An Intelligent Geographic Information System For Design

by

Lee John Finniear, B.Sc.

A Doctoral Thesis
Submitted in partial fulfilment of the requirements
for the award of

Doctor of Philosophy

of the Loughborough University of Technology
January, 1991

This work is dedicated to

Daniella

The First of a New Generation

Born : 2nd January, 1991

Dear Daniella. You're only two weeks old now, but by the time you get to read this you will probably be in your 'teens. I hope you are going out, enjoying life and having a wild time. As I write this your parents are not much older than you as you read. They are now going through the same trials, tribulations and joys that you must be going through. The 'Old Man' and 'Old Dear' may seem past it, but remember they have already been where you are. So if at times you think they don't understand, please think again, because deep down they probably do........

Wishing you all the luck, life and happiness this crazy, mixed up world has to give

Lee

TABLE OF CONTENTS

ABSTRACT

Recent advances in geographic information systems (GIS) and artificial intelligence (AI) techniques have been summarised, concentrating on the theoretical aspects of their construction and use. Existing projects combining AI and GIS have also been discussed, with attention paid to the interfacing methods used and problems uncovered by the approaches. AI and GIS have been combined in this research to create an *intelligent GIS for design*. This has been applied to off-shore pipeline route design. The system was tested using data from a real pipeline design project.

Engineering design often involves the use of large quantities of geographic and spatial data. Despite this geographic information systems (GIS) have failed to become a worthwhile design tool. The results from conventional GIS, it appears, do not justify their use in a design context. This research explores the possibility of adding *intelligence* to a GIS, such that it can *understand* the data it holds in the context of the design problem, and thus make a more effective contribution to the solution being sought.

PIRATE is an intelligent GIS for design, specifically applied to the design of off-shore pipeline routes. By integrating a GIS with an AI toolkit, PIRATE enables knowledge base rules to directly access and manipulate geographic features in the GIS. AI/GIS integration is ensured by the use of spatial dualism, with the location based feature description in the GIS, and the object based description in the frame hierarchy of the AI toolkit. The system requirements and design rules were found as a result of extensive interviews with practicing pipeline engineers. PIRATE is operated from a graphic user interface which allows overlay mapping, GIS interrogation, interactive pipeline design and assessment. Logical dependency allows the system to be used as a *geographic design spreadsheet*, providing the engineer with a rapid tool for optimising his design.

PIRATE was tested using charts and data from the Sable Island Gas Pipeline Project, a route requirement over a bathymetrically complex ocean floor on the Nova Scotian continental shelf. PIRATE was loaded with the bathymetry and a subset of the features in the region, and proved itself effective in analysing routes crossing the region. Recommendations have been made for expanding the PIRATE system, and for the development and use of intelligent GIS for other applications.

DECLARATION

No portion of the research referred to in this thesis has been submitted in support of an application for another degree or qualification at this or any other university or other institution of learning.

ACKNOWLEDGEMENTS

I would like to give my heartfelt thanks to John Dickens and Dr. John Strodachs, my research supervisors, for their friendship, guidance and understanding throughout the years of my research.

My gratitude is extended to British Gas plc., who together with the Science and Engineering Research Council, provided financial support under a Co-operative Award in Science and Engineering. In particular I am indebted to Tony Nixon and George Simons, who unwaveringly gave their support throughout the study period.

My thanks are also extended Miles Jones and the other staff of J.P. Kenny & Partners for their assistance.

I owe a sincere debt of gratitude to all members of KBI and Associates, who's capable advice has guided me through the most difficult periods in my years at Loughborough. I hope our association continues.

I would like to pay tribute to all members of staff of the Department of Civil Engineering, Loughborough University of Technology, and to the following individuals and organisations who have given freely of their advice and assistance,

All staff at Applications in CADD, Shepshed
The AI Applications Institute, Edinburgh
Bruce Gittings, Department of Geography, Edinburgh University.
P. Baker and Marjorie, Graphics Section, British Gas plc.
Nick Taylor, Managing Director, AEGIS Survey Consultants Limited.

Many thanks must also go to Dr. John Allen, "my third supervisor", whose patience and cheerfulness was unparalleled during his ardent reading and my relentless re-writing of this thesis.

Finally, to my family and friends, who have had to bear the brunt of my temper, exuberance and despondency, thank you all for coping and smiling.

LIST OF CHAPTERS

LIST OF APPENDICES

LIST OF FIGURES

LIST OF PLATES

LIST OF TABLES

CHAPTER 1

1. INTRODUCTION

"Geographic Information Systems [GIS] are the biggest step forward in the handling of geographic information since the invention of the map"
(Department of the Environment, 1987)

Engineering design relies heavily on geographic and spatial information. However, GIS have failed to become a worthwhile design tool. This is because to date GIS have been deliberately constructed as isolated, general purpose tools. They have no links to, or ability to understand the rules, consequences and restrictions which make up 'the design process'.

For centuries geographic information has been stored and displayed in the form of maps and charts. These rely on human abilities to translate the symbols on the map into a mental model of the real situation. This interpretation will depend on the use to which the information is being put, and the past experiences and expertise of the user as he adds meaning to the symbols in terms of their explicit or expected properties.

Computer systems are now being used to replace the conventional paper map as a means of storing and displaying geographic information. With advantages in interrogation speed, capacity and ease of update, GIS have found widespread uses in such diverse areas as forestry management and building society office selection. However, despite being able to present data graphically and in many different forms, GIS generally rely on the human user to interpret the results in the context of the problem being solved. Conventional GIS have neither the intelligence to understand the data they hold nor the ability to act on this understanding.

In recent years research in *artificial intelligence* has yielded a new breed of computer program, called the *expert system*. This is specifically designed to emulate the decision making processes of a human expert. An expert system uses logic to link rules and facts about a particular area of expertise, or *domain*, to produce new information and solutions for any given situation. Applications have been widespread, from hazard monitoring in a nuclear power station, to choosing the best wines for a particular meal.

In this thesis the potential of GIS is extended into the design domain by linking the capabilities of GIS with the rule handling abilities of AI systems. It will be shown that the two widely dissimilar technologies of GIS and AI can be successfully combined, into a commercially viable *intelligent GIS* for design.

The idea of combining GIS with AI is not new, but it has generally been applied to making existing GIS functions perform better, rather than attempting to contribute significantly to the eventual use of the data. The major research projects which have attempted the latter function are discussed in the thesis.

Research with British Gas plc., the sponsors of this work, and J.P. Kenny & Partners, a leading off-shore design consultancy, indicated that existing manual techniques for off-shore pipeline route design were both time consuming and tedious.

During the design process the sub-sea geographic data on which the design is based tends to be imprecise or expensive to collect, and becomes available at unpredictable stages. Producing charts showing geographic constraints and analysing potential routes is time consuming. Late changes to design parameters, for example pipe diameter, can have far reaching effects on all aspects of route assessment. Re-designing the pipeline in the light of such changes requires extensive manual recalculation and re-assessment. The effort needed for manual pipeline design is such that in practice few alternative routes are given full consideration, giving the possibility of a more efficient route being missed.

By integrating expert system technology with a geographic information system, this research has produced an intelligent GIS, called the **Pipeline Route Analysis and Testing Environment (PIRATE)**. By incorporating the design knowledge of practicing pipeline engineers, PIRATE is able not only to store and manipulate geographic information, but to understand it in the context of the problem it is addressing, such that it can make a valid, intelligent contribution to the solution being sought.

PIRATE was tested using information from a real pipeline project. This case study was provided by J.P. Kenny & Partners, and it represents one of the most extensive and complex projects that they have undertaken in recent years. Results from this study proved that PIRATE could significantly reduce the time taken for pipeline route design and assessment. It was concluded from this that the objective of creating a viable intelligent GIS for design had indeed been achieved.

The work presented in this thesis brings together the disciplines of geographic information systems and artificial intelligence. The thesis has been structured to acquaint the reader with essential aspects of each area before their integration is considered.

CHAPTER 2

2. GEOGRAPHIC INFORMATION SYSTEMS

2.1 INTRODUCTION

Geographic information has been stored and accessed for centuries in the form of paper maps and charts. Maps are an invaluable asset for a wide range of human activities, from navigation to land use planning. Traditional methods of map production involve the skills of a cartographer manually drawing each geographic feature. Non-spatial properties of each feature are indicated by the type of symbol the cartographer uses. An area of forest, for example, would perhaps be shaded in green.

With the advent of computer technology capable of storing large volumes of information, computers have been applied to the handling of geographic information. These computer systems are known as *geographic information systems* (GIS).

Throughout the 1980's the number of commercially available GIS has expanded rapidly, prompting the UK Government to commission an enquiry into the potential of these new systems (DoE, 1987,1988). In 1989 the Association for Geographic Information (AGI) was formed to provide a forum for discussion on this new technology. The AGI have issued a comprehensive definition of what constitutes a GIS:

> "A system for handling data which is directly or indirectly spatially referenced to the Earth. It may be used for capturing, storing, validating, maintaining, manipulating, analysing, displaying or managing such data. It is normally considered to involve a spatially referenced computer database and appropriate software. A primary function of a GIS is it's ability to integrate data from a wide variety of sources." (Purvis, 1989).

Commercial GIS software is now available in a wide range of forms, from small systems capable of running on a personal computer, to large corporate mainframe installations. Lists have been published showing the extent and capabilities of competing GIS software (Mapping Awareness, 1989).

Research in GIS is very active at this time. New data storage and manipulation concepts are being investigated, whilst the number of GIS application areas are increasing due to diverse research initiatives. The practical aspects of installing a GIS have also been the subject of numerous research papers, as early implementors find unexpected pitfalls and benefits in GIS use.

This chapter concentrates on the theoretical aspects of GIS data storage and handling, a thorough understanding of which is needed to appreciate the findings of this research. However, the chapter also discusses aspects relating to GIS implementation and use, the capture of geographic data and examples of GIS used in practice.

2.2 Data Storage

Geographic data may be neatly divided into spatial and non-spatial components, the spatial component representing the physical presence of an object in space, and the non-spatial component representing the properties associated with the object. For example, a road has spatial information such as its centre line route and width, and non-spatial information such as the type of surfacing and it's cost per kilometre.

In common with many other computer applications, GIS data is usually physically stored in digital form on a magnetic medium, such as a disk or tape. The way in which this digital data is interpreted by the system, the meaning that is assigned to the content and order of the data elements as they are abstracted from the medium, is governed by the *GIS data model* that the system uses.

A data model is a conceptual framework that gives meaning to data items. It acts on data like camera lenses or filters do on light, focussing it into an understandable image for the user. In a typical GIS there are three layers in the data model. At the lowest level, nearest to the physical data storage is the data model of the database management system (DBMS). This takes control of the physical positioning and physical form of the data on the storage medium, interfacing with the 'user' by presenting the data in a standard format. The relational DBMS, for example, uses rows and columns of a table as a way of presenting data. The DBMS data model also provides specific functions for manipulating the data that appears in this view. Non-spatial data can be accessed by the GIS user directly from the DBMS, but in the case of spatial data the DBMS 'user' is the next layer in the GIS data model. A number of commercial DBMS are available to fulfil this function, though some GIS authors elect to design their own procedures for physical storage control. DBMS for GIS are discussed in Section 2.4.

For spatial data, the layer above the DBMS is the *spatial data model.* This effectively translates the numeric values of the DBMS view into a representation of space. By giving a spatial framework to the data, a coordinate system, the concepts of lines, points, areas, distances etc., the numbers from the DBMS yield spatial entities.

A third layer, still higher in abstraction can be said to exist. This is the *user data model,* which interprets the spatial entities as representing actual geographic objects or characteristics in the real world. The user data model controls how the data is displayed and how it can be manipulated. Spatial and non-spatial data come together in the user data model to provide a more complete depiction of the geographic objects being represented. The eventual product of this model is an understanding in the users mind of what characteristics the 'real world' region actually displays.

