data store view of the Smallworld GIS are already aware of what a rwo_id is for. Its main purpose is to index a RWO in a way that produces clustering naturally. This is achieved by configuring the rwo_id into components called the universe_id, the world_id, the priority, and the spatial tag. These components have been described in the previous sections.

The challenge involves the implementation of a rwe_id in the Smallworld GIS. Since the spatial component comes last in the rwo_id, one possibility is to set up the world component in a way that it could be used to model time instead. This entails the same indexing technique being used for both space (rwo_id) and time (rwe_id) in order to incorporate space-time paths within the Smallworld GIS. This procedure has been taken for investigating if indexing time and space in the same way is actually appropriate for implementing a space-time path.

The dimensionality configuration of space and time in the spatio-temporal data model have been established as (X,Y) coordinates for RWOs and a (valid-time) coordinate for RWEs. Coordinates are stored as 32 bit integers in Smallworld GIS. This implies a maximum 'visible world' of (2^{31},2^{31}-1) or (2,147,483,648 ; 2,147,483,647). In fact, a visible world is the maximum area which can be seen by a user. Its size depends on which units have been chosen by a user for a world. For example, with the centimetre chosen as the database unit, the maximum world size would be 4,294,967,296 cm^2. In the case of the time dimension, if day has been chosen as the database unit, this would imply a maximum range period for the world of 2,147,483,648 days, not considering the case of leap years. As a result, the world concept in the Smallworld GIS will be the extent of an area where RWOs can be found, and the period of time when RWEs can occur.

Therefore, the implementation decision was taken to adopt a partitioning scheme which subdivides data on space and time in order to define how RWOs and RWEs are clustered in the B*-tree structure of Smallworld. The implementation route was using the world component of both rwo_id and rwe_id and incorporating them as space and time dimensions in each quadtree level. The results are illustrated in Table 7.9 in which kilometre (km) and year have been used as the measurement units for the world. This table shows the relation between quadtree levels, the required bits and the length of the 'visible world'.

Table 7.9 - Proposed 'visible worlds' for a spatio-temporal database

quadtree levels	required bits	length of the 'visible world' (database size)	
1	3	2,147 km	5,965,232 years
2	5	1,074 km	2,982,616 years
3	7	537 km	1,491,308 years
4	10	268 km	745,654 years
5	12	134 km	372,827 years
6	14	67 km	186,413 years
7	17	33.6 km	93,206 years
8	19	16.8 km	46,603 years
9	21	8.39 km	23,301 years
10	24	4.19 km	11,650 years

However, the results of implementing the time and space dimensions in the same way have raised some problems. The C spatial scanner of the Smallworld GIS is aware of the presence of the components of a rwo_id and rwe_id, which are universe, world, and priority, and it only allows you to specify ranges of values for the coordinates. This implies that both RWO and RWE *must* have two coordinates in the system, X and Y for a RWO, and for example, valid and transaction time for a RWE. In the spatio-temporal data model, only valid changes have been considered, and therefore, a RWE is deemed to handle the valid time dimension *only*.

The results found in Table 7.9 have also demonstrated that the way a 'visible world' is organised and represented has important consequences for the storage and representation of RWOs and RWEs. In the Smallworld GIS, a representation of a 'visible world' has two main components:

1. A scheme for partitioning the entire visible world into cells (quadtree levels);

2. A one-way mapping that relates RWOs to the cells that inhabit them.

After implementing the spatio-temporal data model, the representation of a 'visible world' has added one more component:

3. A one-way mapping that relates RWEs to the cells that inhabit them.

The consequence of this implementation is that the concepts of space and time are associated to 'layers' or 'themes'. In the spatio-temporal data model , there is no correspondence between layers and space-time paths. In fact, space-time paths emphasise the representation of *objects* in space over time rather than organising space and time into layers. Hence, the definite conclusion reached in this research is that the Smallworld GIS and any other GIS having the same type of system's structure, are more appropriate for implementing *bitemporal elements* in order to represent time. Bitemporal elements portray the concept of a 'temporal layer' in which the cells represent the association between valid time and transaction time dimensions.

In recognition of this fact, the implementation of the space-time path might require the development of new data structures in GIS systems. Current emphasis on using comparative search trees (such as a quadtree structure) in GIS systems, is certainly influenced by the great success they had as a solution for earlier applications. However, new data structures might be required for improving indexing techniques in a way that they represent the space **and** time dimensions of space-time paths in the spatio-temporal data model. Multidimensional indexes may appear in commercial GIS products. They will provide a means for data partitioning in databases in a way that the data of multiple dimensions are indexed into a single value which represents the interaction of all dimensions. This is probably a more appropriate way for implementing space-time paths in a GIS because it will allow a two-way mapping which relates RWOs and RWEs to a partition of space, and vice-versa. Some research is being carried out about new indexing techniques such as [Kolovson 1990, Schneider and Kriegel 1992, Varma 1994].

The final implementation decision was to use timestamp fields for the temporal properties of a RWE as previously mentioned in Chapter 6. The Smallworld GIS provides ds_time, ds_date, and ds_time_date types which can be attached to a field of a RWE. Both nominal (for example, today, May 27) and ratio types (for example, event A starts at 8 and ends at 12) are needed as timestamps for a RWE according to the spatio-temporal data model. However, the Smallworld GIS supports only nominal timestamps.

7.5.2 The Schema Specification for Implementing the Spatio-Temporal Data Model

Excepting the metaclass definition, the whole definition of classes of objects and their properties in the spatio-temporal data model has been implemented in Smallworld GIS using the Case Tool application. Appendix D is the report obtained from this implementation.

In the spatio-temporal data model, the relationships between classes of objects are deemed to be of three different types: aggregation (has), association, and inheritance. These have different cardinalities, such as 1:1 and 1:n. Unfortunately, these relationships can only be implemented by using *join* relations between ds_collections in the Smallworld GIS. Figure 7.55 illustrates the general view of such an implementation constraint imposed by the system configuration of the Smallworld GIS.

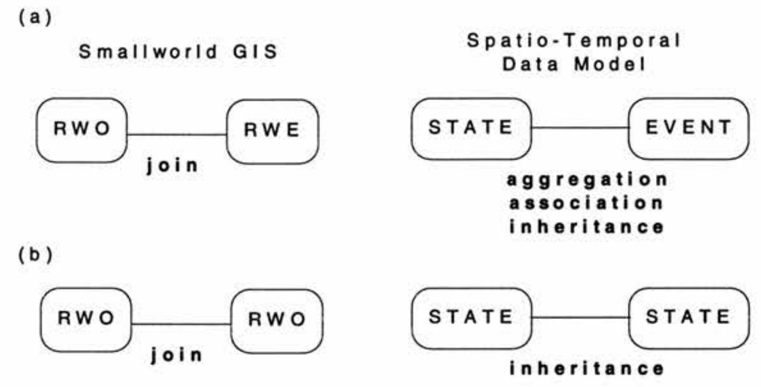


Figure 7.55 - Implementation of the relationships
(a) existence circumstance of a space-time path
(b) the mutation circumstance of a space-time path

The Smallworld GIS supports three possibilities in defining join relationships. They are:

- dd_join: the direct relationship between two tables;

- dd_join_via: the join via one intermediate table;

- dd_conditional_join: a collection of joins, which one field is used depending on the value of the another field.

These joins have been used to establish the relationships between RWOs and RWEs. Conceptually, this represents the creation of the spatio-temporal relationships among states and events within the Smallworld GIS. The join relationships between RWOs and RWEs will determine the orientation taken by a space-time path.

When a join relationship is defined in the Smallworld GIS, the methods which will be used by the system to retrieve the objects are also defined. They define the operations which allow a user to navigate in a space-time path by moving among RWOs and RWEs. They have been used to implement the Historical Views of space-time paths of the spatio-temporal data model. Historical Views are illustrated in the next chapter.

The cardinality of a join relationship plays an important role in defining how RWOs can be connected to other RWEs. It distinguishes between independent and dependent objects in every join relationship in the system. An independent object can exist in the system without being connected to a join relationship. In contrast, a dependent object only exists if there exists a join relationship connected to it.

Implementing the space-time paths between RWOs and RWEs has required two different types of cardinality available in the Smallworld GIS: 1:n and 0:n. The value 0 used for the cardinality in Smallworld means 0 and 1. The cardinality 1:n between a RWE and a RWO establishes that the RWE is the independent object whilst the RWO is the dependent object. This implies that each RWO must be associated with one and only one RWE. On the other hand, a RWE can have one or several RWOs connected to it (Figure 7.56).

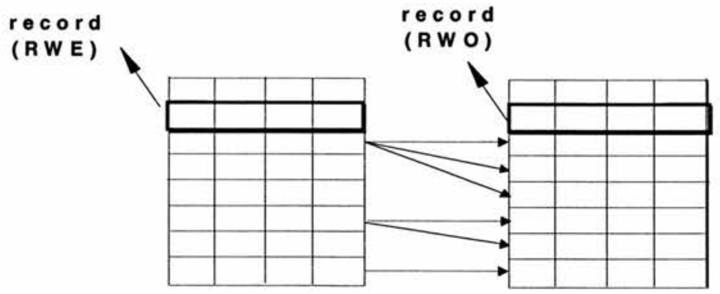


Figure 7.56 - A join relationship with 1:n cardinality

The join relationship with 1:n cardinality has also been employed for implementing the relationship between RWOs. In this case, such a join relationship is responsible for associating an evolutionary state (RWO) with its revolutionary state (another RWO). The revolutionary state is then the dependent object, meanwhile the evolutionary state is the independent one.

The cardinality 0:n of a join relationship is very interesting in terms of the implementation of the spatio-temporal data model because it allows different RWEs to occur over the same RWO. Figure 7.57 illustrates this relationship, where RWEs are the dependent objects whilst RWOs are the independent ones.

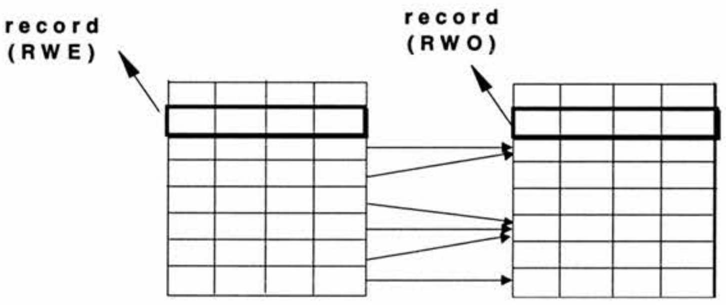


Figure 7.57 - A join relationship with 0:n cardinality

Some examples of implementing these join relationships are:

NewBoundary	
Superstructure	: 1:n to delimitation
Substructure	: 1:n with demarcation
GroundFeatureRevolutionaryState	
Superstructure	: 1:n to groundfeature
Substructure	: 0:n with perambulation

Appendix D provides the overall description of RWOs and RWEs according to the join relationships which have been implemented in the Smallworld GIS. In the next chapter, the interaction between RWOs and RWEs through their join relationships is illustrated.

7.5.3 Version References and Configuration

The version management mechanism developed for the spatio-temporal data model implies versioning at the instance level of versioned classes, as well as the classification of their attributes and version states. Basically, these are the main elements in the temporal data management of the spatio-temporal data model which need to be implemented.

Versioning at the instance level has been implemented in the Smallworld GIS. It uses the available structure of records of a ds_collection in the data model of the Smallworld GIS. As a result, a version is implemented as a ds_record and methods can be invoked on this ds_record in order to perform the Promote and Derive operations (See Chapter 6, p.139). The visualisation of versions is achieved by the use of 'pop up' windows which display a version at the instance level of a versioned class. This is illustrated in the next chapter.

In relation to the attribute classification of versioned classes, all types of attributes (version significant, non-version significant and invariant attributes) have been incorporated into the Smallworld GIS. The type of non-version significant attributes is already available in the system. They are attributes whose values are updated in a destructive manner.

Invariant attributes have been incorporated into the Smallworld GIS by attaching an update constraint to the ds_fields of the OldBoundaryRevolutionaryState, GroundFeatureRevolutionaryState and OldBoundary classes (See Table 6.7, p.136). This is achieved by defining a trigger method for each ds_field in a way that its value is not allowed to be updated. This method is triggered every time a user tries to update invariant attributes.

Version significant attributes have also been implemented by attaching an update constraint to the fields of OldBoundary and GroundFeature classes. This is achieved by defining trigger update methods for each field of these versioned classes. If a user tries to update a version-significant attribute, a menu pops up on the screen informing the user to update the version significant attribute in a non-destructive manner (See Appendix D for Magik code).

Moreover, the versioned, stable, and transient versions states have been incorporated into the Smallworld GIS. The drawback found in the implementation was related to the interface between the structure of the versioning mechanism of Smallworld and the one proposed by the spatio-temporal data model.

In the spatio-temporal data model, the versioning mechanism is based on the Backward Oriented Accumulative strategy. Reference attributes have been used to implement a successor-in-time version (i.e. ds_record) in a way that the relationship between two versions has been implemented through the foreign keys of the ds_records. It has proved to be very helpful in creating versioned, stable, and transient versions for the valid changes. The main advantage is that a successor-in-time version is created only if an update procedure is triggered in a ds_field that belongs to the ds_record (See Chapter 8 for examples in terms of public boundaries). In terms of performance during the browsing of the which already exist versions (i.e. ds_records), increase in the access times to the older versions was not apparent. However, further studies need to be carried out in order to quantify the performance timings.

On the other hand, the versioning structure of the Smallworld GIS is deemed to manage long transactions of the VMDS. This kind of structure is related to the transaction time as being the time when the updated data are stored in the VMDS. In fact, it is simply the time measured by a system clock, when an update occurs.

Therefore, the major finding in this part of the investigation is the incompatibility between these two versioning mechanisms. In the Smallworld GIS, transaction time is a value which can not be specified by a user or derived from the model. In the spatio-temporal data model, valid time is specified by the user. These are specific semantics which make it impossible to use the same versioning mechanism in order to treat transaction and valid time uniformly. The consequence of this point of view is that bitemporal changes and valid changes will require different approaches for developing an appropriate version management mechanism for each case.

7.6 Conclusions

One of the major limitations that has emerged in implementing the spatio-temporal data model is the requirement that relationships can only be created using join relationships between RWOs and RWEs within the Smallworld GIS. The primary organisation in the VMDS is a table (ds_collection), and unfortunately, this implies that relationships such as inheritance, can not be implemented within the system. A tabular database managed by an object-oriented programming environment seems to have its own limitations in incorporating basic concepts of object-orientation.

Methods defined in a ds_view have also shown certain limitations. They can be defined as trigger methods of three types: update, insert, and delete. The actual polymorphism concept of object-orientation is unavailable when using ds_collections in the VMDS. The object-oriented environment available in a Magik image was essential for manipulating the application within the Smallworld GIS. This is a complex environment, many of whose features lack appropriate documentation, so it has a long learning curve.

This chapter has shown the results of implementing the spatio-temporal data model into the Smallworld GIS. These results have provided the evidence that the object representation of the spatio-temporal data model can be implemented and tested within an integrated environment, such as a GIS. The union of time-geography and the object-orientation paradigm within a vehicle of a GIS system, has proved to be achievable in principle although the available technology at present has a number of limitations.

In the next chapter, an example application is presented in order to illustrate more fully the implementation of the spatio-temporal data model in the Smallworld GIS.

Chapter 8

Implementation of the Spatio-Temporal Data Model

8.1 Introduction

In this chapter, an example application is presented in order to demonstrate the most important aspects of the implementation of the spatio-temporal data model. The four basic scenarios, which are the Public Boundary Entry Scenario, the Evolution Tracking Scenario, the Update Scenario, and the Archiving Scenario, have been used to illustrate the application. The execution of these scenarios provides an understanding of the execution of the application in general. The overall goal is to show how space-time paths can be created within the Smallworld GIS.

Some sample datasets of the East Cambridgeshire region have been used to illustrate the 'boundary making' process. These datasets are provided from within the Smallworld GIS. The 'boundary making' examples are fictitious as the aim is mainly to create a prototype implementation rather than build an application system at this stage.

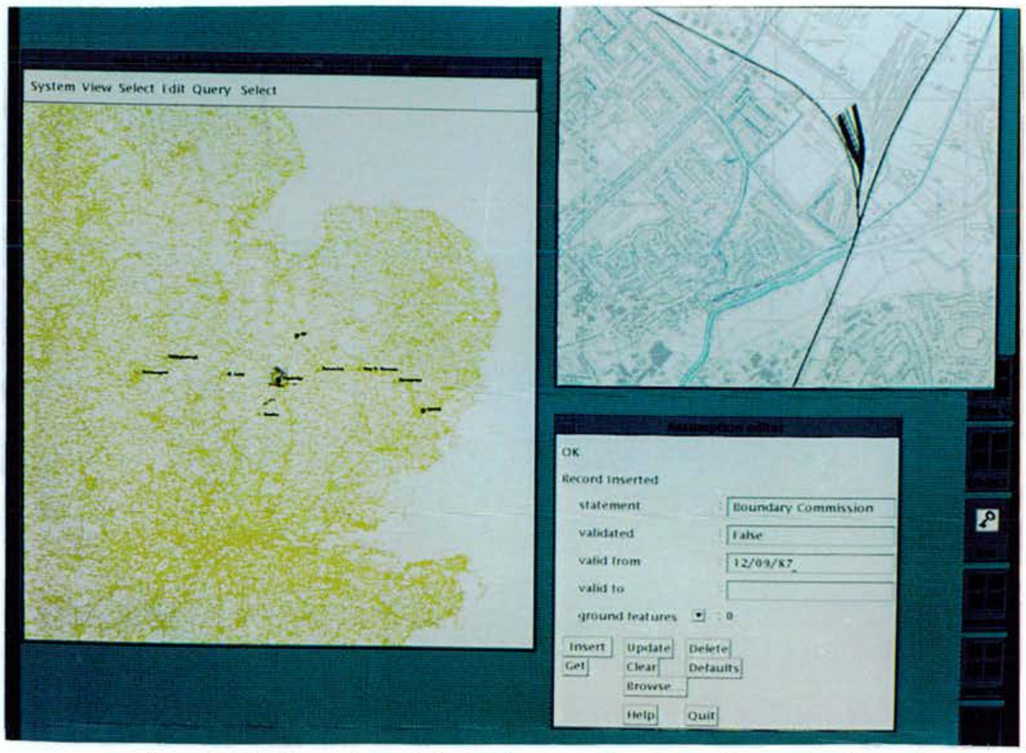
8.2 The Public Boundary Entry Scenario

The Public Boundary Entry Scenario represents the evolutionary states and events which are concerned with the allocation process within the system. This means that the user will select ground features, based on some assumptions, to be a future public boundary. The final allocation decision will be taken by the user who will assign a ground feature to be a draft boundary.

The execution of such a scenario has been divided into four stages which are illustrated in the following figures of this section. The first stage focuses on the creation circumstance of a space-time path of a public boundary. It represents the origin of a space-time path within the Smallworld GIS.

8.2.1 The Creation Circumstance of a Space-Time Path - Stage 1

In our example, the origin has been attached to the Assumption event of the spatio-temporal data model. The user has made a statement based on, for example, a Boundary Commission demand that determines that a possible future public boundary will be created in a certain region. Figure 8.58 illustrates this situation in which the Assumption Editor menu (at the bottom right side of the figure) represents the RWE (Real-World Event) involved in the creation of a space-time path. It shows the properties previously defined for the Assumption class of the spatio-temporal data model: statement, validated, and valid from-to. In this example, the Boundary Commission demand for creating a public boundary has not been validated by an Order or Act. This demand has been valid since September 12th, 1987.