The extent to which the user data model is actually implemented as computer code in a GIS varies from system to system. The result of the user data model is a state of understanding in the users mind, and therefore an interface must exist at some point in the model between the system and the user. Physically, of course, this interface is the computer screen, mouse and keyboard. The key issue, however, is the way in which the system presents the data to the user. A GIS may simply give the data as points, lines and areas, for example, with non-spatial information linked to these. Alternatively the GIS could also model the relationships between the spatial elements, and could perhaps allow the creation of complex objects as collections of simple ones. These facilities would lessen the burden on the user in both understanding the geographic information, and manipulating it correctly.

Clearly, if a user could be given instant access to *any* information about a region the situation would be ideal. In practice the storage and management of the potentially infinite data volumes implied in this scenario is impossible with current methods. Collecting the data in the first place, and ensuring it was kept up to date, would be an impossible task.

When an organisation decides to use a GIS it normally has a particular set of duties that it wishes the system to perform. The information stored in the GIS database must be sufficient to fulfil these functions, but should not be in excess of this. Any superfluous data will reduce response times, waste storage space, and misuse time in data collection and update. The decision as to what information to include or omit, and what operations on the data are needed to achieve the desired results, are crucial to the success of the GIS implementation.

When commercial GIS software is written, the authors rarely have knowledge of the actual applications the GIS will be used for. Hence the level of detail that will be required in the database cannot be stated in advance. The types of interrogation or query that the user will need are also unknown. Commercial GIS must therefore provide enough general facilities not only for storing data at any level of detail, but also for manipulating, interrogating and displaying the data. If any of these system features are limited, the potential applications of the GIS are also limited.

In the 'real world' all geographic objects are three dimensional solids that can exist in isolation. Few GIS model objects as solids, however, because users rarely need this amount of information. For example, the boundary between two industrial plots may be a brick wall, which is itself a three dimensional solid. However, a GIS user is commonly only interested in the wall to the extent that it behaves as a boundary. A GIS would therefore *symbolise* the wall as a line, representing only its *property* as a boundary. Typically in GIS polygons represent the shape of an object, lines represent object boundaries or centre lines, and points indicate individual locations.

Individual characteristics, or *properties*, in a region are often the focus for study. For example, an engineer may be interested in the variation in soil shear strength over a region where a dam must be sited. In these cases geographic objects themselves need not be modelled. The variation in a characteristic can instead be classified into discrete ranges, then represented graphically by polygons enclosing values in each range. An alternative is to use pre-defined polygons and aggregate characteristics that fall within them, an example would be the study of the variation in the disabled population between Health Authority Districts. Statistics resulting from such studies can be more easily compared using fixed polygon boundaries (Lane, 1990). In either case the GIS symbolism of points, lines and polygons can effectively deal with the data.

The spatial relationship between one geographic feature and another is known as the *topology*. For example, if a line representing a fence boundary is moved the polygons on either side will change in area, perimeter and possibly non-spatial properties. Topology allows this dependency to be made explicit, forming a fabric of interconnecting access pathways which can maintain the integrity of the database. Topology also allows rapid retrieval of data about related items. Biljon (1987) identifies three types of topological relationship, *incidence*, *containment* and *exclusion*. Incidence refers to the adjacency or connectivity of features, containment is when one polygon wholly encloses another feature, and exclusion is when a polygon wholly excludes another feature. A feature in this case may be another polygon, a line or a point.

Topology itself is independent of spatial location, but merely depicts connective paths between features. The London Underground map is a typical topologic chart. The stations are not necessarily shown in the correct spatial location and neither are the rail links, but the connectivity between stations is explicit and accurate.

The storage of topology in GIS requires more complex facilities than those for the storage of objects alone. Various degrees of representation have thus evolved to satisfy the needs of different applications, avoiding any unnecessary data or functional overhead. These are detailed in Section 2.3.1.

Geographic objects are inherently complex, often made up of a collection of other more primitive objects. A building site, for example, is a collection of roads, open ground and building plots. Making enquiries about the building site may necessitate accessing the component objects. A GIS may allow for the creation and manipulation of complex objects of this type.

GIS vary in their abilities to meet the modelling criteria stated above. Some are more simplistic, storing polygons, lines, points and non-spatial properties, but not including topology or complex object definition. Others strive to supply all facilities, at the expense of complexity, a high storage overhead and increased development costs. The result is a range of products which, if carefully selected, can offer effective solutions for a wide range of applications.

2.3 Spatial Data Structures

The spatial data model of a GIS can be implemented in a number of ways. Three common strains exist, each differing in the way spatial information is represented as numeric data. These categories are known as *vector, raster* and *hierarchical* data models. Within each category alternative *data structures* are documented. Data structures refer to the more practical aspects of how numeric data is held, and how it is translated to and from the spatial view.

2.3.1 Vector Data Structures

Vector data structures represent feature polygons by explicitly storing their boundaries. Lines and points are held as precise geometric representations, giving advantages in accuracy and minimal data storage. Maguire & Raper (1990) classify vector structures as either topological or unstructured.

The unstructured vector structure stores each geographic feature as a string of cartesian coordinates. No attempt is made to model topological relationships, and this lack of structure has lead to the popular yet unflattering pseudonym 'spaghetti structure' being used to describe the representation.

The topological structure has two sub-classes, directional and complex. Between them they are, according to Maguire & Raper, the most widely used of all GIS data structures. The topological directional structure stores not only the end nodes of each line, but also the direction of input, creating a *directed edge*. This can yield the relationship between lines and adjacent polygons. The topological complex structure goes one stage further by explicitly storing polygon identifiers to the left and right of the directed edges. By using a topological structure much of the searching and analysis difficulties associated with spaghetti structures are avoided.

A typical overall vector data model is exhibited by the Kork GIS, (Keating et al, 1987). In this the topologic level is considered the central component of a three tiered data structure. The cartographic tier is that closest to the user, where the definition of polygons is by bounding lines and points. The topologic level is automatically derived from this, consisting of the nodes, edges and faces of the topological complex structure. The physical level concerns itself with optimising the position of data on the storage medium so that rapid access for common queries is achieved. Parallels with the three tiers of the GIS data model in Section 2.2 can be seen.

This is attained by clustering data from geographically close features into close proximity on the physical storage. Paging data from the medium can reduce the number of accesses needed to respond to common query types. This level requires the overriding, or at least the careful control, of any integral DBMS. The DBMS would normally have exclusive control of physical data positioning.

Reasons for the current popularity of the vector data structure are numerous. Display devices and hard copy methods both favour the use of vector data, and storage is compact and exact. However, vector systems suffer from a number of drawbacks, as Table 2.1 (page 13) illustrates (Burrough, 1989).

2.3.2 Raster Data Structures

Raster data structures do not model the boundaries of feature polygons, but instead use a regular grid of cells to represent areas directly. A polygon can be defined by stating which cells fall within it and which do not. At the simplest level a matrix of zeros and ones can declare the existence of a feature at any location. The choice of a zero or one is known as a binary digit, or *bit*, and is the lowest denomination in computer information storage. Declaring existence requires one bit per raster cell. Figure 2.1 highlights the difference between vector and raster representations.

By increasing the number of bits per cell more data can be stored. A single byte (8 bits) per cell can hold up to 255 alternative integer values. This is ideal for representing thematic data where in this case up to 255 categories can be supported. Location based queries, for example "What is at this point?", yield rapid responses. The system merely accesses the cell containing the point location. This is a marked contrast from vector structures where the enclosing polygon must first be found by geometric calculation using the boundary definitions. With the spaghetti structure in particular this can involve lengthy database searches.

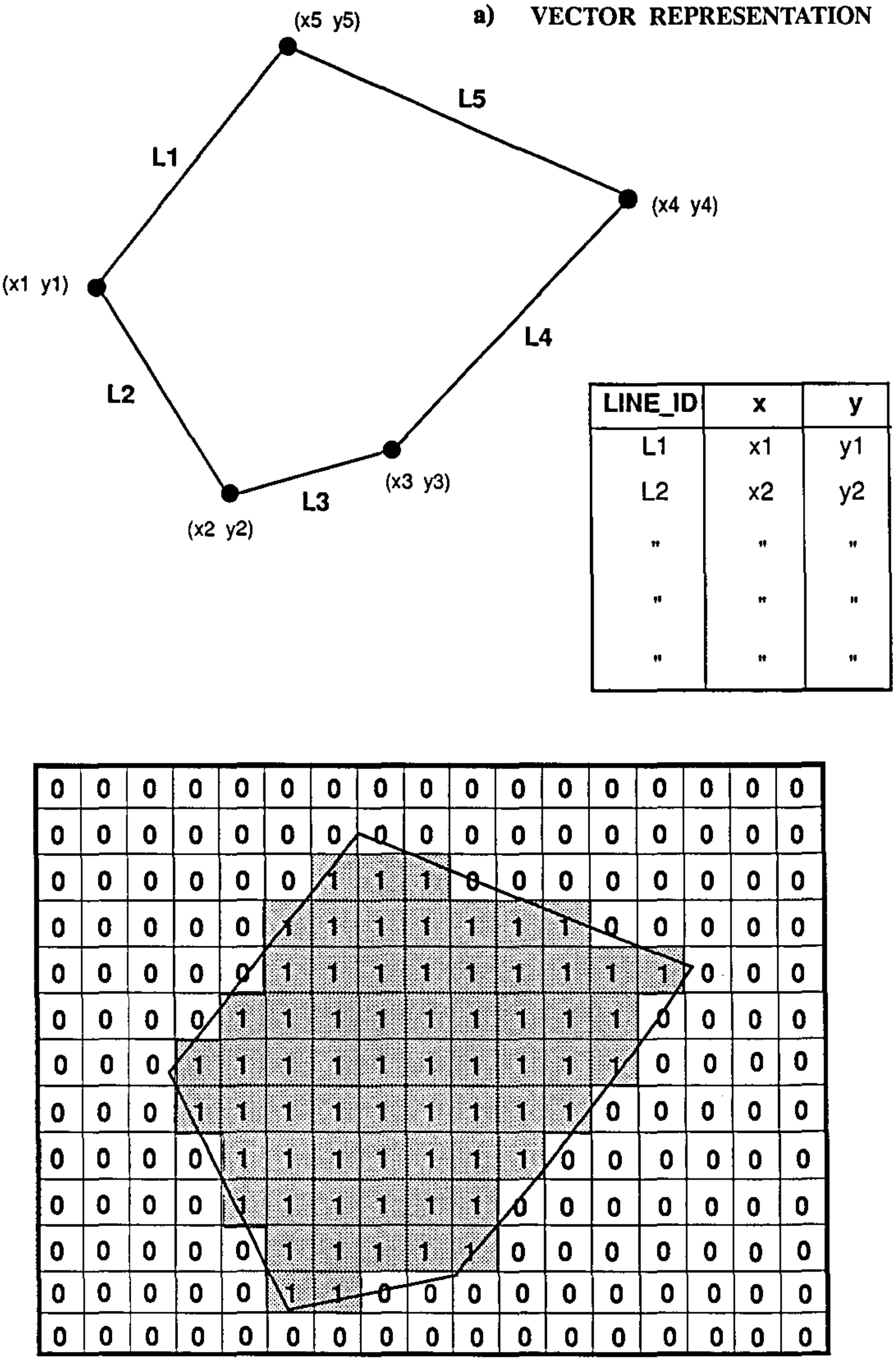
Advantages

- Good representation of phenomenological data structure
- Compact data structure
- Topology can be completely described with network linkages
- Accurate graphics
- Retrieval, updating and generalisation of graphics and attributes are possible.

Disadvantages

- Complex data structures
- Combination of several vector polygon maps or polygon and raster maps through overlay creates difficulties
- Simulation is difficult because each unit has a different topological form
- Display and plotting can be expensive, particularly for high quality colour and cross hatching
- The technology is expensive, particularly for the more sophisticated software and hardware
- Spatial analysis and filtering within polygons are impossible

TABLE 2.1 ATTRIBUTES OF VECTOR DATA STRUCTURES (FROM BURROUGH, 1989)



a) VECTOR REPRESENTATION

LINE_ID	x	y
L1	x1	y1
L2	x2	y2
"	"	"
"	"	"
"	"	"

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0
0	0	0	0	0	1	1	1	1	1	1	1	0	0	0	0	0
0	0	0	0	0	1	1	1	1	1	1	1	1	1	0	0	0
0	0	0	0	1	1	1	1	1	1	1	1	1	0	0	0	0
0	0	0	1	1	1	1	1	1	1	1	1	1	0	0	0	0
0	0	0	1	1	1	1	1	1	1	1	1	0	0	0	0	0
0	0	0	0	1	1	1	1	1	1	1	0	0	0	0	0	0
0	0	0	0	1	1	1	1	1	1	0	0	0	0	0	0	0
0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0
0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b) RASTER REPRESENTATION

Figure 2.1 A COMPARISON OF VECTOR AND RASTER REPRESENTATIONS

cd\5786P

Point and line data are vector by nature. When they are stored in a raster grid their spatial representation becomes an unconnected series of cells, and the notion of an explicit line element is lost. Thus any application wishing to use the lines explicitly, such as for network analysis, cannot do so. Hybrid systems, such as the vaster structure suggested by Peuquet (1983), have been suggested to overcome these difficulties. Dual representations are an alternative approach, with point and line features stored as vectors and area features in the raster form. The processing requirements do increase, however, as operations such as scaling and rotation need algorithms for both the raster and vector representations.

The size of each cell, or its *resolution*, governs the precision of the polygon depiction. Coarser resolution leads to more acute aliasing at polygon boundaries, causing a 'saw tooth' profile to be exhibited. In applications where a high precision is required the resolution must be chosen with care. Clearly the more cells used to represent a region, the higher the storage overhead, and the higher the precision.

Rather than represent each cell explicitly, a number of systems incorporate compression techniques for their databases. Mark (1986) describes run-length encoding, a method where successive cells with the same value are stored as a single record. Facsimile compression techniques have also been employed (Mason, 1989). Researchers in fractal geometry have obtained compression ratios in excess of 10,000:1, though at the expense of extremely lengthy processing times (Barnsley & Sloan, 1988). Although compression is possible the data storage overhead still appears to be a major concern among potential raster GIS users. The main advantages and disadvantages of raster systems have been tabulated by Burrough (1989), and this is given in Table 2.2.

Raster based GIS is becoming more popular. Mass storage costs are plummeting and new hard copy devices are often raster based. The range of skills available to software developers is also becoming wider, partly due to the increase in popularity of other raster based technologies such as facsimile communication and image processing. The choice of data structure for GIS applications is therefore not as clear as it once might have been.

Advantages

- simple data structures
- The overlay and combination of mapped data with remotely sensed data is easy
- Various kinds of spatial analysis are easy
- Simulation is easy because each spatial unit has the same size and shape
- The technology is cheap and is being energetically developed

Disadvantages

- Volumes of graphic data
- The use of large cells to reduce data volumes means that phenomenologically recognisable structures can be lost and there can be serious loss of information
- Crude raster maps are considerably less beautiful than vector maps drawn with fine lines
- Network linkages [ie: linked line and point features] are difficult to establish
- Projection [and] transformation are time consuming unless special algorithms or hardware are used.

TABLE 2.2 ATTRIBUTES OF RASTER DATA STRUCTURES (FROM BURROUGH, 1989)

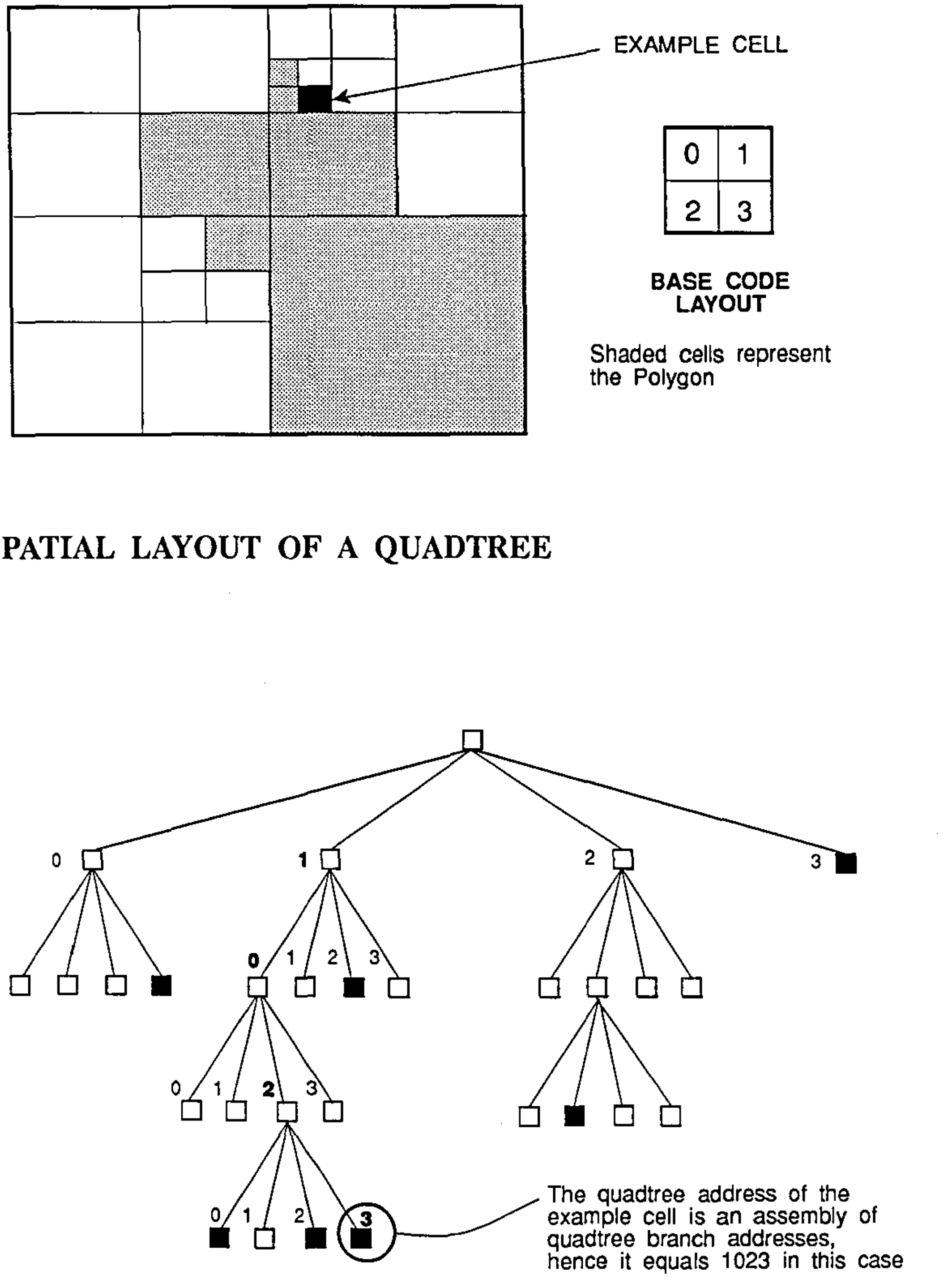
2.3.3 Hierarchical Data Structures

Hierarchical data structures may be considered a sub-set of raster structures in that they use cells to represent component regions of feature polygons. However, whereas most raster structures use a fixed cell size, the hierarchical cell resolution is variable. The approach recursively subdivides a region into smaller cells until either a cell becomes homogeneous (wholly 'black' or 'white'), or the minimum resolution is reached. This variable granularity is designed to minimise storage requirements.

Figure 2.2(a) shows a quadtree decomposition which typifies the hierarchical nature of these structures. The quadtree is based on a square grid cell, other arrangements include triangular and hexagonal tesselations (Holroyd, 1986). Each cell is recursively subdivided into 4 child cells, such that a tree structure is produced. The lowest cells, or leaf nodes in the tree, are homogeneous and combine to form the final quadtree representation of the region. The position of each cell in the tree indicates not only its location in the geographic region, but also it's cell size (Samet et al,1984a Hogg,1988).

According to Samet et al (1984b) quadtrees were traditionally implemented using explicit pointers to connect parent to child. A pointer is an item of data which is the address, or location, of another item of data. By following pointers a system can track from one data item to another. Later the disadvantage of the extra storage needed to store the pointers prompted interest in a pointer-less representation termed the linear quadtree. By labelling each quadrant of a decomposition with a digit (say 0 to 3) the location of any cell in the tree can be represented as a single reference number. Figure 2.2(b) shows a typical quadtree leaf and the derivation of the quadtree address. By representing the cell position in the tree the address holds the location and size of the cell within the geographic region. The number of cells required to model the polygon in Figure 2.2(a) to the illustrated resolution is 7 quadtree cells compared with 103 standard raster cells. The potential saving of storage space can clearly be seen.

Methods for quickly deriving the quadtree address of any cartesian coordinate pair were first defined by Morton (1966), although at that time the term 'quadtree' had not been coined. By taking the binary representation of each coordinate and interleaving the individual bits a new binary number is produced. Converting this to base 4 (the quadtree arithmetic base) produces an address which is that of the cell enclosing the original coordinates.



Shaded cells represent the Polygon

The quadtree address of the example cell is an assembly of quadtree branch addresses, hence it equals 1023 in this case

a) SPATIAL LAYOUT OF A QUADTREE

b) TREE STRUCTURE REPRESENTING THE SHADED POLYGON

Figure 2.2 THE QUADTREE HIERARCHICAL DATA STRUCTURE

If the ordering of the addresses is mapped directly into physical storage it is likely that adjacent cells will be represented in a similar physical storage location. However, at high level boundaries of a region, say either side of the region centre line, the spatial clustering does not hold. Despite this limitation fast spatial access to information is a commonly stated advantage of linear quadtrees.

The address order forms a curve which can be shown spatially, Figure 2.3. The spatial clustering of address allocation can be seen from the curve. Mark & Goodchild (1986) call this the Morton Order curve, and discuss its properties in relation to other space filling curves. It can be seen that the curve is self similar, its basic pattern being repeated over all levels of the cell hierarchy. This property is useful in that it makes it simple to process the data at different scales. The recursive nature of the curve also lends itself to manipulation using recursive languages such as the artificial intelligence language *Lisp*. The use of Lisp as the control language in Samet's early work is indicative of this (Samet et al, 1984b).

The properties of the quadtree address have become an intensive area for study. In particular the effects of normal arithmetic operations on the addresses has attracted a research following and the term *tesseral arithmetic* (Diaz & Bell, 1986). Applying multiplication to a quadtree address, for example, will lead to a change of scale and a rotation of the cell represented by that address. Denham et al (1986) note that "as tesseral multiplication takes the same time as conventional multiplication [but cartesian scaling/rotation takes longer], that scaling and rotation operations could be quicker using tesseral arithmetic and quadtrees".

Quadtrees model two dimensional regions, but the hierarchical structure can be extended to three or more dimensions. Gargantini (1982) and Kavoras (1985) both describe the octree, where each cell is a cube, with recursive subdivision into eight smaller cubes. Temporal modelling could be facilitated by an extension of this theme into higher dimensions.