OK

Record inserted

statement	Boundary Commission
validated	False
valid from	12/09/87
valid to	
ground features	0

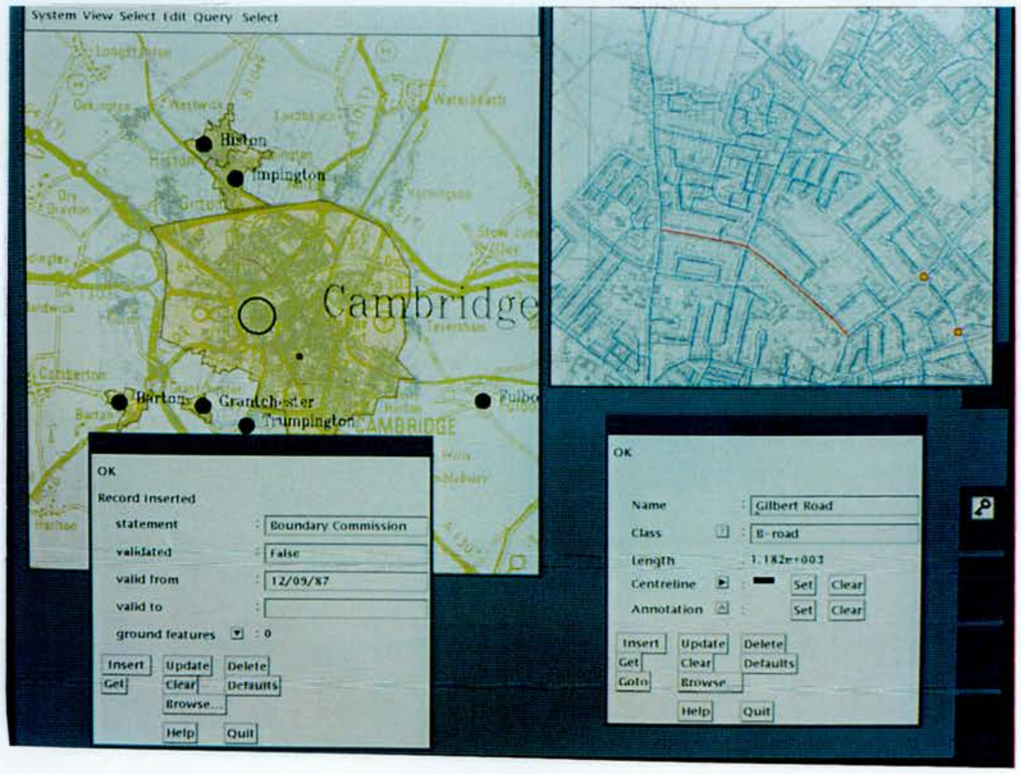
Figure 8.58 - Execution of the Public Boundary Entry Scenario - Stage 1

The user can select any RWO that belongs to the GroundFeature, DraftBoundary, NewBoundary, or OldBoundary classes or any RWE that belongs to the Assumption, Allocation, Delimitation, Demarcation classes. The origin of a space-time path in the VMDS can never be modified. In this example, the origin of a space-time path is created once the properties of the Assumption class have been inserted in the ds_view of the VMDS. This is the case, even though the space-time path does not exist at this stage. In other words, the join relationship between the RWE (i.e. the ds_record of the Assumption class) and the RWO (i.e. the ds_record of the GroundFeature class) has not yet been generated in the Smallworld GIS. Therefore, the `ground features` element in the Assumption Editor menu (at the bottom right side of Figure 8.58) shows the number 0.

8.2.2 The Existence Circumstance of a Space-Time Path The Allocation Process - Stages 2 and 3

The second stage illustrates an example of an existence circumstance of a space-time path having as its origin the RWE previously created in Stage 1. In this example, the user is searching for the ground feature designated by the Boundary Commission, for example, Gilbert Road, which is deemed to belong to the proposed public boundary. This is accomplished by querying where Gilbert Road is located by using the Edit Menu provided by the Smallworld GIS.

In Figure 8.59, the Edit menu (bottom right) shows the properties which belong to Gilbert Road. In addition, the Application menu (top right) highlights where this road is located.



OK
Record inserted

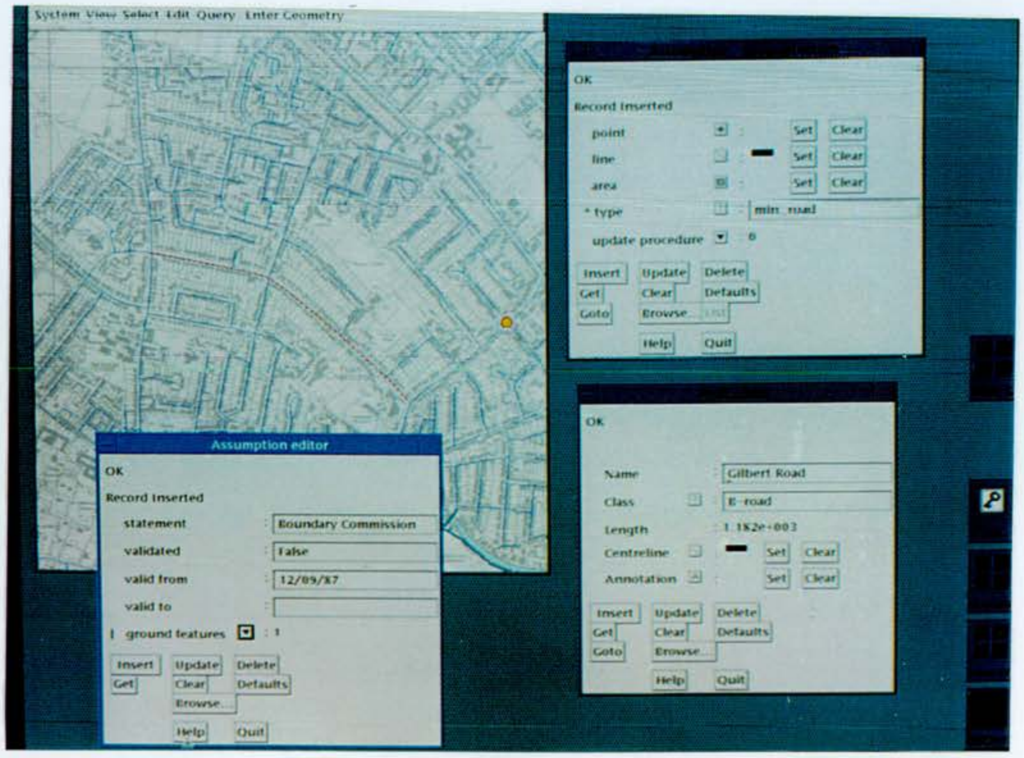
statement	Boundary Commission
validated	False
valid from	12/09/87
valid to	
ground features	0

OK

Name	Gilbert Road
Class	B-road
Length	1.182e+003
Centreline	
Annotation	

Figure 8.59 - Execution of the Public Boundary Entry Scenario - Stage 2

Once Gilbert Road has been identified on the screen, the user can then make the assumption that this road can be a public boundary as a result of the Boundary Commission demand. This is achieved by clicking on the arrow beside the `ground features` element in the Assumption Editor menu (bottom left of Figure 8.59) which will activate the Assumption - Ground Feature Editor menu (top right of Figure 8.60).



OK
Record inserted

point		Set Clear
line	▬	Set Clear
area		Set Clear
* type	min road	
update procedure	0	

OK

Name	Gilbert Road
Class	E-road
Length	1.1K2e+003
Centreline	▬ Set Clear
Annotation	Set Clear

Assumption editor

OK
Record inserted

statement	Boundary Commission
validated	False
valid from	12/09/X7
valid to	
ground features	1

Figure 8.60 - Execution of the Public Boundary Entry Scenario - Stage 3

In fact, the activated Assumption - Ground Feature menu (top right of Figure 8.60) is the transient version state of a ds_record of the GroundFeature class. It shows the properties of the GroundFeature class: point, line, area, and type. This transient version state will become a stable version state once the Derive operation is invoked in the spatio-temporal data model. Such an operation derives a stable version state from the given transient version state (Figure 8.61).

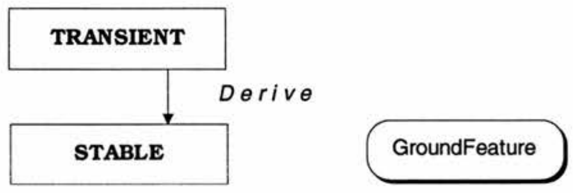


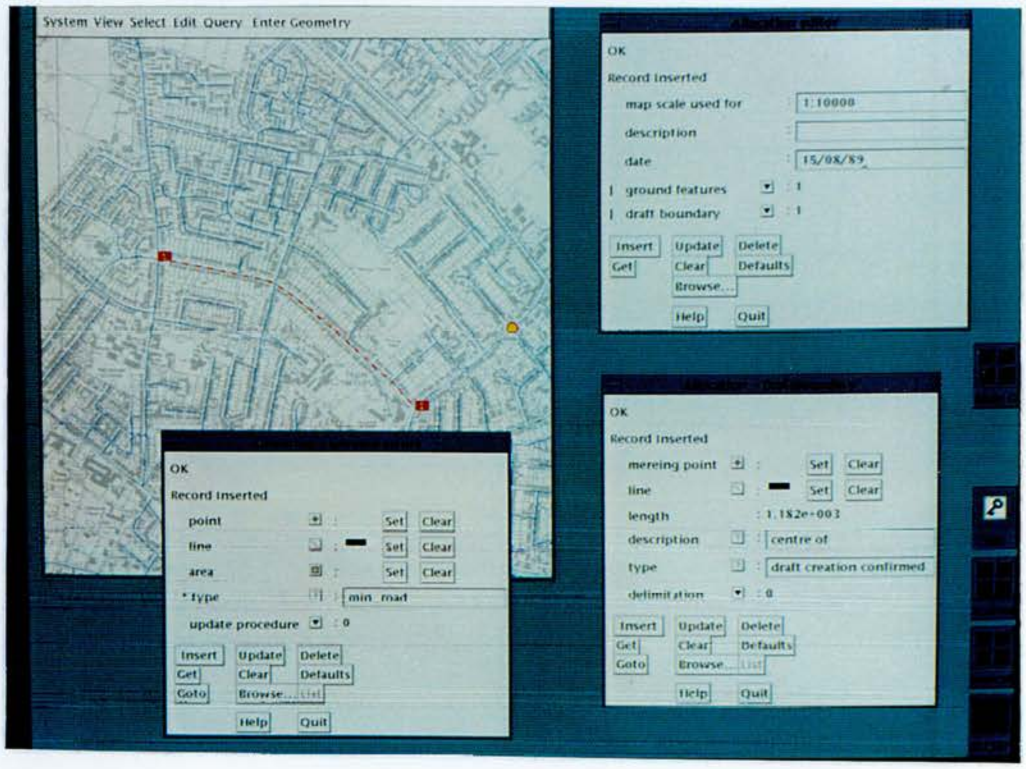
Figure 8.61- Creation of a new version of the GroundFeature class

The Derive operation is invoked when the user assigns the values of the properties of the GroundFeature class and inserts them into the VMDS by clicking on the `Insert` element of the Assumption - Ground Feature menu (top right side of Figure 8.60). A stable version state is then created in the Smallworld GIS. After creating the stable version state of a ds_record of the GroundFeature class, the `ground features` element in the Assumption Editor menu (bottom left side of Figure 8.60) is updated to the number 1, telling the user that one stable version of a ground feature has been created for the specific assumption. In other words, there exists a space-time path between the instances of the Assumption and GroundFeature classes. In terms of the spatio-temporal data model, there exists a join relationship between the ds_records of both Assumption and GroundFeature classes.

This implies that the version-significant attributes (point, line, and area) of this stable version can only be updated in a non destructive manner (See Table 7.7, p.142). An update in one of these attributes pertaining to the GroundFeature class will create a validate version state which will be the new ds_record of the GroundFeatureRevolutionaryState class. The `update procedure` element of the Assumption - Ground Feature Editor menu (top right side of Figure 8.60) shows the number 0 which informs the user that no update procedure has been carried out. Update procedures are illustrated in the next section where the execution of the Update Scenario is discussed in more detail.

8.2.3 The Existence Circumstance of a Space-Time Path
The Allocation Process - Stage 4

In the next stage (Stage 4), the allocation decision is taken by the user, and therefore, the ground feature is selected to be a draft boundary. In terms of the spatio-temporal data model, the GroundFeature, Allocation and DraftBoundary classes are affected by this allocation decision. The execution of this scenario is accomplished by activating the Allocation Editor menu (top right side of Figure 8.62) which represents the Allocation class within the system. The user can provide information about the map scale used for the allocation, a textual description of such an event, and the date on which the allocation took place (Figure 8.62).



OK
Record inserted

map scale used for	1:10000
description	
date	15/08/89
ground features	1
draft boundary	1

OK
Record inserted

mereing point	
line	▬
length	1.182e+003
description	centre of
type	draft creation confirmed
delimitation	0

OK
Record inserted

point	
line	▬
area	
*type	min road
update procedure	0

Figure 8.62 - Execution of the Public Boundary Entry Scenario - Stage 4

The Allocation - Ground Feature menu (bottom left side of Figure 8.62) shows that the user has selected Gilbert Road to be a draft boundary. The Application menu (top left side of Figure 8.62) highlights where this road is located. The user can then assign this ground feature to be a draft boundary. This is achieved by clicking on the `draft boundary` element of the Allocation Editor menu (top right side of Figure 8.62) which will activate the Allocation - DraftBoundary Editor menu (bottom right side of Figure 8.62). The activated menu represents the transient version state of the ds_record of the DraftBoundary class. The stable version state is derived from it, once the user has inserted the attribute values into the VMDS by clicking on the `Insert` element of the Allocation - Draft Boundary menu (bottom right side of Figure 8.62). In this case, all attributes of the DraftBoundary class have been defined as non-version significant attributes in the spatio-temporal data model (See Table 7.7, p.142). They can be updated without creating a new version state.

The available elements of each of the menus displayed at this stage (Figure 8.62) are:

- the `ground features` element which shows the number 1 indicating that Gilbert Road has been selected to be a draft boundary;
- the `draft boundary` element which shows the number 1 indicating that Gilbert Road has been allocated to be a draft boundary;
- the `update procedure` element which shows the number 0 indicating that no update procedures have been carried out so far;
- the `delimitation element` which shows the number 0 indicating that the delimitation process has not yet occurred at this stage.

In demonstrating the execution of this Public Boundary Entry Scenario within the system, it is important to point out the constraints designed in the spatio-temporal data model which address the behaviour of the evolutionary states to generate more realistic operational Potential Path Spaces. For example, a draft boundary state will never be created if its corresponding ground feature state does not previously exist in the VMDS of the Smallworld GIS. In the same way, an allocation can not take place without the previous existence of a ground feature state. This has been achieved by defining update methods for the invariant and version-significant attributes of the classes of objects in the spatio-temporal data model as discussed in Chapter 6 (See Appendix D for the Magik code).

8.3 The Evolution Tracking Scenario

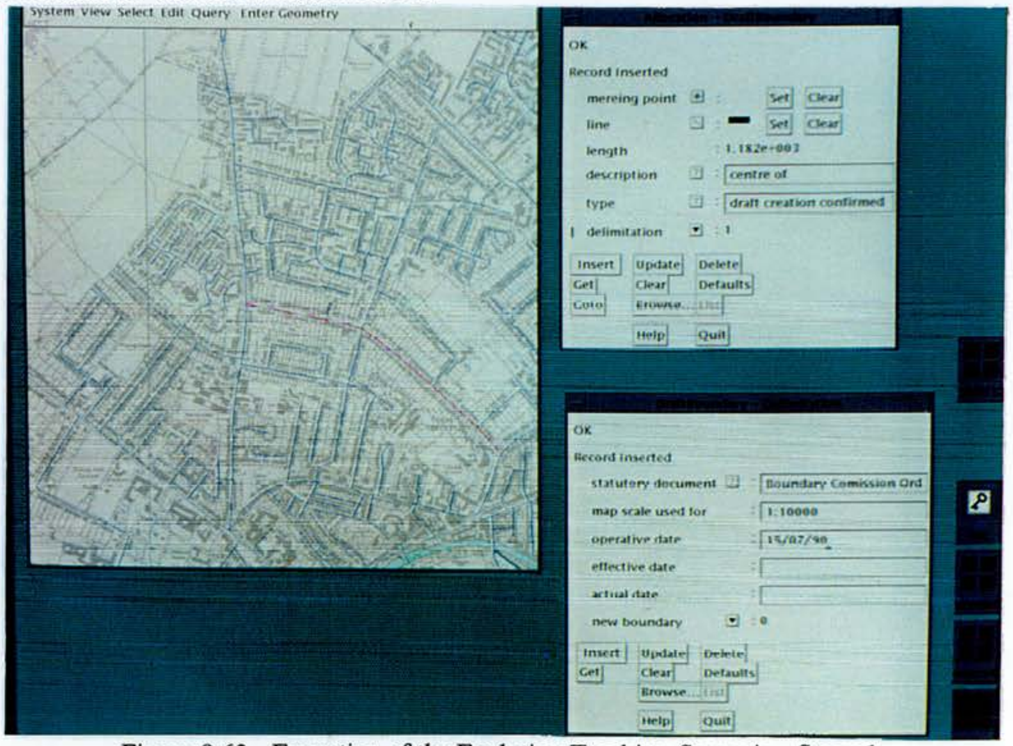
Both delimitation and demarcation processes are described in the Evolution Tracking Scenario. In the delimitation process, a draft boundary state is confirmed by an Act or Order, and therefore it assumes a new boundary evolutionary state. For the demarcation process, the position of this new boundary is ratified on the ground by surveyors, and afterwards, the old boundary state is assigned to the boundary.

The execution of this scenario has been divided into four stages. The DraftBoundary, Delimitation, NewBoundary, Demarcation and OldBoundary classes of the spatio-temporal data model are employed to illustrate our example. The focus is on demonstrating the independent incremental modification mechanism developed for the space-time path of the spatio-temporal data model. Since the inheritance mechanism is not supported by the datastore view of the Smallworld GIS, the independent incremental mechanism has been implemented through the join relationships between the classes. The purpose has been to illustrate how space-time paths could be handled by a user within a GIS.

8.3.1 The Existence Circumstance of a Space-Time Path The Delimitation Process - Stages 1 and 2

The first stage in this scenario focuses on the delimitation process for the existence circumstance of a space-time path. In our example, Gilbert Road is used to illustrate the delimitation process. At this point, the draft boundary state has been selected by the user as illustrated by the Allocation - Draft Boundary Editor menu (top right side of Figure 8.63). The draft boundary is highlighted in the Application Menu (left side of Figure 8.63).

The delimitation event is activated within the system by clicking on the `delimitation` element of the Allocation - Draft Boundary Editor menu (top right) as illustrated in Figure 8.63. The Draft Boundary - Delimitation Editor menu (bottom right) appears containing the relevant attributes which belong to the Delimitation class of the spatio-temporal data model. These include the statutory document and the map scale used for the delimitation, as well as the operative, effective, and actual dates related to the delimitation event.



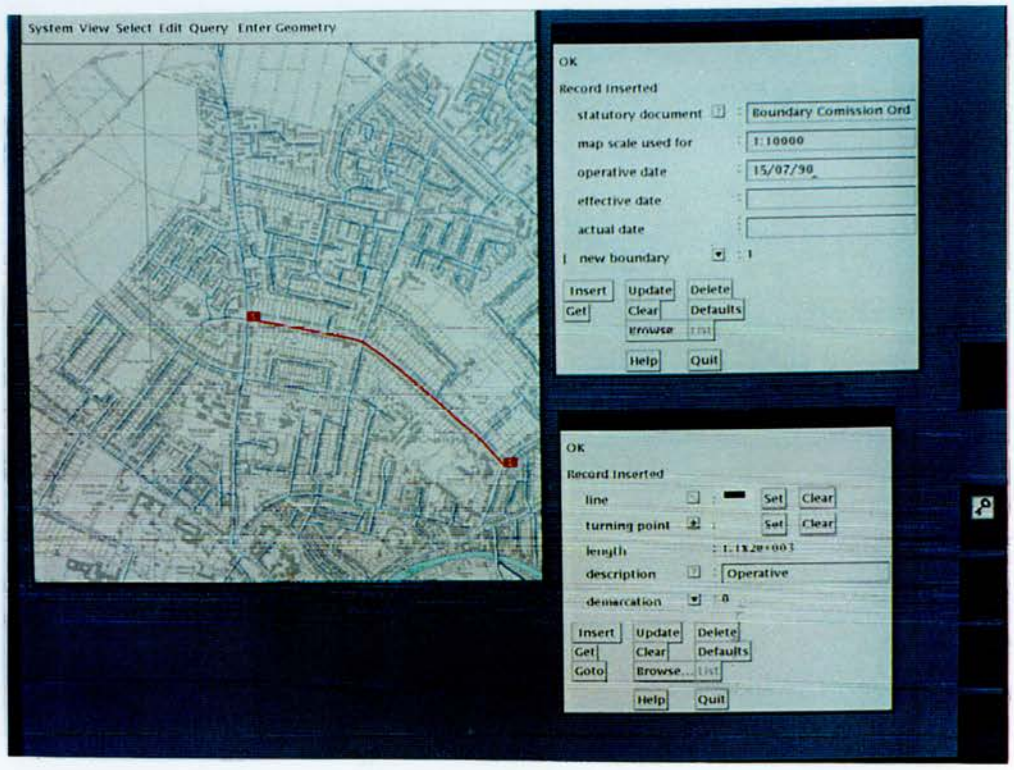
OK
Record inserted
mereing point :
line :
length : 1.182e+003
description : centre of
type : draft creation confirmed
delimitation : 1

OK
Record inserted
statutory document : Boundary Comission Ord
map scale used for : 1:10000
operative date : 15/07/90
effective date :
actual date :
new boundary : 0

Figure 8.63 - Execution of the Evolution Tracking Scenario - Stage 1

Once the attribute values of the Delimitation class have been inserted into the VMDS, the `delimitation` element of the Allocation - Draft Boundary Editor menu (top right side of Figure 8.63) shows the number 1. This number indicates as many delimitations as have been inserted by the user into the VMDS. On the other hand, the `new boundary` element of the Draft Boundary - Delimitation Editor menu (bottom right side of Figure 8.63) displays the number 0 since the new boundary state has not yet been created in the VMDS.

Once the user has decided to create the new boundary state in the system, he/she has to click on the new boundary element of the DraftBoundary - Delimitation menu (bottom right of Figure 8.63) in order to activate the Delimitation - New Boundary menu (bottom right of Figure 8.64). This activated menu represents the transient version state of the ds_record of the NewBoundary class.



OK
Record inserted
statutory document : Boundary Comission Ord
map scale used for : 1:10000
operative date : 15/07/90
effective date :
actual date :
new boundary : 1

OK
Record inserted
line :
turning point :
length : 1.1x2e+003
description : Operative
demarcation : 0

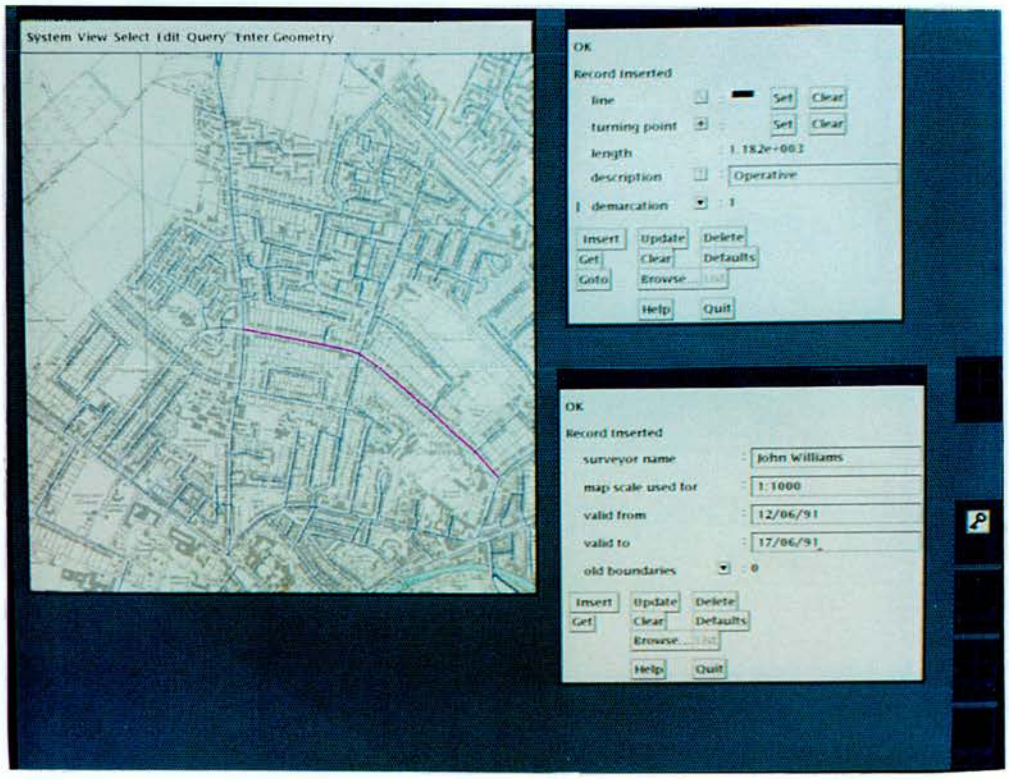
Figure 8.64 - Execution of the Evolution Tracking Scenario - Stage 2

As previously mentioned, the stable version state is created when the user inserts the values of the attributes of the NewBoundary class by clicking on the Insert element of the Delimitation - NewBoundary menu (bottom right of Figure 8.64). As a result the `new boundary` element of the Draft Boundary - Delimitation Editor menu (top right of Figure 8.64) is automatically updated to the number 1. The user is then aware of the existence of a new boundary state in the VMDS of the Smallworld GIS. In this case, all attributes belonging to this new boundary state have been defined as non-significant attributes within the spatio-temporal data model. Their updates do not create new versions in the system.

8.3.2 The Existence Circumstance of a Space-Time Path The Demarcation Process - Stages 3 and 4

The description of both Stages 3 and 4 has been devised in such a way that it avoids mentioning the conceptual elements behind the application. This has been important in demonstrating how simple and repetitive the stages are in building up space-time paths within the Smallworld GIS. In general, the user has only to follow the successor-in-time creation of different states within the system. A newer state can only be created if its corresponding older state in the space-time path already exists.

Stage 3 denotes when the demarcation event takes place within the system. This involves the creation of an old boundary state from its corresponding new boundary state. The user clicks on the `demarcation` element of the Delimitation - New Boundary Editor menu (bottom right side of Figure 8.64) activating the display of the NewBoundary - Demarcation Editor menu (bottom right side of Figure 8.65). This menu displays the relevant information about the demarcation event. This includes the properties defined for the Demarcation class which are the name of the surveyor in charge of carrying out the demarcation, the map scale used, and the period of time taken by the surveyor for demarcating the boundary on the ground.

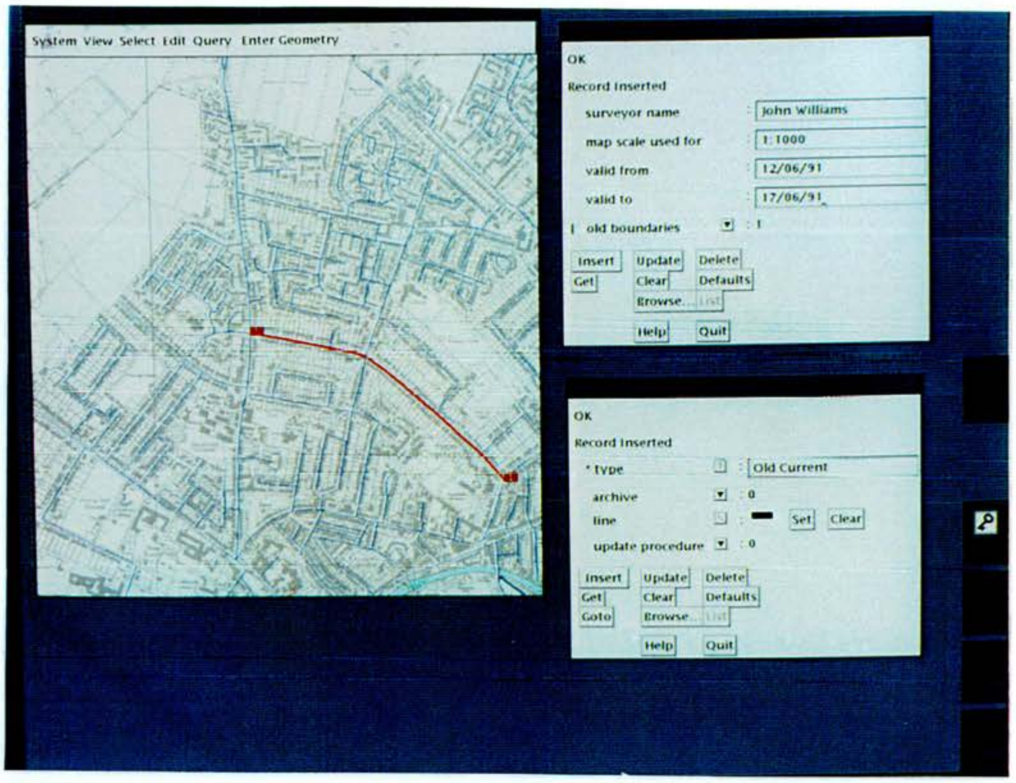


OK
Record Inserted
line :
turning point :
length : 1.182e+003
description : Operative
demarcation : 1

OK
Record Inserted
surveyor name : John Williams
map scale used for : 1:1000
valid from : 12/06/91
valid to : 17/06/91
old boundaries : 0

Figure 8.65 - Execution of the Evolution Tracking Scenario - Stage 3

At this stage, the old boundary state has not yet been created in the system. The `old boundaries` element in the NewBoundary - Demarcation menu shows the number 0. The user activates the Demarcation - Old Boundary Editor menu (bottom right of Figure 8.66) by clicking the `old boundaries` element in the New Boundary - Demarcation Editor menu (bottom right of Figure 8.65). Once the attributes related to an old boundary state have been inserted into the VMDS by the user, the `old boundaries` element in the New Boundary - Demarcation Editor menu (top right of Figure 8.66) shows the number 1.



OK
Record Inserted

surveyor name	John Williams
map scale used for	1:1000
valid from	12/06/91
valid to	17/06/91
old boundaries	1

OK
Record Inserted

* type	Old Current
archive	0
line	▬
update procedure	0

Figure 8.66 - Execution of the Evolution Tracking Scenario - Stage 4

The Demarcation - Old Boundary Editor menu (bottom right side of Figure 8.66) shows two important elements. First, the `archive` element which if activated will archive the selected old boundary. This is discussed in more detail in Section 8.5. Second, the `update procedure` element indicating that no update procedure has been carried out over the selected old boundary state. This is discussed in more detail in the following section.

Although NewBoundary and OldBoundary classes have different display scales as designed in the spatio-temporal data model, these have not been used in the illustrations for this section for purposes of clarity.

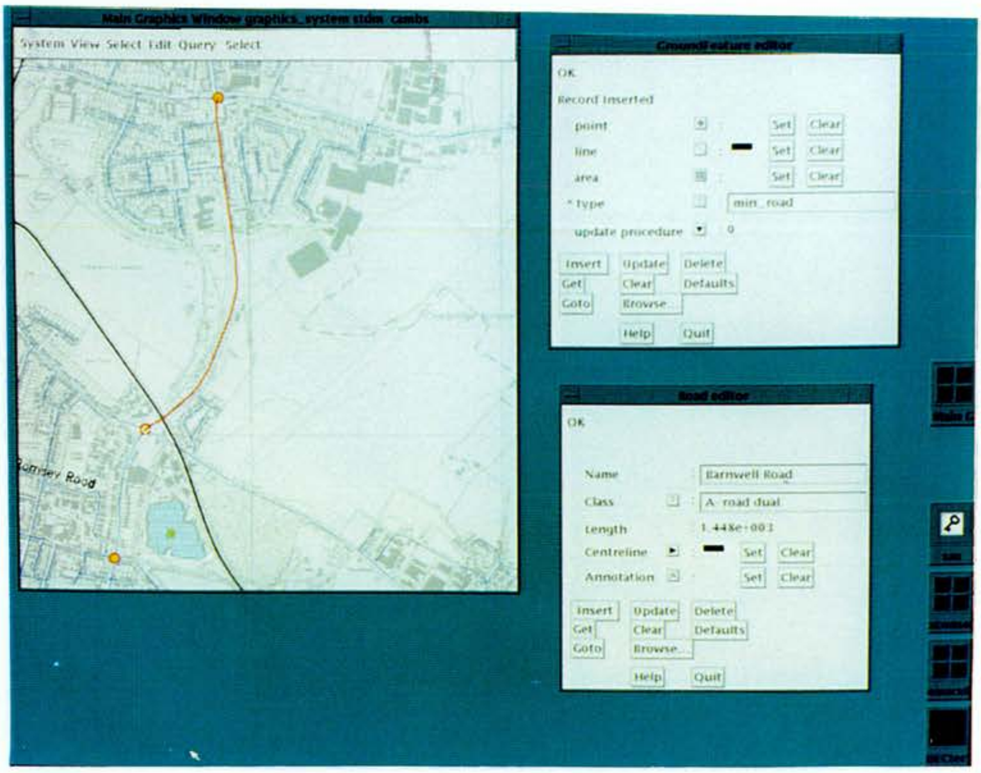
8.4 The Update Scenario

The Update Scenario handles the update procedures due to natural changes and new demarcation descriptions occurring over public boundaries. Three update procedures have been defined as 1) creation of a new object, 2) creation of a new object from a previously existing one, and finally, 3) the relocation of an existing object (See Chapter 5, p.105). In this scenario, the focus is on illustrating the main aspects involved in creating revolutionary states due to these update procedures, by using the overlapping incremental modification of the spatio-temporal data model.

The first update procedure has already been illustrated in Section 8.2, and it will not be discussed further. Stages 1 to 3 illustrate the stages involved in performing the second update procedure. And finally, Stages 4 and 5 illustrate the third update procedure of the Update Scenario.

8.4.1 The Mutation Circumstance of a Space-Time Path Creation of a New Object From a Previous Existing One - Stages 1 to 3

In this example, a ground feature has been selected by a user which is the Barnwell Road object, as shown in the Road Editor menu (bottom right side of Figure 8.67), and it is highlighted in the Application Menu (right side of Figure 8.67). The Ground Feature menu (top right side of Figure 8.67) shows the instance of the GroundFeature class which represents Barnwell Road. In fact, this represents the stable version state of the ds_record of this class which is about to be updated. The update procedure involved is related to the creation of a new version of the GroundFeatureRevolutionaryState class from its previous version GroundFeature class. The operation involved is named Promote which signifies that valid changes occur over a version state (Figure 8.68).



OK

Record inserted

point
line
area
* type: min_road
update procedure: 0

OK

Name: Barnwell Road
Class: A-road dual
Length: 1.448e+003
Centreline
Annotation

Figure 8.67 - Execution of the Update Scenario - Stage 1

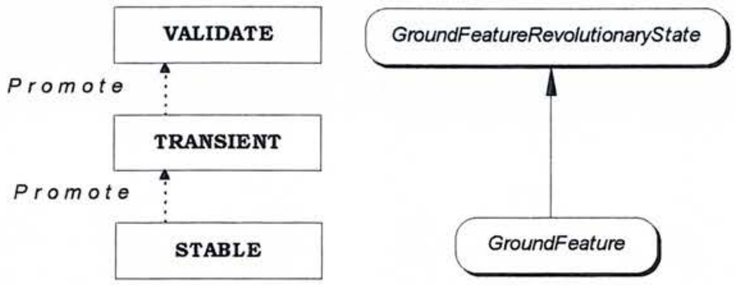
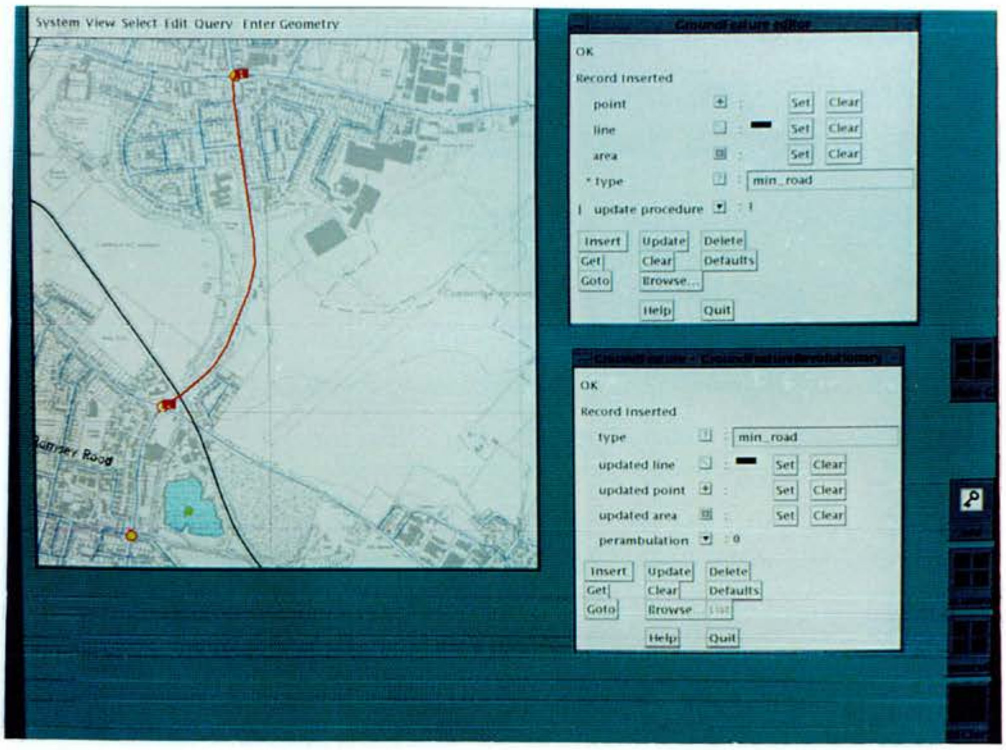


Figure 8.68- Creation of a new version from a previous existing one

The user can perform the update procedure by clicking on the `update procedure` element of the Ground Feature Edit menu (top right side of Figure 8.67). This will activate the Ground Feature - Ground Feature Revolutionary State Edit menu (bottom right side of Figure 8.69). This activated menu represents the transient version state (i.e. the ds_record of the GroundFeatureRevolutionaryState class) which has been promoted from its previous stable version state (i.e. the ds_record of the GroundFeature class). Since it involves the Promote operation in the spatio-temporal data model, the transient version state is created in a way that the invariant attribute (i.e. type) of the GroundFeature class is inherited by the GroundFeatureRevolutionaryState class (Figure 8.69). In this example, the inherited attribute is the min_road type. See Appendix D for the Magik code which is related to the Promote operation.

The version-significant attributes (point, line, area) belonging to the GroundFeature are not inherited. In order to differentiate, the GroundFeatureRevolutionaryState class presents updated point, updated line, and updated area as attributes. All the attributes of the GroundFeatureRevolutionaryState class are now considered as invariant attributes. They cannot be modified or deleted by any user at any time.



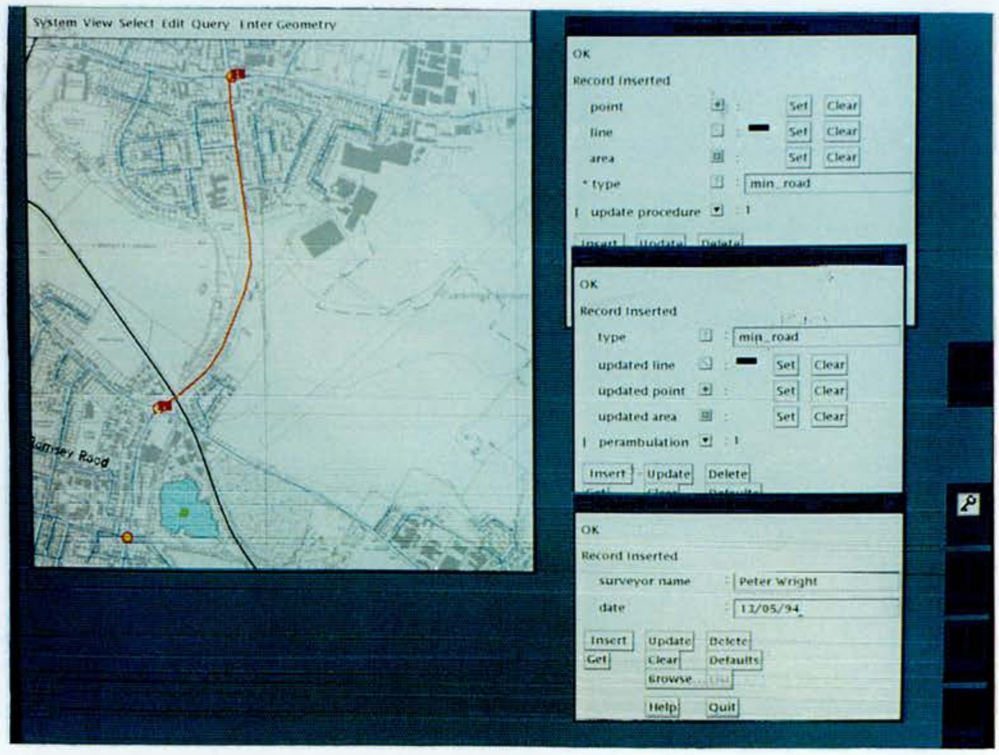
OK
Record Inserted
point :
line :
area :
* type : min_road
update procedure : 1

OK
Record Inserted
type : min_road
updated line :
updated point :
updated area :
perambulation : 0

Figure 8.69 - Execution of the Update Scenario - Stage 2

The user creates a new version of the ground feature object, in this case Barnwell Road, by digitising its new position and inserting it as the updated line attribute of the GroundFeatureRevolutionaryState class. Once the user has inserted the new updated line for the GroundFeatureRevolutionaryState class, the `update procedure` element of the Ground Feature Edit Menu (top right in Figure 8.69) is changed to 1, indicating that one update has occurred to this particular ground feature. The `perambulation` element of the Ground Feature - Ground Feature Revolutionary State Edit menu (bottom right in Figure 8.69) indicates the number 0 which signifies that the user has updated the ground feature on the system, but this has not yet been confirmed on the landscape by the surveyors.