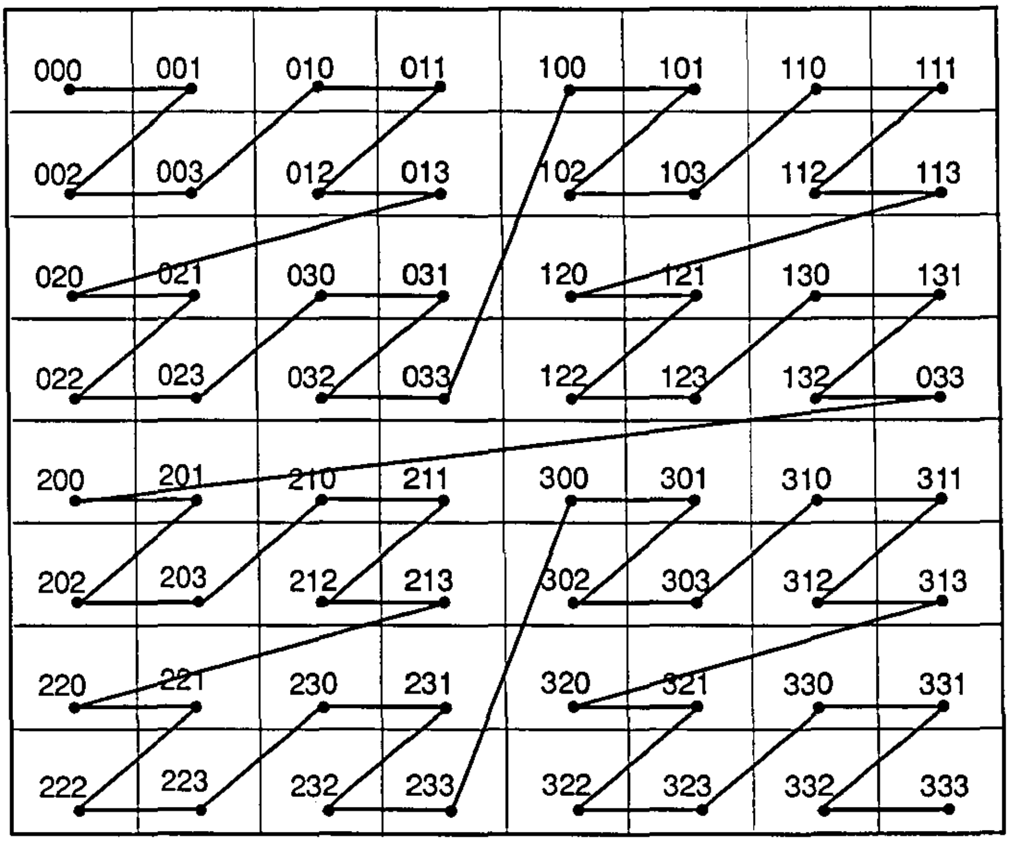


Figure 2.3 THE MORTON ORDER CURVE

Although hierarchical data structures appear to offer many advantages, they also have their failings. With particular reference to quadtrees Waugh (1986) notes that they lack topology, can be difficult to modify, take a long time to generate or update and only offer storage advantages when data is homogeneously clustered. A chequerboard raster image will gain no storage advantage from being quadtree encoded. Denham et al (1986) noted that "relative addressing takes equal time for both tesseral and cartesian systems". As many GIS functions need to use relative addressing, they conclude from this that "a very large class of [GIS] algorithms have nothing to gain from tesseral addressing". The advantages gained using quadtrees for some spatial manipulations must be offset against the high generation cost and poor performance in other areas before a decision is made to employ them.

Waugh also noted that, in common with other raster forms, line data is represented as a series of unconnected pixels. The line as a single spatial entity is lost. Samet et al (1984b,1987) and Callen et al (1986) have reported on techniques for explicit line storage within the quadtree, but all seem unnecessarily complex and difficult to implement when compared to using the vector structure.

2.4 DATA BASE MANAGEMENT SYSTEMS

Geographic information systems are often implemented using a proprietary database management system (DBMS). GIS system designers can then leave physical data storage, access, update, and manipulation tasks to the DBMS.

Three data models are in common use for DBMS, these being the hierarchical, network and relational models. A fourth data model, the object oriented representation, is also becoming popular.

The hierarchical model allows items of data, or *entities*, to be linked. In many situations this is useful, for example when one wishes to store the details about different types of car. In this case the entities which describe a car engine would ideally be linked to the car type that has the engine fitted. The hierarchical model allows links between a *parent* entity and one or more *child* entities. Thus a car type could be a parent, and the individual parts that make up the car type would be children. The hierarchical model allows a whole 'family tree' to be built, with children of one entity being parents to other entities. The overall specification of the logical structure of the database, without the actual data in place, is known as the *database schema.*

In the hierarchical model an entity is allowed multiple child entities, but each is allowed only one parent. In situations where a number of entities need to be linked to the same child, the child data must be repeated for each parent that needs it. For example, if two or more different car types use the same engine unit, all data describing the engine unit would have to be repeated for each type of car. Similarly if 300 bolts of a particular type were used in each of four different engines, 1200 repetitions of the bolt description would be required. Aside from the obvious storage overhead, data integrity can also be a significant problem. For example, if the specification of the bolt changed then all copies of its description would have to be found and updated.

Network, or plex, data models improve on the hierarchical model by allowing multiple parent and child connections. This avoids much of the data repetition problem. The network model uses a complex system of pointers to define data linkages. The US Department of Defense Conference for Data Systems Languages (CODASYL) have issued recommended guide-lines for the implementation of DBMS using this model (Olle, 1978).

The relational data model was first defined by Codd (1970,1971). Rather than use collections of entities connected by pointers, as in the network or hierarchical models, the relational model uses a collection of flat files, called *tables*. A relational table is analogous to a table on paper, which one might place in a report. It has one or more fixed length *fields*, corresponding to the columns in a normal table. Each row is called a *record*. Codd defined more abstract names which serve little purpose and so are not used in this thesis.

Relationships between items in different tables are formed when two tables have a common field. Those items which have the same value in that common field are said to be related. For example, if a car type table had a field for engine capacity, and a engine table also had a field for engine capacity then the potential exists for relationships to be made. If, for example, a Ford Sierra had an entry of 1600cc in the engine capacity field, and one of the engines in the other table also had an engine capacity entry of 1600cc, then the two data items are related. The JOIN operation will produce a new table which concatenates all related records in the two tables, so that a full list of related items is produced.

There are instances when more than one item in one table will relate to a single item in another table. This is known as a *one to many* relationship, for example if both a Ford Sierra and a Ford Granada had a 1600cc engine type entry. The JOIN function would create a record for each car, repeating the engine information for each car. The key point about the relational philosophy is that data items need never be repeated in the database, as the JOIN command can be used to relate, extract and repeat data where necessary.

Other essential operations allowed by the relational DBMS are SELECT and PROJECT. SELECT returns records in a table that match criteria laid down by the user. For example one could SELECT all cars that have an engine capacity over 1600cc, and the DBMS would return a new table containing only those records that matched. PROJECT returns only selected fields, making a new table with less 'columns'.

The philosophy behind the object oriented data model is that information about any object cannot consist of data alone, but must also include the actions that the object can perform (Sernadas et al, 1987). The object oriented data model honours this by encapsulating data about the object inside a shell of software procedures which model object behaviour. Access to the data is only possible by using the procedures. Hence the object oriented data model is as much a programming tool as a data storage tool. It has been used extensively in artificial intelligence research, and the reader is referred to Section 3.3.3 for more detail.

As was mentioned in Section 2.2, GIS data falls neatly into spatial and non-spatial categories. The hierarchical DBMS has found favour in neither category due to data repetition. Network DBMS by contrast have proved adept at the storage and manipulation of spatial data. Early relational DBMS could not compete in speed with the network model for spatial applications, but they proved much simpler for non-spatial information. A common approach, exemplified by the commercial GIS ARC/INFO, has been to combine a network spatial model with a relational non-spatial model (Moorehouse, 1990).

The increasing speed and functionality of relational DBMS in recent years, combined with the failings of the hybrid network/relational model, lead to the development of a number of purely relational GIS. The CIS Medusa GIS was one of the first and most ambitious ventures, but was plagued by poor performance and was never commercially released (Bundock,1987 Durrant,1988).

GEOVIEW, on the other hand, was an academic research prototype designed to test whether a relational DBMS *could* provide the necessary flexibility for a GIS. Waugh & Healey (1987) considered the fundamental difficulty as being the mapping of variable length cartographic data into the confines of a fixed field length relational table. Schema were designed for vector, grid and restricted quadtree data formats, with the emphasis placed on minimising the number of tables required for each (Waugh & Healey,1986: Sinha & Waugh,1988). They concluded that a fully relational GIS was practical by using variable field lengths afforded by bulk storage data types, such as those provided by the Oracle RDBMS (Rolland, 1990). Such data types are not a standard part of the relational model, but are useful extensions to it.

A number of researchers have indicated the need for extensions to the relational model in order to facilitate spatial data storage and manipulation (Frank,1988 Guptil,1987 Newell & Theriault,1990). In particular the Structured Query Language (SQL), the 'industry standard' relational language, (ISO9075, 1989) has been cited for extensions to suit spatial data storage (Gradwell,1990). The suggested improvements centre on the need for spatial data types and operators, data types of indefinite length, and user defined data query procedures. With these extensions it is hoped that the next version of SQL will be a practical relational environment for both the spatial and non-spatial parts of a GIS.

Examples of object oriented GIS include PROBE and TIGRIS. PROBE is a research tool which has been applied to network routing (Dayal & Buchmann,1987) whereas TIGRIS is a fully operational commercial GIS, supplied by Intergraph (Herring,1987,1990). Both claim a level of automatic data integrity which was more difficult to achieve using the standard relational model.

2.5 Digital Terrain Models

Digital terrain modelling (DTM) systems were originally developed for highway design and civil engineering applications, but have since been incorporated into GIS to handle elevation data. A DTM is essentially a two dimensional surface contorted in three dimensions to represent the topography of a geographic region. Without DTM the uses of GIS would be restricted, particularly in the areas of land resources assessment and planning. The HORIZON GIS, for example, is designed specifically for environmental planning and relies heavily on DTM for visualisation and analysis (LaserScan,1990).

There are a number of ways in which the DTM surface can be formed and held in the computer database. Two of the most popular are the triangular irregular network (TIN) model, and the grid based model.

The TIN model is designed to be used with elevation data collected at random, rather than at regularly spaced point locations. The modeller operates by joining all known points together with lines to form a lattice of triangles. With vertices at different elevations, these triangles form flat but sloping faces which collectively form the model. The elevation at any point can be found by using linear interpolation across the appropriate triangle face. The TIN model may be considered analogous to the vector data structure for spatial modelling in GIS.

The grid based model consists of a matrix of elevation values, with a value's location in the matrix corresponding to the location in space it refers to. The elevation values can be found by direct sampling from the region in question, or the regularly spaced values can be found by interpolation from more randomly sampled values. The grid based model may be considered analogous to the raster data structure for spatial modelling in GIS.

Other modelling methods are available, for example string models. More general mechanisms for holding elevation data include those that rely on storing point data that was originally sampled. By using a distance weighted averaging function the elevation at any location may be found. Such mechanisms are not common in current GIS, however.

The advantages of the TIN and grid based DTM structures are also analogous to those of the vector and raster GIS structures. TIN models excel where there is sparse, randomly located elevation data. In such cases the storage is compact, and the elevation values are exactly represented at the known points. TIN models suffer on speed of access and speed of creation when the number of points becomes high, for example when there are several thousand points for microcomputer based implementations. Grid models, by contrast, must store the elevation at every grid point, thus incurring high data storage overheads. However, access to the resulting data is much faster, and does not degrade appreciably with an increase in the number of original data points.