The perambulation event takes place, once the change in position of the ground feature is confirmed by the surveyors as having taken place on the landscape. Consequently, the user can insert this information into the system by clicking on the `perambulation` element of the Ground Feature - Ground Feature Revolutionary State Edit menu (bottom right side of Figure 8.69). This will activate the Perambulation menu (middle right side of Figure 8.70) which contains the attributes concerning the perambulation event, for example, the name of the surveyor and the date when the perambulation occurred.



OK
Record Inserted
point :
line :
area :
* type : min_road
I update procedure : 1

OK
Record Inserted
type : min_road
updated line :
updated point :
updated area :
I perambulation : 1

OK
Record Inserted
surveyor name : Peter Wright
date : 13/05/94

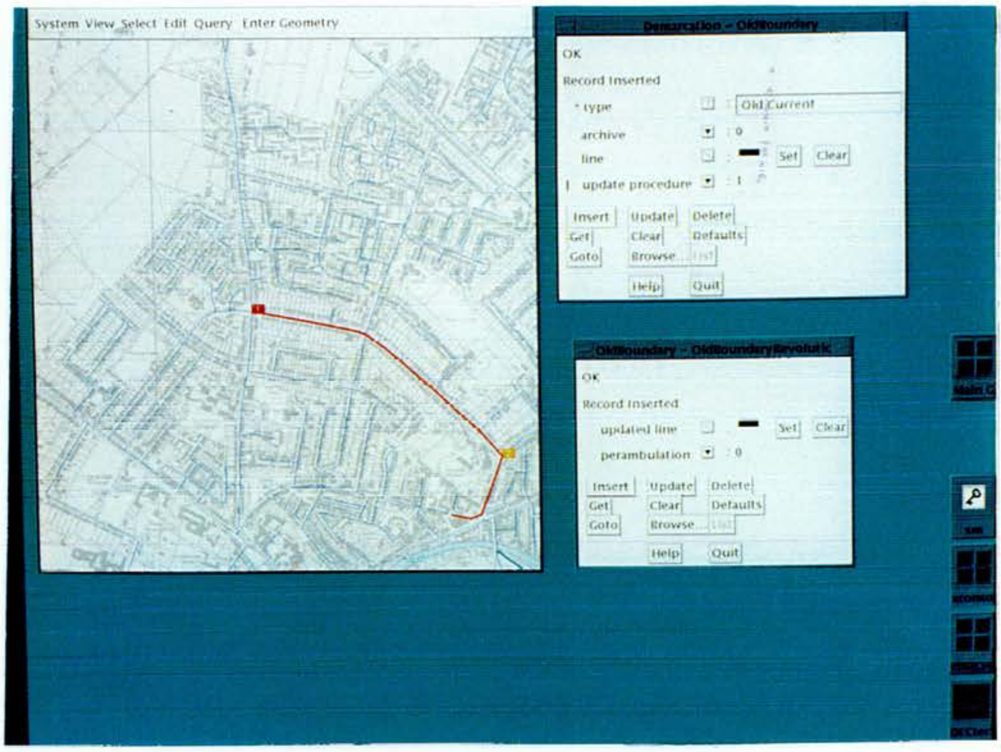
Figure 8.70 - Execution of the Update Scenario - Stage 3

At this stage, the `perambulation` element of the Ground Feature Revolutionary State Edit menu (middle right) indicates that one perambulation event has occurred for the selected ground feature revolutionary state.

8.4.2 The Mutation Circumstance of a Space-Time Path Relocation of an Existing Object - Stages 4 and 5

The last update procedure in the spatio-temporal data model involves the relocation of an existing object. The same boundary used for illustrating the Evolution Tracking Scenario has been selected for illustrating the execution of this update procedure in the Update Scenario. Figure 8.71 shows the relocation of this boundary. This is illustrated by the Demarcation - Old Boundary Edit Menu (top right side of Figure 8.71) in which the stable version state of this boundary is displayed. The update procedure has been carried out by activating the Old Boundary -Old Boundary Revolutionary State Edit menu (bottom right side of Figure 8.71) using the `update procedure` element of the Demarcation - Old Boundary Edit menu (top right side of Figure 8.71).

The new updated line has been inserted by the user into the ds_view of the VMDS, therefore the update procedure indicates the number 1. The Old Boundary - Old Boundary Revolutionary State Edit menu represents the invariant attributes of the validate version state of the ds_record of the GroundFeatureRevolutionaryState class (Figure 8.71).


System View Select Edit Query Enter Geometry
Demarcation - OldBoundary
Insert
Update
Delete
Get
Clear
Defaults
Goto
Browse...
List
Help
Quit
OldBoundary - OldBoundaryRevolutic

OK

Record Inserted

* type : Old Current

archive : 0

line : Set Clear

update procedure : 1

OK

Record Inserted

updated line Set Clear

perambulation : 0

Figure 8.71 - Execution of the Update Scenario - Stage 4

The Promote operation is also invoked for this update procedure in the spatio-temporal data model as illustrated in Figure 8.72.

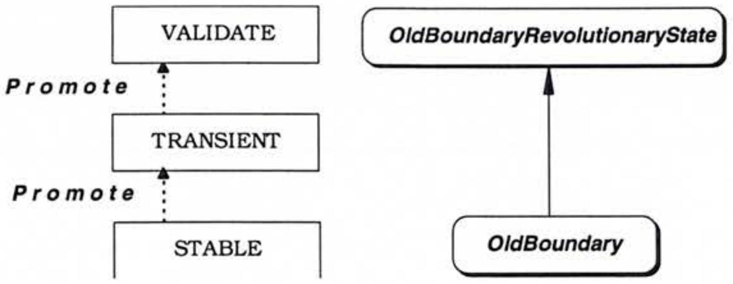
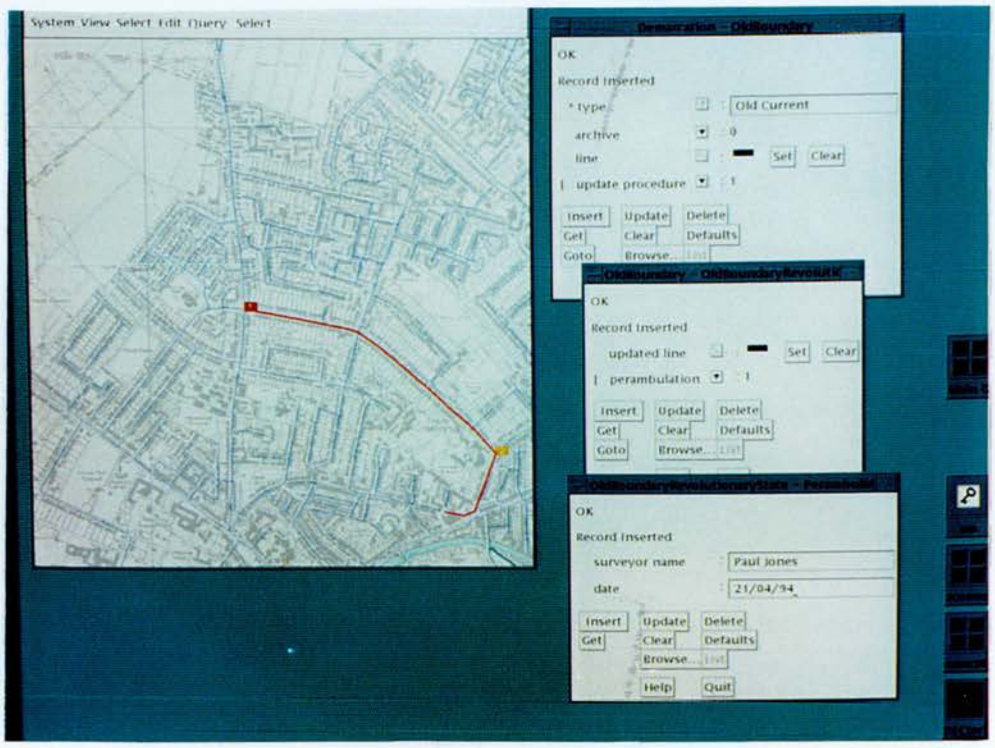


Figure 8.72- Version configuration for the relocation update procedure

Once the old boundary is relocated and its new updated line is inserted into the system, the validate version state of the old boundary revolutionary state is created. This validate version state presents only invariant attributes, and it can not be updated again. Once the confirmation of the public boundary's new position in the landscape is obtained, the user can insert the relevant information about the perambulation event by clicking on the `perambulation` element of the Old Boundary - Old Boundary Revolutionary State Edit menu (bottom right side of Figure 8.71). The Old Boundary Revolutionary State - Perambulation Edit menu (bottom right side of Figure 8.73) is activated, and the user is then able to capture the attribute values of the perambulation event within the system.


System View Select Edit Query Select
Demarcation - OldBoundary
Set
Clear
Insert
Update
Delete
Get
Clear
Defaults
Goto
Browse...
List
OldBoundary - OldBoundaryRevolutio
OldBoundaryRevolutionaryState - Perambulat
Help
Quit

OK
Record Inserted
* type : Old Current
archive : 0
line :
| update procedure : 1

OK
Record Inserted
updated line :
| perambulation : 1

OK
Record Inserted
surveyor name : Paul Jones
date : 21/04/94

Figure 8.73 - Execution of the Update Scenario - Stage 5

8.5 The Archiving Scenario

The Archiving Scenario represents the archiving of old boundaries which are no longer effective. It involves the creation of an obsolete state during the lifespan of a public boundary. This is achieved by clicking on the archive element of the Old Boundary Edit menu as previously illustrated in Figure 8.66.

The Smallworld GIS provides the benefit of allowing storage of obsolete boundary states created by the user on a mass storage device such as a CD ROM, and maintaining the other states on a conventional hard disk.

8.6 Historical Views

Historical views have been designed in the spatio temporal data model in order to visualise the incremental mechanism of space-time paths. Such views provide direct access to historical data concerned with a particular public boundary. This has been achieved by creating HistoricalView classes in the Smallworld GIS. The independent incremental mechanism of the spatio-temporal data model is illustrated in Figure 8.74. Two historical views are displayed as the Delimitation Historical View Editor menu and the Demarcation Historical View Editor menu. Both menus represent the union of all properties which belong to the Parent, Modifier, and Resultant classes in the Evolution Tracking Scenario. The historical view provides a snapshot of all properties of a specific public boundary which are selected by the user through the classes of objects. In Figure 8.74, the Delimitation Historical View provides the union of all properties which belong to DraftBoundary, Delimitation, and NewBoundary classes.

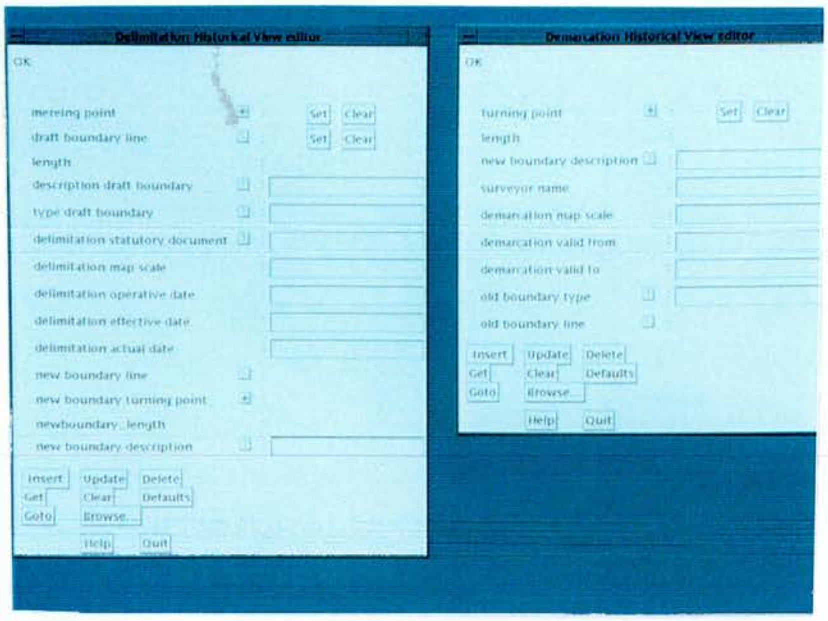


Figure 8.74 - Historical views for the Evolution Tracking Scenario

The overlapping incremental mechanism of space-time paths has been illustrated by creating a historical view termed Update Historical View (Figure 8.75). In this case, the Update Historical View Menu illustrates the properties of the GroundFeature, GroundFeatureRevolutionaryState, Perambulation, OldBoundary, and OldBoundaryRevolutionaryState classes.

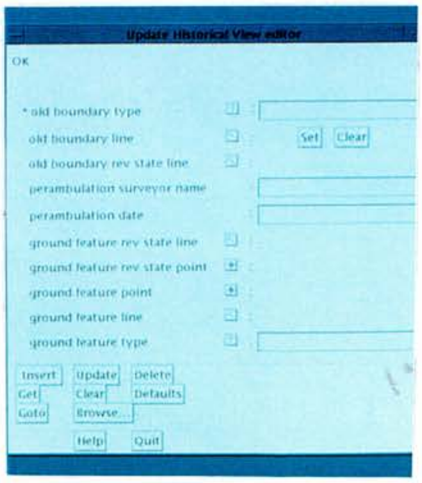


Figure 8.75 - The Update Historical View

In fact, the user can create as many historical views as is necessary in his/her application. In addition, he/she can specify the classes which should belong to each historical view. However, further research work is needed to address the historical views effectively. Computer-aided visualisation of historical views may provide a tool to assist the user in displaying the historical views in graphical, symbolic, and ideally dynamic forms, such as animated graphics, graphs or diagrams, or even a combination of these.

Visualisation may provide the optical information [McCormick et al. 1987], so users have a natural acuity for recognising and interpreting visual patterns [Fedra 1992, Buttenfield 1993], and an intuitive understanding of large amounts of data, processes, and interdependencies between historical views. The role of visualisation, then, is to remove the barrier between the user and the spatio-temporal data model. Visual Data Analysis (VDA) software couples powerful visualisation capabilities with statistical analysis functions, and represents a very pertinent implementation of visualisation which can be coupled with historical views of the spatio-temporal data model.

8.7 Conclusions

This chapter has illustrated some examples of how space-time paths can be created by a user. The states and events designed in the spatio-temporal data model are visualised as Edit Menus within the system. This allows a better understanding of how space-time paths can be implemented in the Smallworld GIS.

The prototype implementation has been undertaken mainly as a 'proof-of-concept' of the ideas developed in the spatio-temporal data model. Several drawbacks have been identified in implementing this prototype into the Smallworld. These include:

- the data store view (ds_view) does not allow references attributes or inheritance relationships between objects (See Table 8.8, p.165). This has meant that foreign keys have had to be used to represent the join relationships in the ds_view, i.e. the relationships between the ds_records of the ds_collections.

- because the Magik image and the ds_view support different levels of referential integrity (See Table 8.8, p.165), the prototype implementation has shown a discrepancy between these levels when they are executed for checking the correctness of the relationships between objects on space-time paths. The garbage collection algorithms in the Magik image are more resilient as a referential integrity approach. They detect a greater variety of errors than the checking support in a ds_view.

- although historical views have been implemented in the system, the need for a dynamic visualisation tool has been proved to be essential for displaying the properties which belong to a historical view. Further development work would be required in order to handle historical views fully within the system.

The prototype implementation has also contributed for pointed out possible improvements in the spatio-temporal data model. These are:

- the spatio-temporal data model captures the essential semantics (space-time paths, events, states) of an application, and through this model the user sees his/her problem domain composed of objects and the classes to which they belong. Classes and their relationships may be further refined, and some classes may be added or deleted. The union of time geography and object-orientation within a vehicle of a GIS system, has proven to be a dynamic modelling activity, in which the schema is constantly being upgraded.

- the constraints in the spatio-temporal data model are simple. Basically, they are methods attached to ds_records or ds_fields. This could be improved by associating knowledge based rules with events and states of the spatio-temporal data model. For example, 'when' event 'if' condition 'then' action.

Chapter 9
Conclusions

9.1 Summary

The focus of this work has been the analysis and design of a spatio-temporal data model based on time-geography, and specifically, the incorporation of such a model into a GIS system.

The main concepts involved in the time-geographic approach are the Space-Time Path and Potential Path Spaces, which have been used for developing the spatio-temporal data model. Both concepts have been considered to be fundamental in their significance for the temporal structuring of space within the spatio-temporal data model. Such temporal structuring of space has been achieved by integrating events, changes, and constraints for a space-time path. In identifying Potential Path Spaces, scenarios have been created in the spatio-temporal data model in which a set of space-time paths are attached to a specific circumstance describing the real-world in a particular space-time place. The conceptual elements of the time-geographic approach have been described in Chapter 2.

These conceptual elements can be utilised in modelling different applications in a GIS. This dissertation has demonstrated the use of such a spatio-temporal data model in an application for the evolution of public boundaries in Great Britain (Chapter 3). However, other different applications in GIS can benefit from utilising this same approach. One example is the prediction of environmental change due to long-term large scale climatic variations [Wachowicz and Broadgate 1993]. In this particular problem domain, prediction of environmental change requires an understanding of the principal mechanisms implicated in long-term large scale climatic variation. Uncovering these mechanisms can only be achieved by the analysis of past environment states as well as the recognition of patterns of change trough time.

A spatio-temporal data model based on time geographic approach provides the semantics of events and states which can be used to describe the environmental changes and can be categorised as effect (an environmental change which can be detected by experience or observation of the environment) and cause (circumstances acting over a period of time which produce an environmental change). The spatio-temporal data model proposed in this dissertation, enables the time-geographic framing of past environmental states (changes) by exploring the derivation of events from object states and vice-versa due to their spatio-temporal interdependence.

In framing public boundary evolution in time and space, the processes which most of the public boundaries would pass through in their lifespan have been modelled. In addition, the different evolutionary states associated with these processes, and the principal revolutionary states revealing the mutations occurring over public boundaries, have been modelled as well. This has been of particular importance in integrating space and time within the spatio-temporal data model.

The main challenge in this research has been concerned with the development and implementation of a spatio-temporal data model based on time-geography, within a GIS. This has been achieved by using the concepts in the object-oriented method proposed by Booch. The effectiveness of this object-oriented analysis and design method in modelling the elements of the Space-Time Path and the scenarios (Potential Path Spaces) has been confirmed in Chapter 5.

One of the main discoveries encountered in the object-oriented analysis has been the need to capture the meaning of change within the spatio-temporal data model. In other words, temporal data management within the spatio-temporal data model has been recognised as vital in dealing with the incremental modification mechanism of space-time paths. Temporal data management is a developing research area and it encompasses such domains as databases, artificial intelligence, and GIS. The diversity of the efforts involved in exploring temporal data management has been illustrated in Chapter 4.

A taxonomy of change has been devised for a GIS on the basis of previously developed concepts in the field of temporal data management. This aimed to provide a better understanding of how valid changes can reveal the density of space-time paths within the spatio-temporal data model (Chapter 6).

Managing valid changes in the space-time path has attested to the need for a version management approach within the spatio-temporal data model. Hence, the version management approach proposed by Ahmed and Navathe [1991] has been incorporated into the spatio-temporal data model (Chapter 6). The object-oriented features utilised in this versioning method allow for an integration between a versioning mechanism and the spatio-temporal data model. One result of this integration has been that the managing of versions can be implemented in a GIS system. In fact, this has provided a mechanism that prevents unnecessary proliferation of versions within the system.

The implementation of the object-oriented elements required by the spatio-temporal data model has mainly been as a 'proof-of-concept' approach. Although some drawbacks have been found in implementing the spatio-temporal data model in the Smallworld GIS, the results have been satisfactory for exploring the issues involved in developing such a model (Chapters 7 and 8).

Implementing the concepts developed in the spatio-temporal data model has introduced the philosophy of object-orientation in dealing with space and time within a GIS. This has been mainly in the notion of the existence of a state of an object at any point in time on a space-time path. Such a notion has been represented by the evolutionary and revolutionary states of a space-time path. This implies a new way of thinking about how space is organised in GIS systems.

9.2 Main Conclusions

The thesis of this dissertation is that the time-geographic approach is effective for integrating time and space in GIS and that this will reduce the difficulties currently encountered in constructing spatio-temporal data models.

A distinct 'object representation' has been developed in the spatio-temporal data model which has been presented in this dissertation. The principle elements of this representation have been identified as the space-time path, the evolutionary state, the revolutionary states, and the constraint (Potential Path Space). The result is a coherent framework providing extensive support for the construction of a spatio-temporal data model.

The review of background and related research clearly showed that most previous investigations have concentrated on one very well defined aspect of spatio-temporal data modelling. This research represents an attempt to draw together the results of research in time geography (Chapter 2), object-orientation (Chapter 5) and temporal data management (Chapter 4) into a spatio-temporal data model and thus provide a sound basis for future investigation in this field.