The author has performed extensive research aimed at assessing terrain modelling systems for applications such as GIS, highway design, land surveying and cartography (Finniear, 1986a, 1986b, 1986c, 1987b, 1987c, 1987d, 1988a: Mayo et al, 1986: Dickens

& Finniear, 1987). Durrant (1988) coined the term *intelligent ground model* when tailoring a TIN based DTM to predictive flow modelling of sediment deposits in tailings dams. The author has also explored the possibility of giving a DTM intelligence during preliminary work for this thesis (Finniear et al, 1988).

2.6 Data Capture

Data for GIS divides neatly into two forms, spatial and non-spatial. Capturing the spatial component poses particular difficulties as the technology is still developing to process it. Non-spatial data capture can also be expensive if it exists in large quantities, as the keyboard is often the only reliable method of input.

Getting information into a GIS is an important consideration, not least because of the costs involved. Data may be captured using a number of techniques, detailed later. With spatial data capture raster and vector representations act as a division, with data capture methods favouring each category depending on the format of the information they produce. Quite naturally those techniques producing vector data are more suited to GIS using vector data structures, and raster data to raster structures.

This categorisation does not preclude raster data from use as input to vector systems, or *vice versa*. This may be desirable to take advantage of the strengths of the alternate data structure. However, to do so requires data conversion. Vector to raster conversion is the most straightforward as topology is lost. Each raster cell must attain a value depending on its position within the geographic region, and this must be found by calculation from the vectors. The process is often called *raster scanning* due to the cell by cell sampling involved.

Conversion from raster to vector is more complex as topology is effectively added. Individual pixels must be grouped as lines or polygons, and this requires recognition software or intensive manual interpretation. The field of feature recognition and classification has become an intensive research area as illustrated in Section 4.4.2.

The common denominator for all GIS is the base coverage map. It is to this backdrop that all other features are referred, and without it the operator would be unable to interpret any result. Sources of digital base maps do exist for some areas of the UK.

The Ordnance Survey and utility companies are co-operating to achieve national coverage by the mid-1990s. If a base coverage is not available in digital form there are a number of alternative actions that can be taken.

The first, and perhaps the most obvious, method is to manually digitise a paper copy of the base map. This produces a fully structured coverage in vector form, which can be manipulated at will. The process is expensive and tedious, however, and operators are prone to making errors on complex coverages.

Petrie (1987) notes that automated line following systems, such as LaserScan FASTRAK, can perform digitising at greater speed, although still requiring operator intervention at times. He also reviews the hardware for auto-digitising, and discusses raster scanning. This alternative produces base coverage maps in minutes rather than hours or days. However, the scanned image is raster based. The file has no topology, lines are considered to be unconnected pixels, and no features exist for non-spatial attributes to be connected to. If the coverage is purely for viewing or reference then this is both adequate and cost effective.

Direct capture of data from the 'real world' is also a viable option. Remote sensing, either by satellite or aerial photography, is in common use for land assessment (Kennie & Matthews, 1985:Bullard & Dixon-Gough, 1985). Satellites, by applying filters to pick out different radiation frequencies, can achieve images to 10 metre resolution (SPOT, 1988). Certain spectral bands can differentiate between vegetation types, whilst infra-red bands have proven valuable in such diverse operations as detecting oil spillages and thermal discharge monitoring (Mason & Amos, 1985). Active sensors, bombarding the Earth's surface with microwaves or radar and measuring reflected signals, can discern surface roughness and is thus particularly useful in ocean measurement (Gudmandsen, 1983). The output from these sensors is raster, with individual cells holding a single sensed wavelength in each band. Integrating remotely sensed data into GIS is currently an area of intense research (Robinson Barker, 1988; Zhou, 1989).

Aerial photography is most valuable when stereo pairs of photographs are used in the derivation of height information. Photogrammetry now offers acceptable accuracy for many uses at lower cost than land surveying. Stereoplotters can also produce vector data rather than raster, allowing increased image structure should the application merit it (Price & Webb, 1987).

Land surveys offer unsurpassed accuracy but tend to be prohibitively expensive for general GIS data collection (Finniear, 1987a). Some situations, such as the detection and mapping of underground utilities, necessitate this approach nonetheless (Finniear, 1989).

2.7 Implementing a GIS

The implementation of successful GIS is far more expensive and prone to failure than computer aided draughting (CAD) systems of similar size, for example. The primary difference is that GIS require extensive data input before they can be used. CAD system managers may have large paper drawing archives eventually requiring entry, but this does not preclude their immediate use on new projects.

Techniques of data capture are thus crucial to any implementation. The quantity and quality of the data needed in the GIS depend largely on the application. Utility companies, for example, may not need a full topologic structure in their base map if it only gives a backdrop to their network of pipes or cables. Conversely the Land Registry, with a need to store boundary detail for 12.5 million properties, must have structured topologic input from 700,000 property maps before its GIS becomes fully operational (Partridge,1990). Cost/benefit analyses for GIS implementations at this scale can be difficult to perform and quantify (Todd, 1989).

Geographic information is inviolable. Different organisations working in the same region will inevitably be using, at least in part, duplicated data from that region. The operations of one group may have a direct effect on those of another. This is particularly true of the utility companies and local councils. The National Joint Utilities Group (NJUG) are experimenting with inter-company GIS to coordinate street works and eliminate PUSWA Works notification documents (DoE,1987; Purvis,1989). The city of Cork in Eire are already realising considerable benefits from an approach involving utilities, the City Council and emergency services contributing to one city wide GIS database (Pollitt, 1989).

GIS users may be individuals, departments within companies, or whole organisations in a multi-department context. Corporate GIS crosses traditional boundaries of discipline and job description, raising the possibility of inter-departmental conflict, job description and industrial relations issues (Finniear, 1989). Medyckyj-Scott

(1989) imparts an enlightening though worrying view of corporate GIS, with constructive ideas for a smoother introduction.

2.8 APPLICATION AREAS AND EXAMPLES

Application areas are becoming extremely diverse. Each one has different needs, and the implementors adopt systems and management strategies to suit. A categorisation of user types appears unachievable as innovative researchers in more incongruous fields continue to diversify the user base.

GIS users fall nonetheless into these broad groups according to why they store geographic information:-

- Storage for record and retrieval purposes
- Analysis of existing data to provide new maps or statistics
- Simulation or predictive work
- Routing of vehicles
- Planning for environmental change

The first group are typified by the utilities who need to maintain up to date records of their entire installed infrastructure. The maintenance schedules, age, position and modification data are imperative for efficient management in these companies.

Analysis of geographic features and the production of new maps is perhaps a more traditional role for GIS. A common technique is to overlay different feature types to produce new maps of particular significance. Kapetsky et al (1988) use GIS to get suitability maps for aquaculture, specifically to isolate geographic areas where catfish farming could be profitable.

Simulation and predictive modelling is raising many hopes and some promising results. Lessard et al (1988) demonstrate that the epidemiology of a disease called east coast fever can be modelled in a GIS through charting the spread of the vector tick *Rhipicephalus appendiculatus*. Pollution spread has also been predicted by this technique (Hession & Shanholtz, 1988).

The Chorley Committee report to the British Government noted that vehicle navigation inefficiencies cost Britain £2,400M per annum. The Autoglide system, proposed by the Department of Transport, plans to use a network of roadside beacons to transmit directions and road conditions to equipment installed in vehicles (DoE, 1987). The system will react in real time to accidents and hold-ups, re-routing vehicles around the problem.

New developments, roads, industrial estates or housing, need meticulous planning to avoid excessive impact on the existing area. By using GIS the region can be thoroughly examined to find the best site and layout for the development. The Department of Surveying, Newcastle University is investigating the potential of applying conventional GIS technology to civil engineering applications (Hosken, 1989).

2.9 SUMMARY

Geographic information systems have been acknowledged as the most significant step forward in the handling of geographic information since the invention of the map. They can provide a repository for storing vast amounts of geographic data, and have facilities for capturing, validating, maintaining, manipulating, analysing and displaying it.

Between the representation of the geographic information on physical storage and the understanding that the GIS user eventually derives from the data, three data models can be said to exist. These are the DBMS data model, the spatial data model and the user data model.

The DBMS data model controls the physical storage and manipulation of data. Network and relational DBMS are most commonly used for GIS implementations, though object oriented data model is now becoming more popular.

The spatial data model governs how the numeric data is translated as a spatial view. Vector, raster and hierarchical data structures can be used for the spatial data model. Vector, using lines and points to represent boundaries, has the most compact storage whilst raster, using a grid of cells, allows greater analysis, faster image overlay and more rapid location based access. Hierarchical data structures are raster based, yet attempt to reduce the data storage overhead by generalising adjacent cells containing the same values. Digital terrain models are extensively used to provide the third dimension in GIS.

The user data model is the translation from the lines, points and other data held in the GIS, into the user's understanding of the geographic features that actually exist in the 'real world' region. The extent to which the GIS software takes part in this translation varies between implementations. However, a key conclusion is that GIS software does not have the inherent ability to understand and reason with the data it holds, in the context of the problem the user is solving. In this respect GIS software falls short in its contribution to the user data model.

The practical application of GIS within organisations has also been discussed, and a wide range of potential uses have been outlined. A key issue is data capture and input. The cost implication is substantial though the options available are wide, from using satellite remote sensing to digitising existing paper maps. Automated techniques for speeding up data capture and interpretation are now becoming available. In large organisations a GIS has diverse potential benefits which span conventional departmental boundaries. This makes cost/benefit predictions difficult, and also means that corporate structure may need appraising prior to GIS installation. However, judging by the rapid increase in installed systems, the potential for the technology when used correctly appears almost limitless.

CHAPTER 3

3. ARTIFICIAL INTELLIGENCE AND EXPERT SYSTEMS

3.1 INTRODUCTION

Artificial intelligence (AI), 'the science of the thinking machine', has moved from obscurity to increasing popularity in the 1980's with the advent of readily available *expert system* tools. These tools have allowed people in disciplines other than computing to gain access to certain AI techniques, particularly concerned with the representation and use of knowledge. The idea of being able to capture the entire expertise of, for example, a human engineer, barrister or physician is a thought that has captivated individuals and organisations alike. Being able to use expertise as a commodity, to distribute, buy and sell it without losing the human expert would allow knowledge to be disseminated far more easily and cheaply than before.

In response to these perceived opportunities the UK Department of Industry (1982) launched the Alvey initiative, providing funding for collaborative research ventures in industry and academia.

In practice the modelling of human expertise is, as could be expected, not straightforward. A wide range of tools and techniques are available, many of which have been used in the work presented in later chapters. This chapter concentrates on providing the theoretical background that will be needed to fully understand the discussions pursued later in this report.

Much controversy has surrounded the use of the term *Artificial Intelligence* since it was first coined with reference to computers in the mid 1950's. Despite it's increasing maturity, the term still begs a concise yet satisfactory definition.

It is generally accepted that AI systems should exhibit properties that the human user would in some way regard as 'intelligent'. However, philosophers have yet to agree on what constitutes human intelligence. Edward A. Feigenbaum, one of the early pioneers in AI, takes a linguistic approach to his definition :

> "The word *intelligence* derives originally from the Latin *legere,* meaning to gather (especially fruit), to collect, to assemble, and hence to choose and form an impression. *Intellegere* means to choose among, hence to understand, perceive, and know. If we can imagine an artifact that can collect, assemble, choose among, understand, perceive, and know, then we have an artificial intelligence ..." (Feigenbaum & McCorduck, 1984).

Early researchers saw the aim of AI as the creation of a machine which could function like the average human in any given situation. Charniak and McDermott consider AI in this most general sense :

> "Artificial intelligence is the study of mental faculties using computational models. The ultimate goal is to build a person, complete with vision, language recognition, a method of deduction and search (thinking and reacting to stimuli), speech and movement." (Charniak & McDermott, 1985)

This thesis is primarily concerned with the creation of software systems that could be said to think.