A detailed examination of the problem and requirements for implementing the spatio-temporal data model has been conducted and is presented in Chapter 4. The results of this investigation show that the spatio-temporal data model is operationally feasible, in that it can be implemented in a GIS.

Evaluating a spatio-temporal data model is a difficult and drawn out process. The approach taken in this dissertation has been to construct a prototype implementation as completely as possible within the time and resources available and to use it to evaluate the feasibility, and identify the strengths and weakness, of the spatio-temporal data model developed. A complete evaluation and in particular an implementation of a whole application system would take a much greater amount of time than that available for this dissertation.

The results of the implementation are presented in Chapter 8 and show that the prototype implementation preserves the principal features of the spatio-temporal data model. Care had to be taken when implementing these features, such as space-time paths and their associated incremental modification mechanism, to preserve the functionality of the spatio-temporal data model. The resulting implementation is therefore well placed to take full advantage of the object-oriented concepts (object identifier, inheritance, polymorphism) which will be available to GIS in the near future.

The strengths of the spatio-temporal data model and its implementation lie in the well defined 'object representation' that has been developed. This representation offers much practical guidance, and implies the feasibility of applying object-oriented methods to the problem of handling space and time in GIS. The time-geography approach brings to light the link between components as events, states, and constraints, and also much of the functionality of modelling space and time in the context of the problems domains in GIS.

The weaknesses lie in the lack of the inheritance mechanism and an appropriate access method for implementing space-time paths. New indexing techniques need to be developed in order to avoid problems in the implementation of these paths. Although this problem is not addressed in this dissertation, Section 7.5.1 indicates the requirements for further research on indexing techniques.

While this research is in the field of spatio-temporal data modelling in GIS, it is hoped that its findings will also contribute in the development of the next generation of GIS systems, which will have improved facilities for handling spatio-temporal data.

9.3 Recommendations for Further Work

The spatio-temporal data model presented in this dissertation emphasises many desirable characteristics , such as:

- a continuous space-time path that supports modelling elements in order to integrate space and time in GIS;
- a sensible compromise between the flexibility offered by object-oriented methods and the drawbacks of implementing an object-oriented data model in a GIS;
- scope for defining bitemporal and valid changes within a spatio-temporal data model.

However, further research is necessary, and mainly related to:

- Developing new indexing techniques for spatio-temporal data modelling.
- Investigating new approaches for storing and manipulating topological relationships in a GIS in order to avoid adding unnecessary complexity to the spatio-temporal data model.
- Design of innovative data structures in GIS so that spatio-temporal data modelling is not limited by the technology available.
- Development of temporal capabilities such as dynamic visualisation and spatio-temporal queries.

References

Ackoff, R.L. (1981) *Creating the Corporate Future*. New York: John Wiley and Sons.

Ahmed, R. and Navathe, S.B. (1991) Version Management of Composite Objects in CAD Databases. *Proceedings of the ACM SIGMOD International Conference on Management of Data,* 218-27.

Allen, J.F. (1983) Maintaining Knowledge About Temporal Intervals. *Communications of the ACM*, **26**, 832-43.

Al-Taha, K.K. and Barrera, R. (1990) Temporal Data and GIS : an Overview. *Proceedings of the GIS/LIS'90 Conference*, **1**, 244-254.

Al-Taha, K.K. , Snodgrass, R.T. and Soo, M.D. (1994) Bibliography on Spatio-Temporal Databases. *International Journal of Geographical Information Systems*, **8**(1), 95-103.

Anderson, T.L. (1982) Modeling Time at the Conceptual Level. In*: Improving Database Usability and Responsiveness*, Scherermann, P. (ed.), Jerusalem, Israel: Academic Press. 273-97.

Armenhakis, C. (1993) Map Animation and Hypermedia: Tools for Understanding Changes in Spatio-Temporal Data. *Proceedings of the Canadian Conference on GIS,* 859-69.

Armstrong, M.P. (1988) Temporality in Spatial Databases. *Proceedings of the GIS/LIS'88 Conference*, **2**, 880-90.

Banerjee, J., Kim, W., Kim H.J. and Korth, H.F. (1987) Semantics and Implementation of Schema Evolution in Object-Oriented Database Systems. *Proceedings of the ACM SIGMOD Conference*, **5**(1), 3-26.

Barrera, R., Frank, A.U. and Al-Taha, K. (1991) Temporal Relations in Geographic Information Systems : a Workshop at the University of Maine. *SIGMOD Record*, **20**(3), 85-91.

Beller, A. (1991) Spatial and Temporal Events in a GIS. *Proceddings of GIS/LIS'91*, **2**, 766-775.

Beller, A, Giblin, T., Le, K.V., Litz, S., Kittel, T. and Schimel, D. (1991) A Temporal GIS Prototype for Global Change Research. *Proceedings of GIS/LIS'91*, **2**, 752-765.

Ben-Zvi, J. (1982) *The Time Relational Model.* PhD dissertation (unpublished), University of California, Los Angeles - UCLA.

Blom, T. and Löytönen, M. (1993) Research Launch System to Monitor Epidemics in Finland. *GIS Europe,* **2**(5), 27-9.

Boehm, B. (1986) A Spiral Model of Software Development and Enhancement. *Software Engineering Notes*, **11**(4), 22-34.

Bonczek, R.H., Holsapple, C.W. and Whinston, A.B. (1981*)* *Foundations in Decision Support Systems*, New York: Academic Press.

Booch, G. (1986) Object-Oriented Development. *IEEE Transactions on Software Engineering*, **12**(2), 211-21.

Booch, G. (1991) *Object-Oriented Design with Applications.* Redwood City, CA: Benjamin/Cummings.

Booch, G. (1994) *Object-Oriented Analysis and Design with Applications.* Santa Clara, CA: Benjamin/Cummings, 2nd Edition.

Booth, J. R. S. (1980) *Public Boundaries and Ordnance Survey 1840-1980.* R.A.G. Powell, Ordnance Survey, Southampton.

Burrough, P.A. (1986) *Principles of Geographical Information Systems for Land Resources Assessment.* Oxford: Clarendon Press.

Buttenfield (1993) Scientific Visualisation for Environmental Modeling: Interactive and Proactive Graphics. *Proceedings of the Second Conference on Integrating GIS and Environmental Modelling*, no pages.

Catell, R.G.G. (1991) *Object Data Management: Object-Oriented and Extended Relational Database Systems.* Reading, MA: Addison-Wesley.

Chance, A., Newell, R.G. and Theriault, D.G. (1990) An Object-Oriented GIS - Issues and Solutions. *Proceedings of the EGIS'90 Conference*, **1**, 179-88.

Chen, P. (1976) The Entity-Relationship Model: Toward a Unified View of Data. *ACM Transactions on Database Systems*, **1**(1), 9-36.

Chen, S.. (1990) *Advances in Spatial Reasoning.* New Jersey: Ablex Publishing Corporation..

Chrisman, N.R. (1993) Beyond Spatio-Temporal Data Models: A Model of GIS as a Technology Embedded in Historical Context. *Proceedings Auto Carto 11*, **1**, 23-32.

Clifford, J. (1982) *A Logical Framework for the Temporal Semantics and Natural-Language Querying of Historical Databases*. PhD dissertation (unpublished), Department of Computer Science, SUNY at Stony Brook.

Clifford, J. (1983) *Towards an Algebra of Historical Relational Databases*. Technical Report, Department of Computer Applications and Information Systems, New York University, December 1983.

Clifford, J. and Ariav, G. (1986) Temporal Data Management: Models and Systems. In: *New Directions for Databases Systems*, Ariav, G. and Clifford, J. (eds.), New Jersey: Ablex. 168- 85.

Clifford, J. and Warren, D.S. (1983) Formal Semantics for Time in Databases. *ACM Transactions on Database Systems*, **8**(2), 214-54.

Coad, P. and Yourdon, E. (1990) *Object-Oriented Analysis*. Englewood Cliffs, NJ: Yourdon Press/Prentice-Hall.

Coad, P. and Yourdon, E. (1991a) *Object-Oriented Analysis*. Englewood Cliffs, NJ: Yourdon Press/Prentice Hall.

Coad, P. and Yourdon, E. (1991b) *Object-Oriented Design*. Englewood Cliffs, NJ: Yourdon Press/Prentice Hall.

Coombes, M., Openshaw, S., Wong, C. and Raybould, S. (1993) GIS in Community Boundary Definition. *Mapping Awareness and GIS,* **7**(4), 41-44.

Copeland, G. and Maier, D. (1984) Making Smalltalk a Database System. *Proceedings of the ACM SIGMOD Conference*, 316-25.

Dadam, P., Lum, V. and Werner, H.D. (1984) Integration of Time Versions into a Relational Database System. *Proceedings of the Conference on Very Large Databases*, 509-522.

Dahl, O.J. and Nygaard, K. (1966) SIMULA - An Algol-Based Simulation Language. *Communications of the ACM,* **9**, 671-78.

Dahl, O.J., Myrhaug, B. and Nygaard, K. (1968) *Simula 67 Common Base Language*. Norwegian Computing Centre, Oslo.

Dean, T.L. and McDermott, D.V. (1987) Temporal Data Base Management. *Artificial Intelligence*, **32**, 1-55.

Easterfield, M. (1993) Personal Communication.

Egenhofer, M.J. and Frank, A.U. (1989) Object-Oriented Modeling in GIS: Inheritance and Propagation. *Proceedings of the AutoCarto 9 Conference*, 588-98.

Egenhofer, M.J. and Al-Taha, K.K. (1992) Reasoning About Gradual Changes of Topological Relationships. In: *Theories and Methods of Spatio-Temporal Reasoning in Geographic Space*, Frank, A.U., Campari, I. and Formentini, U. (eds.), London: Springer-Verlag. 196-219.

Fedra, K. (1992) *Interactive Environmental Software: Integration, Simulation, and Visualisation.* IIASA RR-92-10. International Institute for Applied Systems Analysis, Austria.

Fonseca, A., Gouveia, C., Camara, A.S. and Ferreira, F.C. (1992) Functions for Multimedia GIS. *Proceedings of the EGIS'92 Conference*, **2**, 1095-1101.

Frank, A.U. (1994) Qualitative Temporal Reasoning in GIS - Ordered Time Scales. *Proceedings of the SDH'94 Conference*, **1**, 410-430.

Gardels, K. (1992) SEQUOIA 2000: New Geographic Information Management Technologies for Global Change Research. *Proceedings of the EGIS'92 Conference*, **2**, 922-29.

Gersmehl, P.J. (1990) Choosing Tools: Nine Metaphors of Four-Dimensional Cartography. *Cartographic Perspectives*, **5**, 3-17.

Graham, I. (1994) *Object-Oriented Methods.* London: Addison-Wesley.

Hägerstrand, T. (1975) Space, Time and Human Conditions. In: *Dynamic Allocation of Urban Space.* Karlqvist, A. , Lunqvist, L. and Snickars, F. (eds), Farnborough, UK. Saxon House. 3-14.

Hägerstrand, T. (1970) What about People in Regional Science. *Papers of the Regional Science Association,* **24**, 7-21.

Hall, E. (1966) *The Hidden Dimension.* London: The Bodley Head.

Harley, J.B. (1989) Historical Geography and the Cartographic Illusion. *Journal of Historical Geography,* **15**, 80-91.

Haushofer, K. (1927) *Grenzen im ihrer Geographischen und Politischen Bedeutung.* (The Geographical and Political Significance of Boundaries). Berlin: Vowinckel.

Hazelton, N.W.J., Leahy, F.J. and Williamson, I.P. (1990) On the design of temporally referenced 3-D Geographic Information Systems: Development of Four-Dimensional GIS. *Proceedings of GIS/LIS'90 Conference*, 357-372.

Hoop, van S. and Oosterom, van P. (1992) Storage and Manipulation of Topology in POSTGRES. *Proceedings of the EGIS'92 Conference*, **2**, 1324-36.

Jacobson, I., Christerson, M., Jonsson, P. and Overgaard, G. (1992) *Object-Oriented Software Engineering*. Workingham, England: Addison Wesley.

Johnson, D. and Kemp, Z. (1995) Enhancing a GIS with temporal capabilities. *Proceedings of GISRUK'95*, Extended Abstract, 25-6.

Jones, S.B. (1945) *Boundary-making, a Handbook for Statesmen, Treaty Editors and Boundary Commissioners.* Washington, DC: Carnegie Endowment for International Peace.

Jones, S. and Mason, P.J. (1980) Handling the Time Dimension in a Data Base. *Proceedings of the International Conference on Data Bases*, 65-83.

Jones, S., Mason, P.J. and Stamper, R. (1979) LEGOL 2.0: A Relational Specification Language for Complex Rules. *Information Systems*, **4**(4), 293-305.

Käfer, W., Ritter, N. and Schöning, H. (1990) Support for Temporal Data by Complex Objects. *Proceedings of the Very Large Data Bases Conference*, 24-35.

Kemp, Z. and Kowalczyk A. (1994) Incorporating the Temporal Dimension in a Geographical Information System. In*: Innovations in GIS*, M.F. Woboys (ed.), London: Taylor and Francis.

Kim, W. (1991) Object-Oriented Database Systems: Strengths and Weakness. *Journal of Object-Oriented Programming*, July-August, 21-3.

Kim, W. and Lochosvky, F.H. (1989) *Object-Oriented Concepts, Applications and Databases.* Reading, MA: Addison-Wesley.

Kolovson, C.P. (1990) Indexing Techniques for Multi-Dimensional Spatial Data and Historical Data in Database Management Systems. PhD dissertation (unpublished), University of California at Berkeley.

Kraak, M. and MacEachren A.M. (1994) Visualization of the Temporal Component of Spatial Data. *Proceedings of the SDH'94 Conference*, **1**, 391-409.

Krasner, G. (1981) The Smalltalk-80 Virtual Machine. *Byte*, **6**(8), 12-20.

Langran, G. (1988) Temporal GIS Design Tradeoffs. *Proceedings of the GIS/LIS'88 Conference*, 890-899.

Langran, G. (1989) A Review of Temporal Database Research and its Use in GIS Applications. *International Journal of Geographical Information Systems*, **3**(3), 215-232.

Langran, G. (1992a) *Time in Geographic Information Systems.* London: Taylor & Francis.

Langran, G. (1992b) States, Events, and Evidence: The Principle Entities of a Temporal GIS. *Proceedings of the GIS/LIS'92 Conference*, **1**, 416-25.

Langran, G. (1993) Issues of Implementing a Spatiotemporal System. *International Journal of Geographical Information Systems* **7**(4), 305-14.

Lapradelle, P. de (1928) *La Frontière: Etude de Droit International.* (The Boundary: a Study of International Law). Paris: Les Editions Internationales.

Laurini, R. and Thompson, D. (1992) *Fundamentals of Spatial Information Systems.* San Diego, CA: Academic Press.

Lenntorp, B. (1976) *Paths in Space-Time Environments. A Time-Geographic Study of Movement Possibilities of Individuals.* Sweden: The Royal University of Lund.

Lenntorp, B. (1978) A Time-Geographic Simulation Model of Individual Activity Programmes. In: *Timing Space and Spacing Time Volume 2: Human Activity and Time Geography,* Carlstein, T., Parkes, D. and Thrift, N. (eds), Sweden: The Royal University of Lund. 162-80.

Loomis, M.E.S. (1992) Object Versioning. *Journal of Object-Oriented Programming*, January, 40-43.

Lum, V.P., Dadam, P., Erbe, R., Guenauer, J., Pistor, P., Walch, G., Werner, H. and Woodfill, J. (1984) Designing DBMS Support for the Temporal Dimension. *Proceedings of the ACM SIGMOD Conference on Management of Data*, 115-130.

Makin, J. (1992) *An Object-Oriented Simulation of a Complex Geographical System Using GIS.* MSc dissertation (unpublished), University of Edinburgh.

Mårtensson, S. (1978) Time Allocation and Daily Living Conditions: Comparing Regions. In: *Timing Space and Spacing Time Volume 2: Human Activity and Time Geography.* Carlstein, T., Parkes, D. and Thrift, N. (eds), Sweden: The Royal University of Lund. 181-97.

McCormick, B.H., Defanti, T.A. and Brown, M.D. (1987) Visualisation in Scientific Computing. *SIGGRAPH Computer Graphics Newsletter*, **21**(6).

McKenzie, L.E. and Snodgrass, R.T. (1991) Evaluation of Relational Algebras Incorporating the Time Dimension in Databases. *ACM Computing Surveys*, **23**(4), 501-543.

Miller, H.J. (1991) Modelling Accessibility Using Space-Time Prism Concepts Within Geographical Information Systems. *International Journal of Geographical Information Systems*, **5**(3), 287-301.

Milne, P., Milton S. and Smith, J. (1993) Geographical Object-Oriented Databases - a Case Study. *International Journal of Geographical Information Systems*, **7**(1), 39-55.

Moellering, H. (1980) The Real-Time Animation of Three-Dimensional Maps. *American Cartographer*, **7**(1), 67-75.

Mueller, T. and Steinbauer, D. (1983) Eine Sprachschnittstele zur Versionenkontrolle in CAM-Datanbaken. In: *Informatik-Fachberichte*, Berlin: Springer, 76-95.

Navathe, S.B. and Ahmed, R. (1986) *A Temporal Relational Model and a Query Language.* Technical Report, Computer And Information Sciences Department, University of Florida, April.

Newell, R.G. and Batty P.M. (1993) GIS Databases are Different. *Proceedings of the AGI'93 Conference*, 3.2.1-3.2.4

Newell, R.G., Theriault, D.G. and Easterfield, M. (1994) *Temporal GIS - Modelling the Evolution of Spatial Data in Time.* Smallworld Technical Report, Paper 6.

ODMG (1994) Response to the March 1994 ODMG-93 Commentary. *SIGMOD Record*, **23**(3), 3-7.

Olander, L. and Carlstein, T. (1978) The Study of Activities in the Quartenary Sector. In: *Timing Space and Spacing Time Volume 2: Human Activity and Time Geography.* Carlstein, T., Parkes, D. and Thrift, N. (eds), Sweden: The Royal University of Lund. 198-213.

Paddison, R. (1983) *The Fragmented State. The Political Geography of Power.* London: Basil Blackwell Publisher Limited.

Peuquet, D. (1994) It's about time: A Conceptual Framework for the Representation of Temporal Dynamics in Geographic Information Systems. *Annals of the Association of American Geographers,* **84**(3), 441-61.

Peuquet, D. and Wentz, E. (1994) An Approach for Time-Based Analysis of Spatio-Temporal Data. *Proceedings* of the *SDH'94 Conference*, **1**, 489-504.

Pred, A. (1977) The Choreography of Existence: Comments on Hägerstrand's Time Geography and Its Usefulness. *Economic Geography*, **53**, 207-21.

Prescott, J.R.V. (1987) *Political Frontiers and Boundaries.* London: Unwin Hyman Limited.

Rackham, L.J. (1987) *The Creation of a Prototype Relational Database for Public Boundaries and Administrative Areas in Scotland.* MSc dissertation (unpublished), University of Edinburgh.

Rackham, L.J. (1992) Development of a System for the Management and Supply of Data on Administrative Areas and Public Boundaries. *Third Meeting CERCO Working Group IX - Updating of Digital Maps and Topographic Databases,* 1-13.

Ramachandran, B. (1992) *Modelling Temporal Changes in the Structure of Real-World Entities Within a GIS Environment Using an Object-Oriented Approach.* MSc dissertation (unpublished), University of Edinburgh.

Ratzel, F. (1897) *Politische Geographie.* (Political Geography). Berlin: Oldenbourg.

Reed, D. (1978) *Naming and Synchronization in a Decentralized Computer System.* PhD dissertation (unpublished), M.I.T.

Rescher, N. and Urquehart, A. (1971) *Temporal Logic.* New York: Springer Verlag.

Rojas-Vega, E. and Kemp, Z. (1994) Object-Orientation and Spatial Data Modelling: a Formal Approach. *Poster Session, UKRGIS'94 Conference.*

Rubenstein, R. and Hersh, H. (1984) *The Human Factor: Designing Computer Systems for People.* Bedford, USA: Digital Press.

Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F. and Lorensen, W. (1991) *Object-Oriented Modelling and Design.* Englewood Cliffs, NJ: Prentice-Hall.

Sack, R.D. (1986) *Human Territoriality: its Theory and History.* Cambridge: Cambridge University Press.

Schneider, R. and Kriegel, H.P. (1992) Indexing the spatio-temporal monitoring of a polygon object. *Proceedings of the SDH'92 Conference*, **1**, 209-20.

Shohan, Y. and Goyal, N. (1988) Temporal Reasoning in Artificial Intelligence. In: *Exploring Artificial Intelligence: Survey Talks from the National Conferences on Artificial Intelligence,* Shrobe, H. E. and the American Association for Artificial Intelligence. (eds), San Mateo, CA: Morgan Kaufman Publishers.

Snodgrass, R.T. (1987) The Temporal Query Language TQuel. *ACM Transactions on Database Systems*, **12**(2), 247-98.

Snodgrass, R.T. (1990) Temporal Databases Status and Research Directions. *SIGMOD Record*, **19**(4), 83-9.

Snodgrass, R.T. (1992) Temporal Databases. In: *Theories and Methods of Spatio-Temporal Reasoning in Geographic Space*, Frank, U., Campari, I. and Formentini, U. (eds), London: Springer-Verlag, 22-64.

Snodgrass, R.T. and Ahn, I. (1985) A Taxonomy of Time in Databases. *Proceedings of the ACM-SIGMOD Conference on Management of Data*, 236-46.

Snodgrass, R.T. and Ahn, I. (1986) Temporal Databases. *Computer*, **19**(9), 35-42.

Snodgrass, R.T. , Al-Taha, K. and Soo, M. D. (1993) Bibliography on Spatiotemporal Databases. *SIGMOD Record,* **17**(1), 10-21.

Soo, M.D. (1991) Bibliography on Temporal Databases. *SIGMOD Record*, **20**(1), 14-23.

Stonebraker, M. (1987) The Design of the POSTGRES Storage System. *Proceedings of the Very Large Databases Conference*, 289-300.

Stroustrup, B. (1988) What is Object-Oriented Programming? *IEEE Software*, May, 10-20.

Tansel, A.U. , Clifford, J. , Gadia, S. , Jajodia, S. , Segev, A. and Snodgrass, R. (1993) *Temporal Databases - Theory, Design, and Implementation.* Redwood City, CA: Benjamin/Cumming.

Thewessen, T. , Van de Velde, R. and Verlouw H. (1992) European Grandwater Threats Analyzed with GIS. *GIS Europe*, **1**(3), 28-33.

Thrift, N. J. (1983) On the Determination of Social Action in Space and Time. *Environment and Planning D : Society and Space.* **1**(1), 23-58.

Varma, H. (1994) A new tool for managing very large volumes of Hydrographic Data. *Proceedings of the Hidrographic Conference,* 50-4.

Vrana, R. (1989) Historical Data as an Explicit Component of Land Information Systems. *International Journal of Geographical Information Systems*, **3**(1), 33-49.

Wachowicz, M. and Broadgate, M.L. (1993) A Significant Challenge: Prediction of Environmental Changes using a Temporal GIS. *Proceedings of the AGI'93 Conference*, 2.25.1-2.25.5

Wasserman, A.I., Pircher, P.A. and Muller, R.J. (1990) The Object-Oriented Structure Design for Software Design Representation. *IEEE Computer*, March, 50-62.

Whigham, P.A. (1993) Hierarchies of Space and Time. *Proceddings of COSIT'93 Conference*, **1**, 190-201.

Wiederhold, G., Fries, J.F. and Weyl, S. (1975) Structured Organization of Clinical Data Bases. *Proceedings of the AFIPS National Conference*, 479-485.

Wegner, P and Zdonik, S.B. (1988) Inheritance as an Incremental Modification Mechanism or What like is and isn't like. *Proceedings ECOOP'88*, 55-7.

Worboys, M.F. (1992) Object-Oriented Models of Spatiotemporal Information. *Proceedings of the GIS/LIS'92 Conference*, **2**, 825-34.

Worboys, M.F. (1994) Unifying the Spatial and Temporal Components of Geographical Information. *Proceedings of the SDH'94 Conference*, **1**, 505-517.

Worboys, M.F., Hearnshaw, H. and Maguire, D. (1990) Object-Oriented Data Modelling for Spatial Databases. *International Journal of Geographical Information Systems*, **4**(4), 369-83.

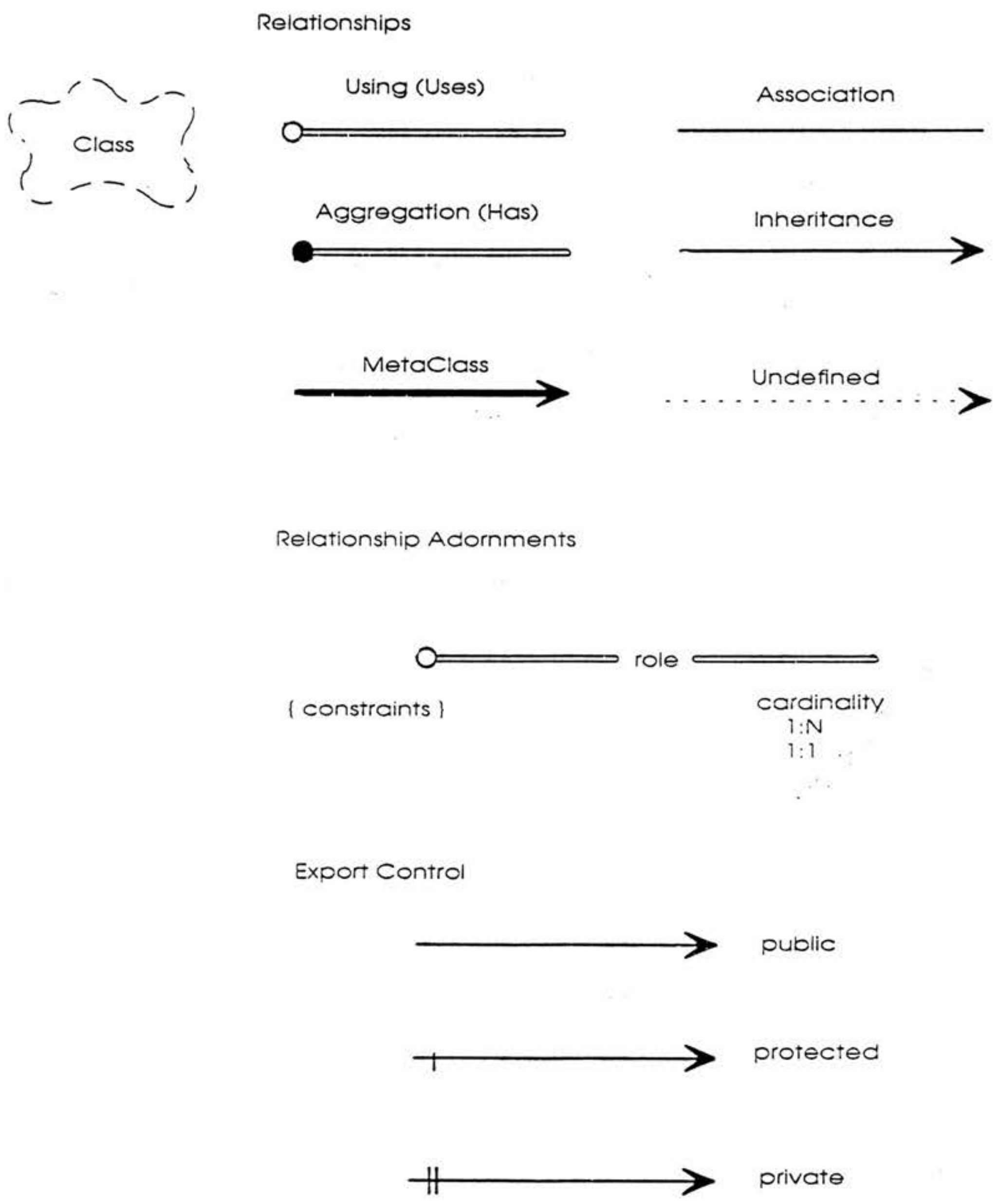
X3/SPARC/DBSSG/OODBTG (1991) *Final Technical Report, American National Standards Institute*. National Institute of Standards and Technology, Gaithersburg, MD 20899, USA.

Xiao, Q., Raafat, H. and Gauthier, D. (1989) A Temporal/Spatial Database Structure for Remotely Sensed Image Data Management within a GIS. *Proceedings of GIS/LIS'89*, **1**, 116-122.

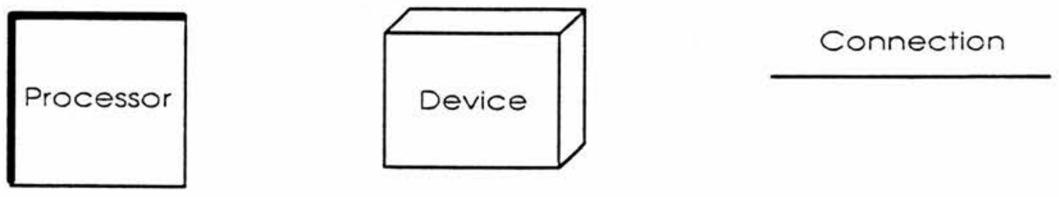
Appendix A

Notation Summary for the Booch's Object-Oriented Method

ObjectMaker Representation of
Booch's Class Diagram Notation

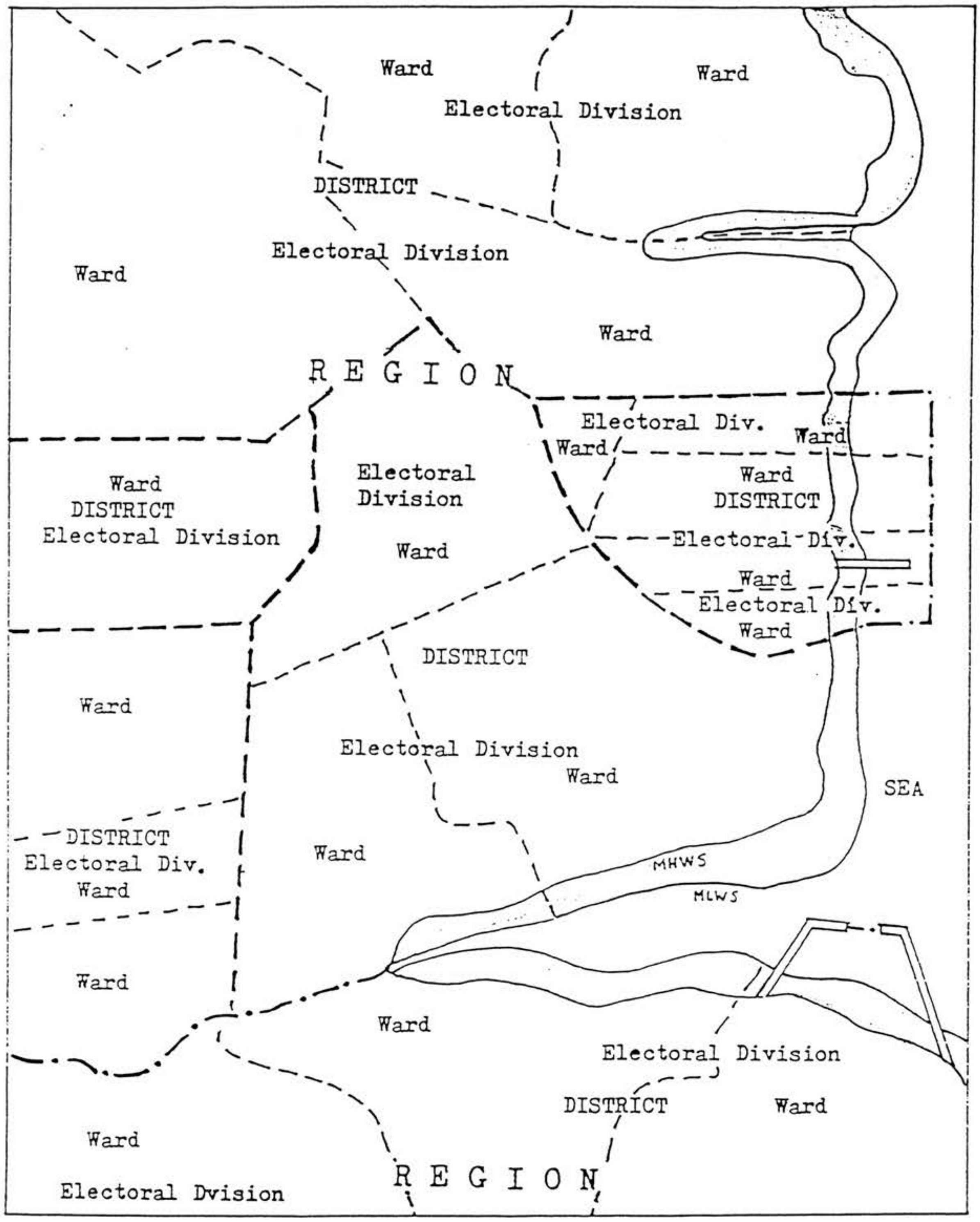


ObjectMaker Representation of
Booch's Process Diagram Notation

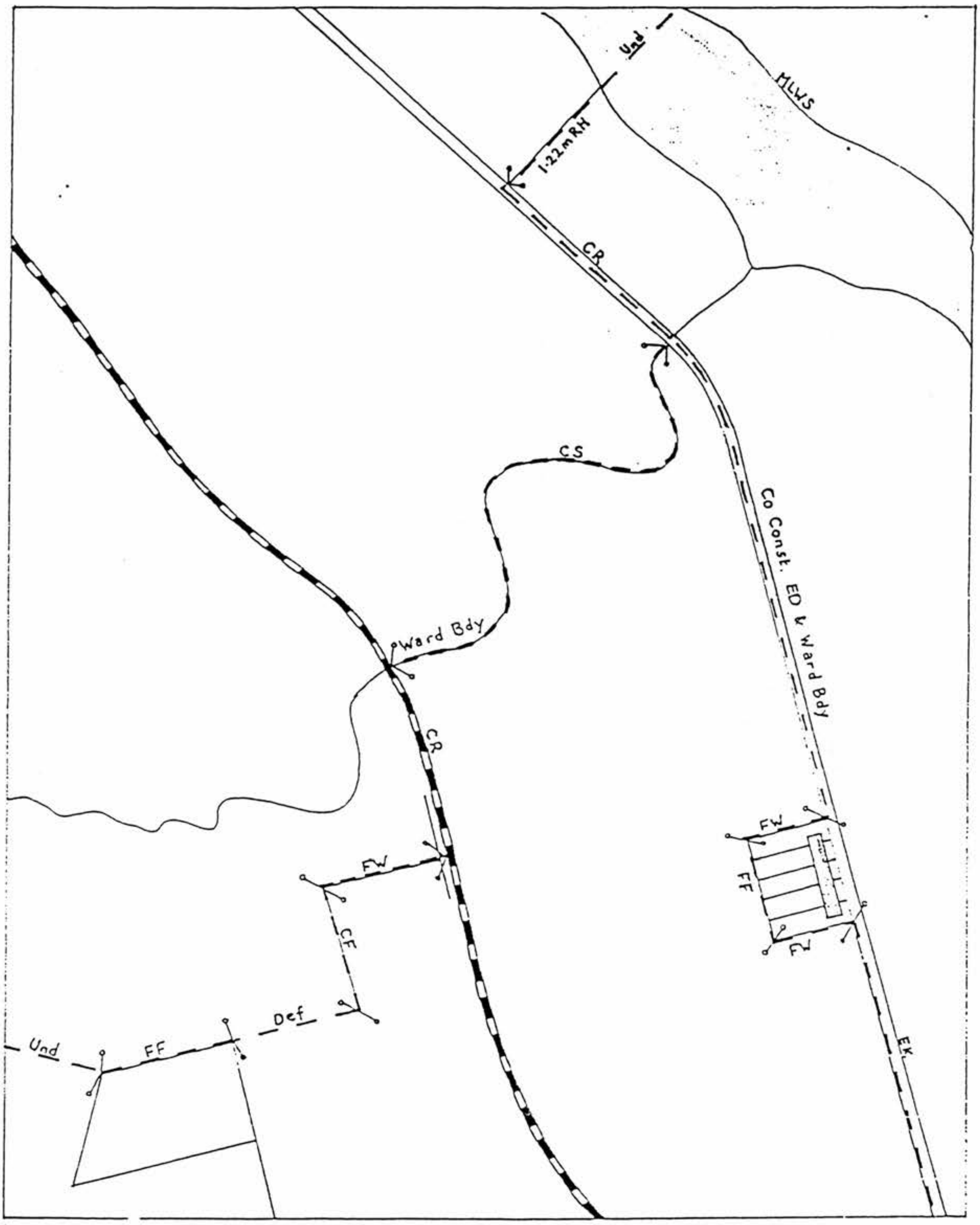


Appendix B

Schematic Diagrams Illustrating the Graphical Elements Utilised for Depicting Public Boundaries on Ordnance Survey Basic Scales



(Source: Rackham 1987, p.39)

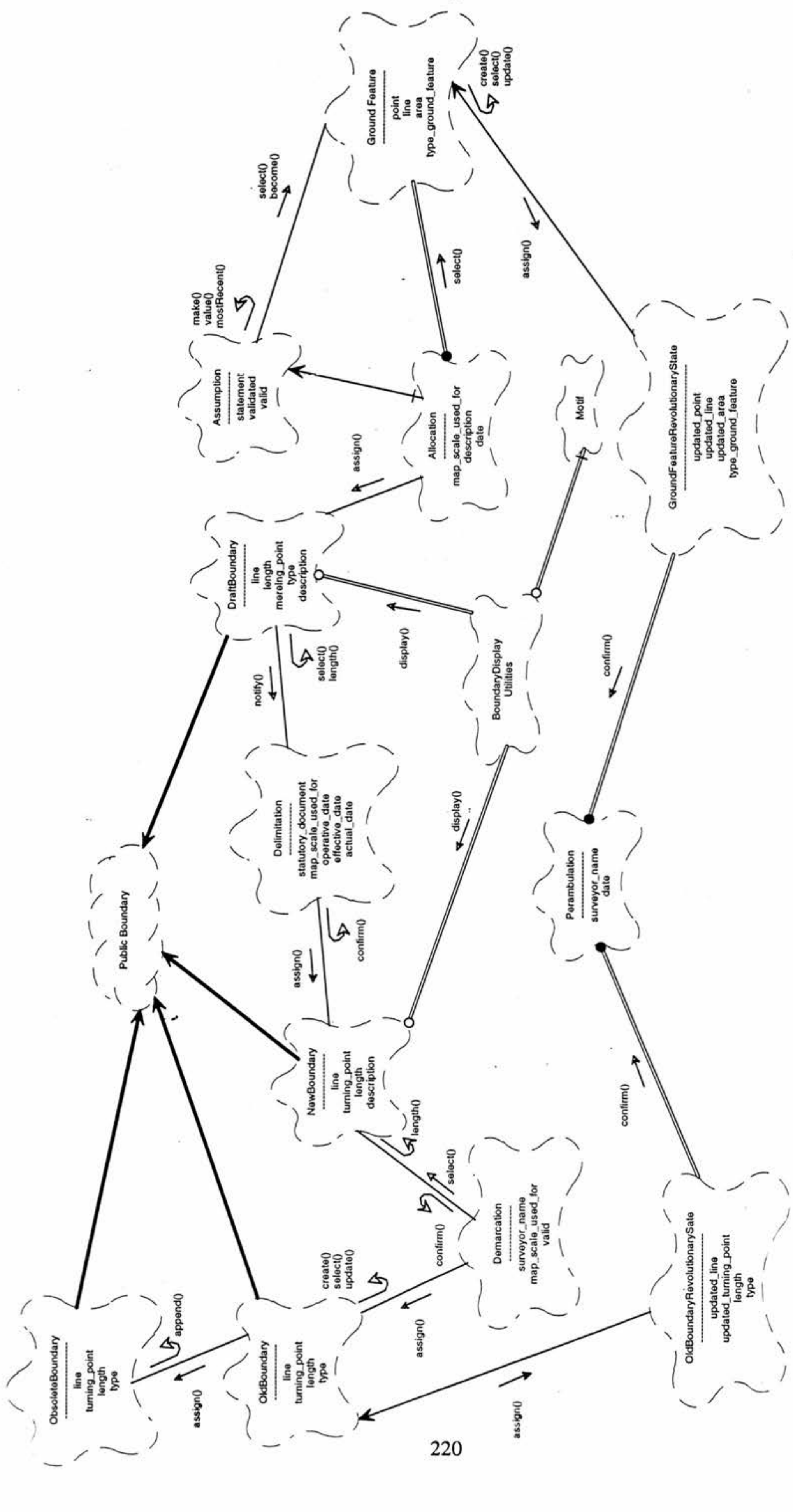


K E Y

— — — public boundaries	CR = centre of road/rly	FW = face of wall	
—┃— change of mereing	CF = centre of fence	RH = root of hedge	
MLWS Mean Low Water Spring Tides	CS = centre of stream	Def = defaced	
	EK = edge of kerb	Und = undefined	
	FF = face of fence		

(Source: Rackham 1987, p.34)

Appendix C
Overview of the Spatial Data Model



Appendix D
Smallworld Report

This report consists of two parts:
- Part I - Classes of Objects: It describes the properties and the structure of a ds_collection;
- Part II - Methods : It describes the methods defined within the Smallworld GIS.