Conventional computer programs are designed to solve particular problems by a mixture of calculation, storage and repetition. If this were all that humans did whilst thinking then all programs could be said to display intelligence. However, human decision making is rarely based on pure calculation. Judgements often result from a combination of reference to past experiences, reasoning between likely outcomes, and the use of *heuristics*, or 'rules of thumb', to guide the search for solutions. Note that in this and subsequent Chapters a substantial amount of AI terminology has had to be used. A brief glossary is provided at the end of the thesis should the reader need to refer to it.

Conventional programs are guaranteed to reach a solution only if all necessary information is available and steps for solving the problem have been specifically written into the code. When faced with partial information a human can still attempt to find an acceptable solution. An AI system can also be designed to explore possible solutions if only partial information is given. To do this it uses a store of knowledge of the type a human would use if faced with the same problem. By searching though this *knowledge base* it can propose likely answers and ask the user questions to help it narrow down the number of possible solutions. In doing this an AI system can suggest solutions in areas where a conventional program could not operate.

AI systems are based on the use of symbols rather than numbers. A symbol is a word representing a real world object or situation. Symbols can be related, for example the symbol DOG can be related to the symbol TAIL by the relationship HAS_A. Logical rules can be written into AI systems which can infer facts, for example :

IF the DOG has_a TAIL
THEN the DOG is not a ROTTWIELER

These symbols and rules make up the 'knowledge' the system has about the application. The power of AI systems is the ability to store and manage large collections of symbols and logical rules, an ability not possessed by conventional computer programs.

Although AI systems can depict complex relationships between facts, they tend to lack the capacity for storing large volumes of data without seriously degrading the operating speed. Many AI techniques rely on the ability to 'scan' the entire list of facts to isolate individual items, the longer the list, the longer each scan takes. Also it is common for AI systems to store all facts in the computer memory, for applications using high volumes of data this can become an impossibility. A number of systems now offer links to conventional databases to help by-pass this problem.

3.2 Expert Systems and Knowledge Based Systems

A subset of AI systems generally is the knowledge based system (KBS), and a subset of KBS is the expert system. Waterman (1986) illustrates the relationship in Figure 3.1.

Unlike general AI programs knowledge based systems contain a distinct *knowledge base*, holding knowledge about the application, or *domain*, separate from the search mechanisms and user interfaces. The search strategies are collectively known as the *inference engine*. This has the task of manipulating the knowledge base in the search for solutions.

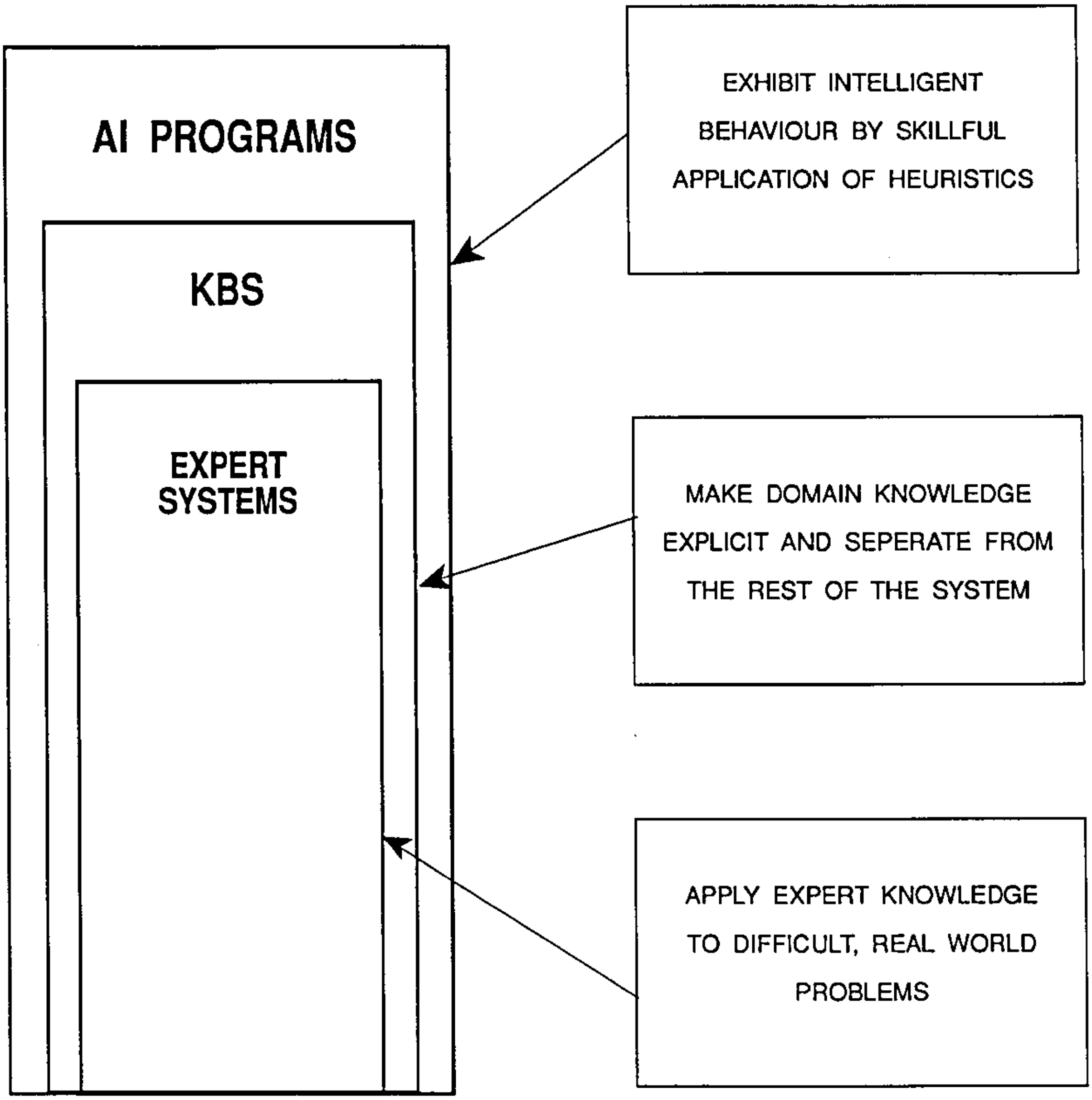

AI PROGRAMS
KBS
EXPERT SYSTEMS
EXHIBIT INTELLIGENT BEHAVIOUR BY SKILLFUL APPLICATION OF HEURISTICS
MAKE DOMAIN KNOWLEDGE EXPLICIT AND SEPERATE FROM THE REST OF THE SYSTEM
APPLY EXPERT KNOWLEDGE TO DIFFICULT, REAL WORLD PROBLEMS

Figure 3.1 AI, KNOWLEDGE BASED SYSTEMS AND EXPERT SYSTEMS IN CONTEXT (Waterman, 1986)

Allwood (1989a) further distinguishes expert systems by noting that the form of the knowledge representation must be comprehensible to both man and computer, and that the system must be able to give the user an explanation as to *how* the answer was found. Expert systems have to provide answers to non-computer experts, and justify those answers in a form that the user can easily understand.

Expert system programs can be purchased which have an empty knowledge base, but are complete in every other sense. These are known as *expert system shells*. By inserting the knowledge himself a user can rapidly create an expert system, avoiding all the extra work of writing the rest of the inference engine and interfaces.

The isolation of the knowledge base is such that different inference engines can be used with the same knowledge base, just as the same inference engine can be used with different knowledge bases. Thus an expert system can 'think' about the same problem in different ways by using different inference engines.

Expert system shells are widely available and are the subject of many reviews in computing literature (Naylor,1987a,b,c Williamson,1987).

3.3 KNOWLEDGE BASES

Knowledge bases store expert knowledge about an application, or *domain*. Acquiring this knowledge in the first place is a subject covered in Chapter 5. The knowledge must be stored in an easy to understand yet functional form. This section deals with knowledge base forms commonly used.

The formal basis for the methods described below is *first order predicate logic* (FOPL). FOPL allows both the representation of facts about classes of objects, and a way of writing rules which can use existing facts to infer new facts. Lisp and PROLOG are two AI languages based on the use of FOPL (Winston & Horn, 1984; Clocksin & Mellish, 1987). The formal definition and terms of FOPL are covered in detail by Pavelin (1988).

3.3.1 Production Rules

Coding expert knowledge into rules is one of most popular forms of knowledge representation. It seems natural for an expert to be able to list the criteria he uses to make decisions in this way. For example, the rule :

IF the client was born in the United Kingdom
AND at least one of his parents was settled in the
United Kingdom at that time
THEN the client is a British subject

is one rule in a legal KBS depicting the British Nationality Act (Council for Science and Society, 1989). The IF part of the rule is known as the *antecedent*, and the THEN part is known as the *consequent*. By producing a large number of these rules it may be possible for an expert to completely cover his decision making process. Legal and medical applications are typical of domains where the rules can be explicitly laid out. The MYCIN expert system for diagnosing bacterial infections was an early example of a purely rule based system which worked well (Buchanan & Shortcliffe, 1984).

Rule based systems use individual, unconnected symbols to represent facts, relying on the rules or the users intuition to make the connection between them. for example, the symbols CLIENT and PARENT_OF_CLIENT have no explicit relationship in the knowledge base, even though to the operator it is obvious that a relationship exists. This can become a distinct liability when a large number of facts are needed to achieve a solution, or where the interrelationships are more subtle.

Rule based expert system shells are simple to write and easy to understand, and are justifiably popular with developers. However, the lack of structure in domain facts and the restriction of having to encode all knowledge as rules can make them hard to use in complex domains. A criticism of early systems was that the rule format was difficult to use and understand. Unfortunately this criticism is still valid for some systems, Appendix E shows this as a failing of the GoldWorksII rule format.

3.3.2 Frames

In 1975, Marvin Minsky proposed a more structured representation for knowledge, which he called *frames* (Minsky, 1975). A frame represents a class of real world objects and contains a description of the class. For example, the animal type ELEPHANT could be represented using a frame. Discrete individual objects which are members of a class are called *instances* of the frame. For example the individual elephant CLYDE, would be represented as an instance of the frame ELEPHANT. The concept is best explained pictorially, Figure 3.2.

Each frame contains *slots*, which are the characteristic qualities that describe the frame. The ELEPHANT example in Figure 3.2 has slots for colour, number of eyes etc. The normal description of an elephant is given in the frame. When an instance is created, *in the absence of more specific information*, the default properties of the generic elephant are *inherited.* Note that in the figure CLYDE has lost a leg. The importance of inheritance is that only specific deviations from the frame description need be stored for the instance, the rest is inferred from the parent frame. Thus storage requirements can be reduced considerably.

Inheritance is also effective between frames. Frames can be arranged in a hierarchy, with child frames inheriting the slots of parent frames. Figure 3.3 is an example of frame inheritance. Note that in the figure the inherited slots are above the dashed line, whereas the slots peculiar to that frame are below the line. The DOLPHIN frame illustrates *multiple inheritance,* as it is both a MAMMAL and a WATER-BASED-CREATURE. Instances can be created for any frame in the hierarchy, but instances can only belong to one frame and they cannot have child instances or frames of their own.

Constraints may be put on the values that a slot can contain. These constraints are commonly known as *slot facets*. Typically only certain slot values might be allowed, for example GOOD, BAD and INDIFFERENT, or only a specific range of numeric values might be permissible. Slot facets are inherited in the same manner as slot values.

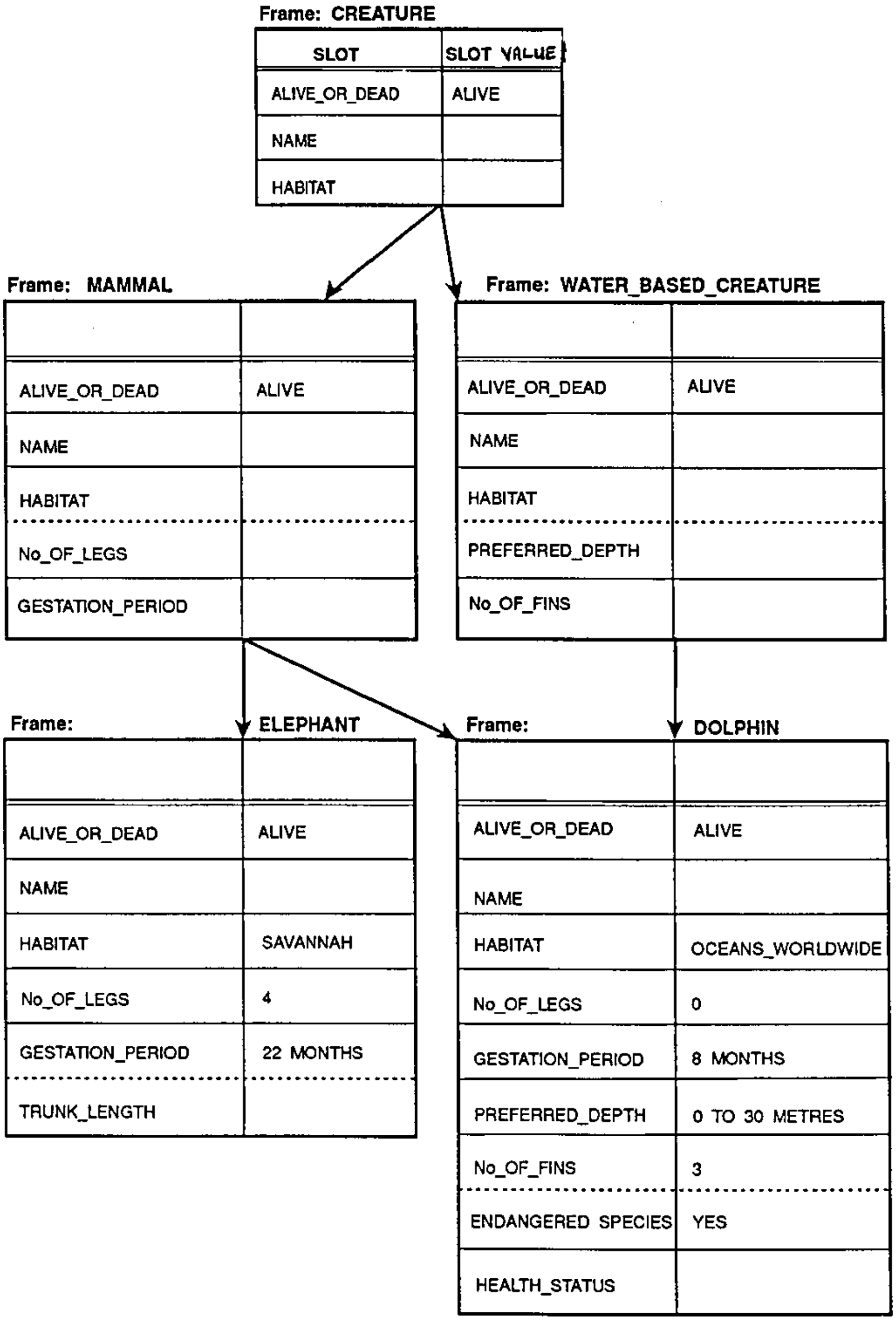
FRAME: ELEPHANT

SLOT	SLOT VALUE
COLOUR	GREY
EYES	2
LEGS	4
SKIN_TYPE	TOUGH
FUR	NO
ALIVE_OR_DEAD	

INSTANCE: CLYDE

SLOT	SLOT VALUE
COLOUR	GREY
EYES	2
LEGS	3
SKIN_TYPE	TOUGH
FUR	NO
ALIVE_OR_DEAD	ALIVE

Figure 3.2 AN EXAMPLE OF FRAMES AND INSTANCES



Frame: CREATURE

SLOT	SLOT VALUE
ALIVE_OR_DEAD	ALIVE
NAME	
HABITAT	

Frame: MAMMAL

ALIVE_OR_DEAD	ALIVE
NAME	
HABITAT	
No_OF_LEGS	
GESTATION_PERIOD	

Frame: WATER_BASED_CREATURE

ALIVE_OR_DEAD	ALIVE
NAME	
HABITAT	
PREFERRED_DEPTH	
No_OF_FINS	

Frame: ELEPHANT

ALIVE_OR_DEAD	ALIVE
NAME	
HABITAT	SAVANNAH
No_OF_LEGS	4
GESTATION_PERIOD	22 MONTHS
TRUNK_LENGTH	

Frame: DOLPHIN

ALIVE_OR_DEAD	ALIVE
NAME	
HABITAT	OCEANS_WORLDWIDE
No_OF_LEGS	0
GESTATION_PERIOD	8 MONTHS
PREFERRED_DEPTH	0 TO 30 METRES
No_OF_FINS	3
ENDANGERED SPECIES	YES
HEALTH_STATUS	

Figure 3.3 EXAMPLE OF FRAME INHERITANCE AND MULTIPLE INHERITANCE

Frames handle static knowledge about domain objects in a complete yet concise manner, and a number of expert system tools are now incorporating this methodology. Most implementations allow the use of frames with production rules. This increases the functionality of frames as they can then be used as the fact base for rules, adding a dynamic element to the previously static knowledge. Production rule systems benefit from using frames as the facts used for rule matching are no longer unstructured. Rules can now refer to a class of objects, checking each member of the class.

The example rule below is given with reference to dolphin frame in Figure 3.3. Note that this rule is in GoldWorksII format. Words preceded with a question mark are variables, and explanatory notes are given to the right of the rule:

```
                                          (EXPLANATORY NOTES)
   IF instance ?animal is DOLPHIN         ;For each dolphin
   WITH NO_OF_FINS ?number                ;find number of fins
   (<> ?number 3)                         ;if this number <> 3
THEN                                      ;then
   instance ?animal is DOLPHIN            ;that dolphin
   WITH HEALTH_STATUS DISABLE             ;is disabled.
```

This rule is working within the frame hierarchy to check every instance of DOLPHIN, no matter how many there are. Also, if the DOLPHIN frame had child frames then all the instances of the child frames would also be checked as they have DOLPHIN as an *ancestor*. This makes the rules extremely powerful and flexible. However, the penalty for this flexibility is that unless the system designer has a complete grasp of the scope of the rule the system may apply it to more of the knowledge base than was intended.

Daemons are another powerful facility available to many frame based implementations. In the simplest sense daemons are procedures which can be attached to one or more slots of a frame, to monitor what happens to the slot values. There are two common types, *when-modified* and *when-accessed.* When-modified daemons sense when the slot value has been changed, and this triggers the execution of the daemon procedure. When-accessed daemons are triggered whenever the slot value is used, irrespective of whether it has been changed or not.

The presence of daemons elevates the frame description from being a passive collection of facts to being an active collection of 'objects which respond when touched'. For example, a daemon attached to the NO_OF_FINS slot of DOLPHIN could automatically sense if the number is changed, and could change the HEALTH_STATUS of the any dolphin instance accordingly. Parallels to the production rule example above can be seen, but wheres rules have to wait in a queue to fire, daemons react immediately and can be far more detailed than a rule consequent. Daemons are inherited through the hierarchy in the same way as slots. The use of daemons is also known as *access oriented programming*.

3.3.3 Objects

Object oriented methodologies attempt to closely depict a real world object not only by describing its characteristic facts, but by explicitly defining the functions it can perform (Pascoe,1986; Sernadas et al, 1987). The functions encapsulate the object description so that raw facts about the object cannot be accessed directly, instead the encapsulating functions have to be used.

Consider the frame based and object oriented descriptions of a lamp, as seen by the program or user :

FRAME BASED LAMP

LAMP

Current State	: ON/OFF
Cost (£)	: 20.95
Power Requirement (w)	: 60

OBJECT BASED LAMP

LAMP

SHOW CURRENT STATE
SWITCH ON
SWITCH OFF
SHOW COST
SHOW POWER REQUIREMENT

The functions of an object are called *methods*. A method is activated when it receives a *message*. To find out the cost of the lamp above, for example, the user must send a message to the method SHOW COST. Objects can only respond to messages they can understand. The lamp example has no methods for changing the cost or power requirement, therefore the user would never be able to change these facts, although he can access their values.

Methods may perform other tasks in addition to changing raw facts about the object. They can, for example, send messages to other objects that might be affected by a change to this object. If the example lamp is sent a message to SWITCH ON, the method may also send a message to a power supply object to say that the lamp is drawing current. The appropriate power supply method would then activate and may send further messages.

The propagation of messages through a network of objects is a key feature of the technique, as it means that the integrity of the representation cannot be violated. All effects of a single change are identified through the chains of messages and methods firing as a result. An analogy can be drawn with a civil engineering structure in equilibrium. A new external force results in a chain reaction through the structure until equilibrium is regained. The object structure must also regain equilibrium.

A hierarchy of objects and object classes can be built in a similar manner to the frame based representation. Methods and values may be inherited in the same way as slots and daemons.

The object oriented methodology has been successfully applied as a conventional programming technique. Most notable of these are the object oriented graphic interfaces of the Apple Macintosh (Simpson, 1986) and Microsoft Windows (Ewing,1989 Sheldon,1990). A number of programming language compilers have also adopted object oriented extensions, for example the C++ superset of the C programming language (Stroustrup,1986 Zortech,1988). The object oriented technique has been seen as a potentially powerful representation for knowledge in AI. Those AI systems that incorporate it tend to be the more powerful *AI toolkits* such as ART, KEE and GoldWorks, (AIAI, 1987a,b,c) They are called AI toolkits because they provide a wide range of different AI techniques which can be used together in a unified environment.

3.3.4 Hybrid Approaches

A number of AI toolkits offer the chance to combine the above representations within a single system. Frames and production rules have already been described as a combination that can be used to good effect. Rules may also be used to send messages to objects, and this approach frees the rules from having to deal with integrity maintenance as the methods can handle this.

Combining frames and objects is also a possibility, indeed both may co-exist with rules in some toolkits. However, the system designer should exercise extreme caution in choosing a hybrid architecture, especially one involving all three techniques. Control in such a system can become impossible to follow or debug at a later stage. The simultaneous firing of methods and daemons, with rules waiting to fire on the results, activating more methods and daemons, can become a debugging nightmare which should be avoided if at all possible.

Having used both object and frame representations in the course of this research the author has noticed great similarity between objects and frames. Both allow for collections of facts to belong to a larger unit, and both can form hierarchies with inheritance. Both also have a dynamic capability, frames through daemons and objects through methods and messages.

Both objects and frames are essentially entities with attached procedures and triggering devices. The major difference is that with objects the message passing is explicit, wheres with frames and daemons the message passing is implicit. Accessing a particular instance slot effectively passes a message to the daemon for it to fire. Also frame based systems allow direct changes to be made to the slot values whereas object oriented systems do not.

Both frames and objects are conceptual models. One can use frames and daemons to emulate an object oriented approach. Likewise, using appropriate methods, object oriented systems can be made to behave like frames. Given this, there are few situations where both objects and frames are essential. The author would suggest that where structured data is needed with little dynamic capability, frames are the ideal choice. Where greater dynamic behaviour is needed there is less distinction and the choice should be based on the programmers preferences and the type of implementation software available.