PART I: CLASSES OF OBJECTS

I. Classes of Objects

Properties :
It describes the editor used for the graphic interface of a ds_collection. It also provides a list of visible ds_fields to the end-user.

Superstructure	It shows the cardinality of a relationship between two ds_collections in such a way that the second class is a child class.
Substructure	It shows the cardinality of a relationship between two ds_collections in such a way that the second class is a parent class.

Field Summary :
It describes the data types of each ds_field defined for a ds_collection.

I. Classes of Objects

Object name : newboundary (NewBoundary)

Properties :

Editor: component_editor
Visible Fields: (default)

: line
: turning_point
: length
: description_new_boundary
: demarcations

Superstructure	: 1:n to delimitation
Substructure	: 1:n with demarcation

Field Summary :

Physical Fields	Field Type	Man?	Default	Unset
* new_boundary_id	sys_id	True		
V description_new_boundary	description_new_boundary	True		
I delimitation_delimitationl	sys_id	True		

Logical Fields	Field Type
V length	ds_float

Geometric Fields	Field Type	Man?	Manifold
V line	chain	False	boundary_making

V turning_point	point	False	boundary_making

Join Fields	Join Object
delimitation	delimitation
V demarcations	demarcation

<u>Field Properties :</u>

Physical Field : new_boundary_id (new boundary id)

Properties :

Key Field	:
Field Type	: sys_id
Mandatory	: True
Editor	: field_editor
Generator	: make_sysid()
Print Width	: 10

Physical Field : description_new_boundary (description)

Properties :

Field Type	: description_new_boundary
Mandatory	: True
Print Width	: 10

Physical Field : delimitation_delimitation_id (delimitation_delimitation_id)

Properties :

Field Type	: sys_id
Mandatory	: True
Editor	: field_editor
Print Width	: 10

Foreign key from delimitation : field delimitation_id

Logical Field : length (length)

Properties :

Field Type	: ds_float
Print Width	: 10

Geometric Field : line (line)

Properties :

Geometry Type	: chain
Manifold	: boundary_making
Mandatory	: False

Geometric Field : turning_point (turning point)

Properties :

Geometry Type	: point
Manifold	: boundary_making
Mandatory	: False

Join Field : delimitation (delimitation)

Properties :

Join Object	: delimitation
Editor	: follow_field_editor
Cascade Delete?	: False
Aspect Field	: delimitation_id

Join Field : demarcations (demarcation)

Properties :

Join Object	: demarcation
Editor	: component_field_editor
Cascade Delete?	: True

Object name : groundfeaturerevolutionarystate (GroundFeatureRevolutionaryState)

Properties :

Editor: component_editor
Visible Fields : (default)

: groundfeature_type_ground_featur
: line
: point
: area
: perambulations

Superstructure	: 1:n to groundfeature
Substructure	: 0:n with perambulation

Field Summary :

Physical Fields	Field Type	Man?	Default	Unset
* ground_feature_revolutionl	sys_id	True		
I groundfeature_ground_featl	sys_id	True		
V I groundfeature_type_groundl	type_ground_feature	True	undefined	

Geometric Fields	Field Type	Man?	Manifold
V line	chain	False	boundary_making
V point	point	False	boundary_making
V area	area	False	boundary_making

Join Fields	Join Object
groundfeature	groundfeature
V perambulations	perambulation

Field Properties :

Physical Field : ground_feature_revolutionary_id (ground feature revolutionary id)

Properties :

Key Field	:
Field Type	: sys_id
Mandatory	: True
Editor	: field_editor
Generator	: make_sysid()
Print Width	: 10

Physical Field : groundfeature_ground_feature_id (groundfeature_ground_feature_id)

Properties :

Field Type	: sys_id
Mandatory	: True
Editor	: field_editor
Print Width	: 10

Foreign key from groundfeature : field ground_feature_id

Physical Field : groundfeature_type_ground_featur (type)

Properties :

Field Type	: type_ground_feature
Mandatory	: True
Default Value	: "undefined"
Print Width	: 10

Foreign key from groundfeature : field type_ground_feature

Geometric Field : line (updated line)

Properties :

Geometry Type	: chain
Manifold	: boundary_making
Mandatory	: False

Geometric Field : point (updated point)

Properties :

Geometry Type	: point
Manifold	: boundary_making
Mandatory	: False

Geometric Field : area (updated area)

Properties :

Geometry Type	: area
Manifold	: boundary_making

Mandatory	: False

Join Field : groundfeature (groundfeature)

Properties :

Join Object	: groundfeature
Editor	: follow_field_editor
Cascade Delete?	: False
Aspect Field	: ground_feature_id

Join Field : perambulations (perambulation)

Properties :

Join Object	: perambulation
Editor	: component_field_editor
Cascade Delete?	: False

Object name : draftboundary (DraftBoundary)

Properties :

Editor: component_editor
Visible Fields: (default)

: mereing_point
: line
: length
: description_draft_boundary
: type_draft_boundary
: delimitations

Superstructure	: 0:n to allocation
Substructure	: 1:n with delimitation

Field Summary :

Physical Fields	Field Type	Man?	Default	Unset
* draft_boundary_id	sys_id	True		
V description_draft_boundary	description_draft_boundary	False		not applil
V type_draft_boundary	type_draft_boundary	True		

Logical Fields	Field Type
V length	ds_float

Geometric Fields	Field Type	Man?	Manifold
V mereing_point	point	False	boundary_making
V line	chain	False	boundary_making

Join Fields	Join Object
allocation	allocation
V delimitations	delimitation

Field Properties :

Physical Field : draft_boundary_id (draft boundary id)

Properties :

Key Field	:
Field Type	: sys_id
Mandatory	: True
Editor	: field_editor
Generator	: make_sysid()
Print Width	: 10

Physical Field : description_draft_boundary (description)

Properties :

Field Type	: description_draft_boundary
Mandatory	: False
Unset Value	: "not applicable"
Print Width	: 10

Physical Field : type_draft_boundary (type)

Properties :

Field Type	: type_draft_boundary
Mandatory	: True
Print Width	: 10

Logical Field : length (length)

Properties :

Field Type	: ds_float
Print Width	: 10

Geometric Field : mereing_point (mereing point)

Properties :

Geometry Type	: point
Manifold	: boundary_making
Mandatory	: False

Geometric Field : line (line)

Properties :

Geometry Type	: chain
Manifold	: boundary_making
Mandatory	: False

Join Field : allocation (allocation)

Properties :

Join Object	: allocation
Cascade Delete?	: False
Aspect Field	: allocation_id

Join Field : delimitations (delimitation)

Properties :

Join Object	: delimitation
Editor	: component_field_editor
Cascade Delete?	: True

Object name : groundfeature (GroundFeature)

Properties :

Editor: component_editor
Visible Fields: (default)

: point
: line
: area
: type_ground_feature
: groundfeaturerevolutionarystates

Superstructure	: 0:n to assumption
	: 0:n to allocation
Substructure	: 1:n with groundfeaturerevolutionarystate

Field Summary :

Physical Fields	Field Type	Man?	Default	Unset
* ground_feature_id	sys_id	True		
* V type_ground_feature	type_ground_feature	True	undefined	

Geometric Fields	Field Type	Man?	Manifold
V point	point	False	boundary_making
V line	chain	False	boundary_making
V area	area	False	boundary_making

Join Fields	Join Object
assumption	assumption
allocation	allocation
V groundfeaturerevolutionarl	groundfeaturerevolutionarystate

Field Properties :

Physical Field : ground_feature_id (ground feature id)

Properties :

Key Field	:
Field Type	: sys_id

Mandatory	: True
Editor	: field_editor
Generator	: make_sysid()
Print Width	: 10

Physical Field : type_ground_feature (type)

Properties :

Key Field	:
Field Type	: type_ground_feature
Mandatory	: True
Default Value	: "undefined"
Print Width	: 10

Geometric Field : point (point)

Properties :

Geometry Type	: point
Manifold	: boundary_making
Mandatory	: False

Geometric Field : line (line)

Properties :

Geometry Type	: chain
Manifold	: boundary_making
Mandatory	: False

Geometric Field : area (area)

Properties :

Geometry Type	: area
Manifold	: boundary_making
Mandatory	: False

Join Field : assumption (assumption)

Properties :

Join Object	: assumption
Cascade Delete?	: False
Aspect Field	: assumption_id

Join Field : allocation (allocation)

Properties :

Join Object	: allocation
Cascade Delete?	: False
Aspect Field	: allocation_id

Join Field : groundfeaturerevolutionarystates (update procedure)

Properties :

Join Object	: groundfeaturerevolutionarystate
Editor	: component_field_editor
Cascade Delete?	: True

Object name : perambulation (Perambulation)

Properties :

Editor: component_editor
Visible Fields: (default)

: surveyor_name
: date

Superstructure	: 0:n to oldboundaryrevolutionarystate
	: 0:n to groundfeaturerevolutionarystate

Field Summary :

Physical Fields	Field Type	Man?	Default	Unset
* perambulation_id	sys_id	True		
V surveyor_name	ds_char_vec (30)	False	Unknown	
V date	ds_date	False	01/01/70	

Join Fields	Join Object
groundfeaturerevolutionarl	groundfeaturerevolutionarystate
oldboundaryrevolutionarysl	oldboundaryrevolutionarystate

Field Properties :

Physical Field : perambulation_id (perambulation id)

Properties :

Key Field	:
Field Type	: sys_id
Mandatory	: True
Editor	: field_editor
Generator	: make_sysid()
Print Width	: 10

Physical Field : surveyor_name (surveyor name)

Properties :

Field Type	: ds_char_vec(30)
Mandatory	: False
Unset Value	: "Unknown"
Print Width	: 30

Physical Field : date (date)

Properties :

Field Type	: ds_date
Mandatory	: False
Unset Value	: date
Print Width	: 10

Join Field : groundfeaturerevolutionarystate (groundfeaturerevolutionarystate)

Properties :

Join Object	: groundfeaturerevolutionarystate
Cascade Delete?	: False
Aspect Field	: ground_feature_revolutionary_id

Join Field : oldboundaryrevolutionarystate (oldboundaryrevolutionarystate)

Properties :

Join Object	: oldboundaryrevolutionarystate
Cascade Delete?	: False
Aspect Field	: old_boundary_revolutionary_id

Object name : motif (Motif)

Properties :

Visible Fields: (default)

: motif_id

Field Summary :

Physical Fields	Field Type	Man?	Default	Unset
* V motif_id	sys_id	True		

Field Properties :

Physical Field: motif_id (motif id)

Properties :

Key Field	:
Field Type	: sys_id
Mandatory	: True
Editor	: field_editor
Generator	: make_sysid()
Print Width	: 10

Object name : oldboundary (OldBoundary)

Properties :

Editor: component_editor
Visible Fields: (default)

: type_old_boundary

: obsoleteboundarys
: line
: oldboundaryrevolutionarystates

Superstructure	: 1:n to demarcation
Substructure	: 1:n with obsoleteboundary
	: 1:n with oldboundaryrevolutionarystate

Field Summary :

Physical Fields	Field Type	Man?	Default	Unset
* old_boundary_id	sys_id	True		
* V type_old_boundary	type_old_boundary	True		
I demarcation_demarcation_id	sys_id	True		

Geometric Fields	Field Type	Man?	Manifold
V line	chain	False	boundary_making

Join Fields	Join Object
demarcation	demarcation
V obsoleteboundarys	obsoleteboundary
V oldboundaryrevolutionarysl	oldboundaryrevolutionarystate

Field Properties :

Physical Field : old_boundary_id (old boundary id)

Properties :

Key Field	:
Field Type	: sys_id
Mandatory	: True
Editor	: field_editor
Generator	: make_sysid()
Print Width	: 10

Physical Field : type_old_boundary (type)

Properties :

Key Field	:
Field Type	: type_old_boundary
Mandatory	: True
Print Width	: 10

Physical Field : demarcation_demarcation_id (demarcation_demarcation_id)

Properties :

Field Type	: sys_id
Mandatory	: True
Editor	: field_editor
Print Width	: 10

Foreign key from demarcation : field demarcation_id

Geometric Field : line (line)

Properties :

Geometry Type	: chain
Manifold	: boundary_making
Mandatory	: False

Join Field : demarcation (demarcation)

Properties :

Join Object	: demarcation
Editor	: follow_field_editor
Cascade Delete?	: False
Aspect Field	: demarcation_id

Join Field : obsoleteboundarys (archive)

Properties :

Join Object	: obsoleteboundary
Editor	: component_field_editor
Cascade Delete?	: True

Join Field : oldboundaryrevolutionarystates (update procedure)

Properties :

Join Object	: oldboundaryrevolutionarystate
Editor	: component_field_editor
Cascade Delete?	: True

Object name : obsoleteboundary (ObsoleteBoundary)

Properties :

Editor: component_editor
Visible Fields: (default)

: oldboundary_type_old_boundary
: line

Superstructure : 1:n to oldboundary

Field Summary :

Physical Fields	Field Type	Man?	Default	Unset
* obsolete_boundary_id	sys_id	True		
V I oldboundary_type_old_bounl	type_old_boundary	True		
I oldboundary_old_boundary_l	sys_id	True		

Geometric Fields	Field Type	Man?	Manifold

V line	chain	False	boundary_making

Join Fields	Join Object
oldboundary	oldboundary

<u>Field Properties :</u>

Physical Field : obsolete_boundary_id (obsolete boundary id)

Properties :

Key Field	:
Field Type	: sys_id
Mandatory	: True
Edito	: field_editor
Generator	: make_sysid()
Print Width	: 10

Physical Field : oldboundary_type_old_boundary (type)

Properties :

Field Type	: type_old_boundary
Mandatory	: True
Print Width	: 10

Foreign key from oldboundary : field type_old_boundary

Physical Field : oldboundary_old_boundary_id (oldboundary_old_boundary_id)

Properties :

Field Type	: sys_id
Mandatory	: True
Editor	: field_editor
Print Width	: 10

Foreign key from oldboundary : field old_boundary_id

Geometric Field : line (line)

Properties :

Geometry Type	: chain
Manifold	: boundary_making
Mandatory	: False

Join Field : oldboundary (oldboundary)

Properties :

Join Object	: oldboundary
Editor	: follow_field_editor
Cascade Delete?	: False
Aspect Field	: old_boundary_id

Object name : allocation (Allocation)

Properties :

Visible Fields: (default)

: map_scale_used_for
: description
: date
: groundfeatures
: draftboundarys

Substructure	: 0:n with draftboundary
	: 0:n with groundfeature

Field Summary :

Physical Fields		Field Type	Man?	Default	Unset
*	allocation_id	sys_id	True		
V	map_scale_used_for	ds_char_vec (20)	False	Undefined	Undefined
V	description	ds_charci_vec (50)	False		
V	date	ds_date	True		

Join Fields		Join Object
V	groundfeatures	groundfeature
V	draftboundarys	draftboundary

Field Properties :

Physical Field : allocation_id (allocation id)

Properties :

Key Field	:
Field Type	: sys_id
Mandatory	: True
Editor	: field_editor
Generator	: make_sysid()
Print Width	: 10

Physical Field : map_scale_used_for (map scale used for)

Properties :

Field Type	: ds_char_vec(20)
Mandatory	: False
Default Value	: "Undefined"
Unset Value	: "Undefined"
Print Width	: 20

Physical Field : description (description)

Properties :

Field Type	: ds_charci_vec(50)

Mandatory	: False
Unset Value	: ""
Print Width	: 50

Physical Field : date (date)

Properties :

Field Type	: ds_date
Mandatory	: True
Print Width	: 10

Join Field : groundfeatures (ground features)

Properties :

Join Object	: groundfeature
Editor	: component_field_editor
Cascade Delete?	: False

Join Field : draftboundarys (draft boundary)

Properties :

Join Object	: draftboundary
Editor	: component_field_editor
Cascade Delete?	: False

Object name : delimitation (Delimitation)

Properties :

Editor: component_editor
Visible Fields: (default)

: statutory_document
: map_scale_used_for
: operative_date
: effective_date
: actual_date
: newboundarys

Superstructure	: 1:n to draftboundary
Substructure	: 1:n with newboundary

Field Summary :

Physical Fields	Field Type	Man?	Default	Unset
* delimitation_id	sys_id	True		
V statutory_document	statutory_document	True		
V map_scale_used_for	ds_char_vec (20)	False	Undefined	Undefined
V operative_date	ds_date	True		
V effective_date	ds_date	False	01/01/01	
V actual_date	ds_date	False	01/01/01	
I draftboundary_draft_boundl	sys_id	True		

Join Fields	Join Object
draftboundary	draftboundary
V newboundarys	newboundary

Field Properties :

Physical Field : delimitation_id (delimitation id)

Properties :

Key Field	:
Field Type	: sys_id
Mandatory	: True
Editor	: field_editor
Generator	: make_sysid()
Print Width	: 10

Physical Field : statutory_document (statutory document)

Properties :

Field Type	: statutory_document
Mandatory	: True
Print Width	: 10

Physical Field : map_scale_used_for (map scale used for)

Properties :

Field Type	: ds_char_vec(20)
Mandatory	: False
Default Value	: "Undefined"
Unset Value	: "Undefined"
Print Width	: 20

Physical Field : operative_date (operative date)

Properties :

Field Type	: ds_date
Mandatory	: True
Print Width	: 10

Physical Field : effective_date (effective date)

Properties :

Field Type	: ds_date
Mandatory	: False
Unset Value	:date.new_from_days_and_milliseconds(25202,43200000)
Print Width	: 10

Physical Field : actual_date (actual date)

Properties :

Field Type	: ds_date
Mandatory	: False
Unset Value	: date.new_from_days_and_milliseconds(25202,43200000)
Print Width	: 10

Physical Field : draftboundary_draft_boundary_id (draftboundary_draft_boundary_id)

Properties :

Field Type	: sys_id
Mandatory	: True
Editor	: field_editor
Print Width	: 10

Foreign key from draftboundary : field draft_boundary_id

Join Field : draftboundary (draftboundary)

Properties :

Join Object	: draftboundary
Editor	: follow_field_editor
Cascade Delete?	: False
Aspect Field	: draft_boundary_id

Join Field : newboundarys (new boundary)

Properties :

Join Object	: newboundary
Editor	: component_field_editor
Cascade Delete?	: True

Object name : publicboundary (PublicBoundary)

Properties :

Visible Fields: (default)

: public_boundary_id

Field Summary :

Physical Fields	Field Type	Man?	Default	Unset
* V public_boundary_id	sys_id	True		

Field Properties :

Physical Field : public_boundary_id (public boundary id)

Properties :

Key Field	:
Field Type	: sys_id
Mandatory	: True
Editor	: field_editor

Generator	: make_sysid()
Print Width	: 10

Object name : oldboundaryrevolutionarystate (OldBoundaryRevolutionaryState)

Properties :

Editor: component_editor
Visible Fields: (default)

: line
: perambulations

Superstructure	: 1:n to oldboundary
Substructure	: 0:n with perambulation

Field Summary :

Physical Fields	Field Type	Man?	Default	Unset
* old_boundary_revolutionarl	sys_id	True		
I oldboundary_type_old_bounl	type_old_boundary	True		
I oldboundary_old_boundary_l	sys_id	True		

Geometric Fields	Field Type	Man?	Manifold
V line	chain	False	boundary_making

Join Fields	Join Object
V perambulations	perambulation
oldboundary	oldboundary

Field Properties :

Physical Field : old_boundary_revolutionary_id (old boundary revolutionary id)

Properties :

Key Field	:
Field Type	: sys_id
Mandatory	: True
Editor	: field_editor
Generator	: make_sysid()
Print Width	: 10

Physical Field : oldboundary_type_old_boundary (oldboundary_type_old_boundary)

Properties :

Field Type	: type_old_boundary
Mandatory	: True
Print Width	: 10

Foreign key from oldboundary : field type_old_boundary

Physical Field : oldboundary_old_boundary_id (oldboundary_old_boundary_id)

Properties :

Field Type	: sys_id
Mandatory	: True
Editor	: field_editor
Print Width	: 10

Foreign key from oldboundary : field old_boundary_id

Geometric Field : line (updated line)

Properties :

Geometry Type	: chain
Manifold	: boundary_making
Mandatory	: False

Join Field : perambulations (perambulation)

Properties :

Join Object	: perambulation
Editor	: component_field_editor
Cascade Delete?	: False

Join Field : oldboundary (oldboundary)

Properties :

Join Object	: oldboundary
Editor	: follow_field_editor
Cascade Delete?	: False
Aspect Field	: old_boundary_id

Object name : assumption (Assumption)

Properties :

Visible Fields: (default)

: statement
: validated
: valid_from
: valid_to
: groundfeatures

Substructure : 0:n with groundfeature

Field Summary :

	Physical Fields	Field Type	Man?	Default	Unset
*	assumption_id	sys_id	True		
V	statement	ds_charci_vec (60)	True		
V	validated	ds_bool	True		
V	valid_from	ds_date	True		

V valid_to	ds_date	False	01/01/70

Join Fields	Join Object
V groundfeatures	groundfeature

Field Properties :

Physical Field : assumption_id (assumption id)

Properties :

Key Field	:
Field Type	: sys_id
Mandatory	: True
Editor	: field_editor
Generator	: make_sysid()
Print Width	: 10

Physical Field : statement (statement)

Properties :

Field Type	: ds_charci_vec(60)
Mandatory	: True
Print Width	: 60

Physical Field : validated (validated)

Properties :

Field Type	: ds_bool
Mandatory	: True
Print Width	: 10

Physical Field : valid_from (valid from)

Properties :

Field Type	: ds_date
Mandatory	: True
Print Width	: 10

Physical Field : valid_to (valid to)

Properties :

Field Type	: ds_date
Mandatory	: False
Unset Value	: date
Print Width	: 10

Join Field : groundfeatures (ground features)

Properties :

Join Object	: groundfeature

Editor	: component_field_editor
Cascade Delete?	: False

Object name : boundarydisplayutilities (BoundaryDisplayUtilities)

Properties :

Visible Fields: (default)

: boundary_display_utilities_id

Field Summary :

Physical Fields	Field Type	Man?	Default	Unset
* V boundary_display_utilitiel	sys_id	True		

Field Properties :

Physical Field : boundary_display_utilities_id (boundary display utilities id)

Properties :

Key Field	:
Field Type	: sys_id
Mandatory	: True
Editor	: field_editor
Generator	: make_sysid()
Print Width	: 10

Object name : demarcation (Demarcation)

Properties :

Editor: component_editor
Visible Fields: (default)

: surveyor_name
: map_scale_used_for
: valid_from
: valid_to
: oldboundarys

Superstructure	: 1:n to newboundary
Substructure	: 1:n with oldboundary

Field Summary :

Physical Fields	Field Type	Man?	Default	Unset
* demarcation_id	sys_id	True		
V surveyor_name	ds_char_vec (40)	False	Unknown	Unknown
V map_scale_used_for	ds_char_vec (20)	False	Undefined	Undefined
V valid_from	ds_date	True		
V valid_to	ds_date	False		01/01/70
I newboundary_new_boundary_l	sys_id	True		

Join Fields	Join Object
newboundary	newboundary
V oldboundarys	oldboundary

Field Properties :

Physical Field : demarcation_id (demarcation id)

Properties :

Key Field:	
Field Type	: sys_id
Mandatory	: True
Editor	: field_editor
Generator	: make_sysid()
Print Width	: 10

Physical Field : surveyor_name (surveyor name)

Properties :

Field Type	: ds_char_vec(40)
Mandatory	: False
Default Value	: "Unknown"
Unset Value	: "Unknown"
Print Width	: 40

Physical Field : map_scale_used_for (map scale used for)

Properties :

Field Type	: ds_char_vec(20)
Mandatory	: False
Default Value	: "Undefined"
Unset Value	: "Undefined"
Print Width	: 20

Physical Field : valid_from (valid from)

Properties :

Field Type	: ds_date
Mandatory	: True
Print Width	: 10

Physical Field : valid_to (valid to)

Properties :

Field Type	: ds_date
Mandatory	: False
Unset Value	: date
Print Width	: 10

Physical Field : newboundary_new_boundary_id (newboundary_new_boundary_id)

Properties :

Field Type	: sys_id
Mandatory	: True
Editor	: field_editor
Print Width	: 10

Foreign key from newboundary : field new_boundary_id

Join Field : newboundary (newboundary)

Properties :

Join Object	: newboundary
Editor	: follow_field_editor
Cascade Delete?	: False
Aspect Field	: new_boundary_id

Join Field : oldboundarys (old boundaries)

Properties :

Join Object	: oldboundary
Editor	: component_field_editor
Cascade Delete?	: True

Object name : historical_view_2 (Delimitation Historical View)

<u>Field Summary :</u>

Physical Fields	Field Type	Man?	Default	Unset
* I draft_boundary_id	sys_id	True		
V I description_draft_boundary	description_draft_boundary	False		not applil
V I type_draft_boundary	type_draft_boundary	True		
V I delimitation_statutory_dol	statutory_document	True		
V I delimitation_map_scale_usl	ds_char_vec (20)	False	Undefined	Undefined
V I delimitation_operative_dal	ds_date	True		
V I delimitation_effective_dal	ds_date	False	01/01/01	
V I delimitation_actual_date	ds_date	False	01/01/01	
I delimitation_delimitationl	sys_id	True		
V I newboundary_description_nl	description_new_boundary	True		
I newboundary_new_boundary_l	sys_id	True		

Logical Fields	Field Type
V I length	ds_float
V I newboundary_length	ds_float

Geometric Fields	Field Type	Man?	Manifold
V I mereing_point	point	False	boundary_making
V I line	chain	False	boundary_making
V I newboundary_line	chain	False	boundary_making
V I newboundary_turning_point	point	False	boundary_making

<u>Field Properties :</u>

Physical Field : draft_boundary_id (draft boundary id)

Properties :

Key Field	:
Field Type	: sys_id
Mandatory	: True
Editor	: field_editor
Print Width	: 10

View field based on field draft_boundary_id in object draftboundary

Physical Field : description_draft_boundary (description draft boundary)

Properties :

Field Type	: description_draft_boundary
Mandatory	: False
Unset Value	: "not applicable"
Print Width	: 10

View field based on field description_draft_boundary in object draftboundary

Physical Field : type_draft_boundary (type draft boundary)

Properties :

Field Type	: type_draft_boundary
Mandatory	: True
Print Width	: 10

View field based on field type_draft_boundary in object draftboundary

Physical Field : delimitation_statutory_document (delimitation statutory document)

Properties :

Field Type	: statutory_document
Mandatory	: True
Print Width	: 10

View field based on field statutory_document in object delimitation

Physical Field : delimitation_map_scale_used_for (delimitation map scale)

Properties :

Field Type	: ds_char_vec(20)
Mandatory	: False
Default Value	: "Undefined"
Unset Value	: "Undefined"
Print Width	: 20

View field based on field map_scale_used_for in object delimitation

Physical Field : delimitation_operative_date (delimitation operative date)

Properties :

Field Type	: ds_date
Mandatory	: True
Print Width	: 10

View field based on field operative_date in object delimitation

Physical Field : delimitation_effective_date (delimitation effective date)

Properties :

Field Type	: ds_date
Mandatory	: False
Unset Value	: date.new_from_days_and_milliseconds(25202,43200000)
Print Width	: 10

View field based on field effective_date in object delimitation

Physical Field : delimitation_actual_date (delimitation actual date)

Properties :

Field Type	: ds_date
Mandatory	: False
Unset Value	: date.new_from_days_and_milliseconds(25202,43200000)
Print Width	: 10

View field based on field actual_date in object delimitation

Physical Field : delimitation_delimitation_id (delimitation_delimitation_id)

Properties :

Field Type	: sys_id
Mandatory	: True
Editor	: field_editor
Print Width	: 10

View field based on field delimitation_id in object delimitation

Physical Field : newboundary_description_new_boun (new boundary description)

Properties :

Field Type	: description_new_boundary
Mandatory	: True
Print Width	: 10

View field based on field description_new_boundary in object newboundary

Physical Field : newboundary_new_boundary_id (newboundary_new_boundary_id)

Properties :

Field Type	: sys_id

Mandatory	: True
Editor	: field_editor
Print Width	: 10

View field based on field new_boundary_id in object newboundary

Logical Field : length (draft boundary length)

Properties :

Field Type	: ds_float
Print Width	: 10

View field based on field length in object draftboundary

Logical Field : newboundary_length (new boundary length)

Properties :

Field Type	: ds_float
Print Width	: 10

View field based on field length in object newboundary

Geometric Field : mereing_point (mereing point)

Properties :

Geometry Type	: point
Manifold	: boundary_making
Mandatory	: False

View field based on field mereing_point in object draftboundary

Geometric Field : line (draft boundary line)

Properties :

Geometry Type	: chain
Manifold	: boundary_making
Mandatory	: False

View field based on field line in object draftboundary

Geometric Field : newboundary_line (new boundary line)

Properties :

Geometry Type	: chain
Manifold	: boundary_making
Mandatory	: False

View field based on field line in object newboundary

Geometric Field : newboundary_turning_point (new boundary turning point)

Properties :

Geometry Type	: point
Manifold	: boundary_making
Mandatory	: False

View field based on field turning_point in object newboundary

Object name : update_historical_view (Update Historical View)

Field Summary :

Physical Fields	Field Type	Man?	Default	Unset
* V I type_old_boundary	type_old_boundary	True		
* I old_boundary_id	sys_id	True		
I demarcation_demarcation_id	sys_id	True		
I oldboundaryrevolutionarysl	sys_id	True		
I perambulation_perambulatil	sys_id	True		
V I perambulation_surveyor_nal	ds_char_vec (30)	False		Unknown
V I perambulation_date	ds_date	False		01/01/70
I groundfeaturerevolutionarl	sys_id	True		
I groundfeature_ground_featl	sys_id	True		
V I groundfeature_type_groundl	type_ground_feature	True	undefined	

Geometric Fields	Field Type	Man?	Manifold
V I line	chain	False	boundary_making
V I oldboundaryrevolutionarysl	chain	False	boundary_making
V I groundfeaturerevolutionarl	chain	False	boundary_making
V I groundfeaturerevolutionarl	point	False	boundary_making
I groundfeaturerevolutionarl	area	False	boundary_making
V I groundfeature_point	point	False	boundary_making
V I groundfeature_line	chain	False	boundary_making
I groundfeature_area	area	False	boundary_making

Field Properties :

Physical Field : type_old_boundary (old boundary type)

Properties :

Key Field	:
Field Type	: type_old_boundary
Mandatory	: True
Print Width	: 10

View field based on field type_old_boundary in object oldboundary

Physical Field : old_boundary_id (old boundary id)

Properties :

Key Field	:
Field Type	: sys_id
Mandatory	: True
Editor	: field_editor
Print Width	: 10

View field based on field old_boundary_id in object oldboundary

Physical Field : demarcation_demarcation_id (demarcation_demarcation_id)

Properties :

Field Type	: sys_id
Mandatory	: True
Editor	: field_editor
Print Width	: 10

View field based on field demarcation_demarcation_id in object oldboundary

Physical Field : oldboundaryrevolutionarystate_ol (oldboundaryrevolutionarystate_ol)

Properties :

Field Type	: sys_id
Mandatory	: True
Editor	: field_editor
Print Width	: 10

View field based on field old_boundary_revolutionary_id in object oldboundaryrevolutionarystate

Physical Field : perambulation_perambulation_id (perambulation_perambulation_id)

Properties :

Field Type	: sys_id
Mandatory	: True
Editor	: field_editor
Print Width	: 10

View field based on field perambulation_id in object perambulation

Physical Field : perambulation_surveyor_name (perambulation surveyor name)

Properties :

Field Type	: ds_char_vec(30)
Mandatory	: False
Unset Value	: "Unknown"
Print Width	: 30

View field based on field surveyor_name in object perambulation

Physical Field : perambulation_date (perambulation date)

Properties :

Field Type	: ds_date
Mandatory	: False
Unset Value	: date
Print Width	: 10

View field based on field date in object perambulation

Physical Field : groundfeaturerevolutionarystate_ (groundfeaturerevolutionarystate_)

Properties :

Field Type	: sys_id
Mandatory	: True
Editor	: field_editor
Print Width	: 10

View field based on field ground_feature_revolutionary_id in object groundfeaturerevolutionarystate

Physical Field : groundfeature_ground_feature_id (groundfeature_ground_feature_id)

Properties :

Field Type	: sys_id
Mandatory	: True
Editor	: field_editor
Print Width	: 10

View field based on field ground_feature_id in object groundfeature

Physical Field : groundfeature_type_ground_featur (ground feature type)

Properties :

Field Type	: type_ground_feature
Mandatory	: True
Default Value	: "undefined"
Print Width	: 10

View field based on field type_ground_feature in object groundfeature

Geometric Field : line (old boundary line)

Properties :

Geometry Type	: chain
Manifold	: boundary_making
Mandatory	: False

View field based on field line in object oldboundary

Geometric Field : oldboundaryrevolutionarystate_li (old boundary rev state line)

Properties :

Geometry Type	: chain
Manifold	: boundary_making
Mandatory	: False

View field based on field line in object oldboundaryrevolutionarystate

Geometric Field : groundfeaturerevolutionarystatel (ground feature rev state line)

Properties :

Geometry Type	: chain
Manifold	: boundary_making
Mandatory	: False

View field based on field line in object groundfeaturerevolutionarystate

Geometric Field : groundfeaturerevolutionarystate2 (ground feature rev state point)

Properties :

Geometry Type	: point
Manifold	: boundary_making
Mandatory	: False

View field based on field point in object groundfeaturerevolutionarystate

Geometric Field : groundfeaturerevolutionarystate3 (groundfeaturerevolutionarystate3)

Properties :

Geometry Type	: area
Manifold	: boundary_making
Mandatory	: False

View field based on field area in object groundfeaturerevolutionarystate

Geometric Field : groundfeature_point (ground feature point)

Properties :

Geometry Type	: point
Manifold	: boundary_making
Mandatory	: False

View field based on field point in object groundfeature

Geometric Field : groundfeature_line (ground feature line)

Properties :

Geometry Type	: chain
Manifold	: boundary_making
Mandatory	: False

View field based on field line in object groundfeature

Geometric Field : groundfeature_area (groundfeature_area)

Properties :

Geometry Type	: area
Manifold	: boundary_making
Mandatory	: False

View field based on field area in object groundfeature

Object name : historical_view_3 (Demarcation Historical View)

Field Summary :

Physical Fields	Field Type	Man?	Default	Unset
V I description_new_boundary	description_new_boundary	True		
* I new_boundary_id	sys_id	True		
I delimitation_delimitationl	sys_id	True		
V I demarcation_surveyor_name	ds_char_vec (40)	False	Unknown	Unknown
V I demarcation_map_scale_usel	ds_char_vec (	False	Undefined	Undefined
I demarcation_demarcation_id	sys_id	True		
V I demarcation_valid_from	ds_date	True		
V I demarcation_valid_to	ds_date	False		01/01/70
V I oldboundary_type_old_bounl	type_old_boundary	True		
I oldboundary_old_boundary_l	sys_id	True		

Logical Fields	Field Type
V I length	ds_float

Geometric Fields	Field Type	Man?	Manifold
I line	chain	False	boundary_making
V I turning_point	point	False	boundary_making
V I oldboundary_line	chain	False	boundary_making

Field Properties :

Physical Field : description_new_boundary (new boundary description)

Properties :

Field Type	: description_new_boundary
Mandatory	: True
Print Width	: 10

View field based on field description_new_boundary in object newboundary

Physical Field : new_boundary_id (new boundary id)

Properties :

Key Field	:
Field Type	: sys_id
Mandatory	: True
Editor	: field_editor
Print Width	: 10

View field based on field new_boundary_id in object newboundary

Physical Field : delimitation_delimitation_id (delimitation_delimitation_id)

Properties :

Field Type	: sys_id
Mandatory	: True
Editor	: field_editor

Print Width : 10

View field based on field delimitation_delimitation_id in object newboundary

Physical Field : demarcation_surveyor_name (surveyor name)

Properties :

Field Type : ds_char_vec(40)
Mandatory : False
Default Value : "Unknown"
Unset Value : "Unknown"
Print Width : 40
View field based on field surveyor_name in object demarcation

Physical Field : demarcation_map_scale_used_for (demarcation map scale)

Properties :

Field Type : ds_char_vec(20)
Mandatory : False
Default Value : "Undefined"
Unset Value : "Undefined"
Print Width : 20

View field based on field map_scale_used_for in object demarcation

Physical Field : demarcation_demarcation_id (demarcation_demarcation_id)

Properties :

Field Type : sys_id
Mandatory : True
Editor : field_editor
Print Width : 10

View field based on field demarcation_id in object demarcation

Physical Field : demarcation_valid_from (demarcation valid from)

Properties :

Field Type : ds_date
Mandatory : True
Print Width : 10

View field based on field valid_from in object demarcation

Physical Field : demarcation_valid_to (demarcation valid to)

Properties :

Field Type : ds_date
Mandatory : False
Unset Value : date
Print Width : 10

View field based on field valid_to in object demarcation

Physical Field : oldboundary_type_old_boundary (old boundary type)

Properties :

Field Type	: type_old_boundary
Mandatory	: True
Print Width	: 10

View field based on field type_old_boundary in object oldboundary

Physical Field : oldboundary_old_boundary_id (oldboundary_old_boundary_id)

Properties :

Field Type	: sys_id
Mandatory	: True
Editor	: field_editor
Print Width	: 10

View field based on field old_boundary_id in object oldboundary

Logical Field : length (length)

Properties :

Field Type	: ds_float
Print Width	: 10

View field based on field length in object newboundary

Geometric Field : line (line)

Properties :

Geometry Type	: chain
Manifold	: boundary_making
Mandatory	: False

View field based on field line in object newboundary

Geometric Field : turning_point (turning point)

Properties :

Geometry Type	: point
Manifold	: boundary_making
Mandatory	: False

View field based on field turning_point in object newboundary

Geometric Field : oldboundary_line (old boundary line)

Properties :

Geometry Type	: chain

Manifold	: boundary_making
Mandatory	: False

View field based on field line in object oldboundary

PART II : METHODS

```
_method int!world.universe

## gets the universe record of this world

          >> _self.source_view.collections [:sw_gis!world_type].at(_self.universe_id)

_endmethod
$

_method int!world.scan_key

## This returns the scan-key for use in the C scanner.

          _if scan_key _is _unset
          _then
                    ur << _self.universe

                    nu << ur.n_bits_universe
                    nw << ur.n_bits_world
                       scan_key << _self.universe_id.shift (32-nu) _or
                                   _self.world_id.shift(32-nu-nw)
          _endif
          >> scan_key
_endmethod
$

_method int!world_type.n_universes
          >> _self.source_collection.size
_endmethod
$


_method int!world_type.n_bits_universe
          >> _self.n_universes-1).bit_width
_endmethod
$

_method int!world.unit_system
          units << unit_system.factors_to_name{self.unit_factor]
          >> unit_system.new(units,units)
_endmethod
$

_pragma(classify_level=advanced, topic={case}, usage={internal}
_method join_relationship.ds_field
#
## This is the update trigger for a ds_field
          if !current_transform! _is _unset
```

```
        _then
                _return
        _endif
        _if (ds_field << _self.ds_field(:value)) _is _unset
        _then
                value.transform(!current_transform!)
        _endif
_endmethod
$

_pragma(classify_level=advanced, topic={case}, usage={internal}
_method join_relationship.ds_record
#
## This is the update trigger for a ds_record
        if !current_transform! _is _unset
        _then
                _return
        _endif
        _if (ds_record << _self.ds_record(:value)) _is _unset
        _then
                value.transform(!current_transform!)
        _endif
_endmethod

_pragma(classify_level=restricted, topic={case}, usage={external})
_method join_relationship.updated_type_field(fname,trig_type,new_val,old_val}
##
## This method is the incremental modification on the type field in a join relationship (1;n, 1:0)
##
        _if old_val _is _unset
        _then
                _return
        _endif
        _if  (t << old_val.as_charvec()) = :geometric
        _then
                _self.clear_out_geometric_info()
        _elif t = :inheritance _orif t= "1:0" _orif
              t = "1:n" _orif t = "view"
        _then
                _self.clear_out_inheritance_info()
                _self.clear_physical_properties ()
                _self.clear_ot_view_info()
        _else
                _self.clear_physical_properties()
        _endif
        make_def? << _self.make_defaults?(_self.object1,_self.object2)
        _self.clear_out_parent_info()
        _if (t << new_val.as_charvec()) = :inheritance
        _then
                _if (o1 << _self.object1).view?
                _then
                        condition.raise(:cannot_inherit_from_view,
                                        :object_name, o1.name)
                _endif
                _if make_def?
                _then
                        _self.object2.inherit_from_another(_self.o1)
                _endif
        _elif t= :view
        _then
                o2 << _self.object2
                _if _not o2.ok_to_become_view?
                _then
                        condition.raise(:o_cannot_be_view,:object_name,o2.name)
                _endif
                _if make def?
                _then
                        o2.inherit_from_another(_self.object1)
                _endif
        _elif t = "1:0" _orif t="1:n") _andif
                make_def?
        _then
                new_index << _self.object2.add_key_fields_from_another(self.object1,_self)
                _if new_index _isnt _unset
                _then
                        _self add_index(new_index)
                _endif
        _endif
        _if _self.physical?
        _then
                _self.add_default_join_fields()
                _self.add_default_structure_records()
                _if make_def?
                _then
                        _self.add_default_join_names()
                _endif
        _endif
_endmethod
$
```