3.4 Inference Methods

Production rules, as noted in Section 3.3.1, are the most popular form of knowledge representation. Making them work in the context of the problem in hand is the job of the inference engine. How the inference engine does this is the subject of the following sections. Three inferencing methods are introduced, each designed to use rules optimally depending on the type of problem being addressed.

3.4.1 Forward Chaining

Forward chaining is possibly the most obvious way to use a set of rules. If facts in the knowledge base match the rule antecedent, the facts in the rule consequent can then be said to be true. These new facts may cause other rules to match and so the process, or *chaining*, continues. Forward chaining derives its name from the way in which the process moves forward from the IF parts to the THEN parts of the rules. It is also termed *data directed reasoning*, as the existing data controls whether or not the rules match.

Forward chaining involves continuous scanning of the knowledge base by the inference engine. Any single change in knowledge base facts may affect which of the rules can fire and which cannot. Simple inference engines test rules in the order they appear in the knowledge base, firing them if they match. These systems cannot control the rule firing order effectively. More comprehensive systems draw up a list, or *agenda*, of rules that could possibly fire with current knowledge, and this is updated every time the knowledge base changes. This agenda is ordered, either using rule priorities or some other criterion. The top rule is fired and this alters the knowledge in some way. As the knowledge base has changed the agenda cannot be considered valid and must be re-compiled. Chaining continues until the recompiled agenda is eventually found empty, meaning that no more rules can be fired.

The effect of forward chaining is illustrated in Figure 3.4. Four example rules are given which refer to FACTS A to G. Assuming FACT_A and FACT_B are initially true the trace shows how the rules match and are fired. In this simple example no agenda is shown as only one rule matches at any time.

Forward chaining uses all the rules to infer every possible new fact. This is ideal in situations where all conceivable solutions need to be known. For example, a farmer may need to know all available herbicides which will kill weeds but not his crop (Colby, 1986). Forward chaining has the advantage that all consequences of a change are bound to be present in the knowledge base before it can be examined by the user.

The scanning process with forward chaining is processor intensive. Not only are all possible facts found whether needed or not, but in systems which use agendas a new agenda must be compiled for every fact found. Forward chaining systems can thus become very slow if large knowledge bases are used. However, if all possible facts are needed in the knowledge base then forward chaining provides the optimal solution.

3.4.2 Backward Chaining

Backward chaining, as its name implies, does the opposite of forward chaining. Instead of using the rules to discover all facts that are true, backward chaining tries to prove whether a particular fact is true. This fact is known as the *goal*. To prove the goal the inference engine examines the consequent elements of each rule to see if the goal would be satisfied if that rule were fired. When such a rule is found its antecedent is tested against the knowledge base to see if the rule can be fired. If this is the case the rule fires and the goal is proven.

If one or more of the elements in the antecedent are not in the knowledge base, they are issued as sub-goals. The system tries to prove these sub-goals in the same way, and thus the chain moves backward.

As the system chains back, a record is kept of all rules making up the chain. When rules are eventually found with antecedents that are fully satisfied, the record is used to fire all rules in the correct order until the goal is proven,

RULES IN THE FORWARD CHAINING KNOWLEDGE BASE

Rule 1
if FACT_A
and FACT_B
and FACT_C
then FACT_D

Rule 2
if FACT_B
then FACT_E

Rule 3
if FACT_A
and FACT_D
then FACT_F
and FACT_G

Rule 4
if FACT_A
and FACT_E
then FACT_C

Given a knowledge base where FACT_A and FACT_B are initially true the following would happen :

TRACE SHOWING THE EFFECT OF FORWARD CHAINING

1. Rule 2 matches and fires -> FACT_E now true
2. Rule 4 matches and fires -> FACT_C now true
3. Rule 1 matches and fires -> FACT_D now true
4. Rule 3 matches and fires -> FACT_F now true
   -> FACT_G now true

Figure 3.4 AN EXAMPLE SHOWING THE EFFECT OF FORWARD CHAINING

Backward chaining is more complex than forward chaining, and is best illustrated with an example, Figure 3.5. In this example the same rules are used as in the forward chaining illustration. The goal is to prove FACT_D, given that FACT_A and FACT_B exist in the knowledge base.

The trace shows rule identification, antecedent checking and the issuing of sub-goals. Once the backward chain is complete a forward chain follows to actually fire the rules and assert the new facts into the knowledge base.

Backward chaining only fires rules that are absolutely necessary in proving the goal. This can be seen from the example in that rule 3 was not fired (compare this with the forward chaining example). It may seem a trivial saving, but in a knowledge base of many hundreds of rules the firing of every possible rule just to prove one fact is a hopeless waste of processing. The limited rule firing does mean that the knowledge base is not always complete and consistent, but for goal proving problems this is not important.

The situation can often arise during backward chaining that a fact is not in the knowledge base or in the consequent of any rule. Unknown facts can be directed to the user by asking him a question using a pre-defined question form. Because backward chaining only considers relevant facts, the system appears to be asking 'intelligent' questions whilst it is trying to solve the problem. Expert systems derive much of their credibility from this type of 'intelligent' dialogue with the user.

3.4.3 Bi-directional Reasoning

Both forward and backward chaining have advantages for certain types of problem. Bi-directional reasoning uses both mechanisms within the same problem solving process, with the aim of taking advantage of the merits of each. Controlling such a process, and enabling the system to decide when it is best to change from one mechanism to the other is not straightforward. There appears to be no standard control strategy that is common to different shells or toolkits.

Rules In the Backward Chaining Knowledge Base

Rule 1
if FACT_A
and FACT_B
and FACT_C
then FACT_D

Rule 2
if FACT_B
then FACT_E

Rule 3
if FACT_A
and FACT_D
then FACT_F
and FACT_G

Rule 4
if FACT_A
and FACT_E
then FACT_C

If the knowledge base starts with FACT_A and FACT_B being true, and the goal is to prove FACT_D, then the following process would ensue :

Trace Showing Effect of Backward Chaining

```
1. To prove FACT_D Rule 1 needs to be proven
2.      -> FACT_A exists and is true
3.      -> FACT_B exists and is true
4.      -> FACT_C is unknown
5.              -> Rule 4 needs to be proven
6.              -> FACT_A exists and is true
7.              -> FACT_E is unknown
8.                      -> Rule 2 needs to be proven
9.                      -> FACT_B exists and is true
10.             -> Rule 2 proven, FACT_E now known
11.     -> Rule 4 proven, FACT_C now known
12. -> Rule 1 proven, FACT_D now known
13. GOAL PROVEN !
```

Figure 3.5 An Example Showing The Effect of Backward Chaining

In GoldWorksII bidirectional reasoning is termed *goal directed forward chaining* (GoldHill, 1989). Forward chaining rules are enclosed within *rule sets* which prevent them from firing as soon as they match. Instead each rule set has an *enabling pattern*, similar to the consequent part of a backward chaining rule. To the inference engine the rule set resembles a backward chaining rule. If, during normal backward chaining, the mechanism identifies the rule set enabling pattern as something which may satisfy a goal, the rule set opens and the rules inside it forward chain. Backward chaining is suspended whilst this is in progress, and it is resumed afterwards.

Goal directed forward chaining is used when forward chaining rules may help during backward chaining, by adding facts into the knowledge base which are relevant to a specific area of the domain. Alternatively it can be used when certain operations are necessary if particular areas of the domain are being explored.

An example would be where a goal depends on facts in a database. If the database is only relevant in a small number of cases within the domain, it need only be examined when these cases are a part of the current goal proving problem. Backward chaining rules cannot read databases (or execute any other operation) until they are fired *after* the goal proving facts have been found. Clearly if the goal depends on the facts in the database a 'Catch-22' situation exists, the rules cannot fire until they have the facts from the database, yet cannot get those facts until they are fired. If this were the only method available the rules would never fire and the goal would not be proven. Forward chaining rules within a rule set, however, can read the database *before* the goal is proven, making goal satisfaction possible.

If the method is used instead of pure forward chaining, the effect is to compartmentalize forward chaining rule base. Only those rule sets relevant to the problem are actually used. In effect, the rule base is split into a number of broad categories represented by goals. The area of the domain which is relevant to the problem is first found, then only appropriate rule sets are opened. A rule may belong to any number of rule sets, so repetition of rules in different rule sets is unnecessary. The saving in processing time and memory space can be substantial as irrelevant areas of the domain can be excluded from forward chaining.

Readers interested in the factors affecting the choice of inference mechanism beyond those outlined here are referred to the work of Reichgelt & Van Harmelen

(1985). An example of different inference strategies being used on the same rules, illustrating the qualities of each, is given by Oxman & Gero (1987).

3.5 UNCERTAINTY

Expert knowledge is often not absolute. Uncertainty can exist which the expert takes into account during the normal course of his work. If the uncertainties are considerable, he may qualify the answers he gives by stating an approximate degree of confidence that he has in his judgements.

The importance of uncertainty varies with the domain, but it is clear that in many areas expert systems have to take uncertainty into account. Johnson & Keravnou (1988) note that in the early expert systems MYCIN and PROSPECTOR, two types of uncertainty were handled, that associated with rules and that associated with supplied evidence. The distinction is important as uncertainty in rules must be estimated by the expert during knowledge acquisition, whereas uncertainty in the evidence must be estimated by the user during expert system application. Clearly if the expert and user calibrate uncertainty differently large errors can be introduced into the result.

Natural language and its use in conversation illustrates the possibility of different individual perceptions of the meaning of words. Expert systems with no uncertainty handling could ask the user a question such as "Was the animal large or small ?". The perception of an individual when considering a dog, for example, can completely alter how this question is answered. Hence the whole line of reasoning pursued by the system can be flawed by perceptive differences between the original expert and the user simply by the meaning of such words as "large". If an expert system is to rely on descriptive language, the precise bounds of word meanings, or typical examples falling into each category, should be provided for the user wherever possible to minimise these distortions.

If a system is to use a more formal representation of uncertainty a number of methods have been proposed. Frost (1986) identifies and explains six theories which he believes to be the most useful. These are :

- Probability Theory
- Certainty Theory
- Dempster/Schafer Theory of Evidence

- Possibility Theory/Fuzzy Logic
- Incidence Calculus
- Plausibility Theory

The author has found not only a wealth of publication on uncertainty, but also a marked variance in opinion as to its usefulness in expert systems. It cannot be doubted that in some expert systems modelling uncertainty has been of use. However, getting the expert or the user to estimate uncertainty values is fraught with difficulty. Hart (1986) notes that much of this is due to people misunderstanding probability, especially in compound events. Slatter (1987) argues that people do not naturally think in terms of probability or statistics anyway, and that Bayes Theorm and other statistical techniques have been effectively refuted as psychological hypotheses about everyday decision making.

Clearly uncertainty is a subject which cannot be ignored in expert systems. However, the form which uncertainty representation should take is controversial, and the actual reliability of expert and user alike in making uncertainty judgements is open to question. With this backdrop it was decided that, whilst the work done in this research should allow for uncertainty representation, the knowledge acquisition (Chapter 5) would not attempt to quantify uncertainty at this stage.

3.6 SUMMARY

Artificial intelligence techniques are primarily designed to manipulate symbols rather than numbers. The symbols can be used to represent real objects or abstract ideas, and relationships may be defined to connect them. These techniques have been applied to modelling human expertise, with such systems being called *expert systems*.

A variety of methods were introduced for the creation of an expert knowledge base. Of these, production rules, using an IF..THEN rule format are a perhaps the most straightforward. In isolation, however, they can be limiting and other representations, like frames and objects, can complement the use of rules by giving structure to the symbols that the rules manipulate.

Rules are statements that can infer new facts given one or more existing facts. In order to use rules to produce additional facts one must use some form of inferencing mechanism. Forward and backward chaining were discussed, the former being optimal in situations where all possibilities need to be known, and the latter when only a few facts need to be proven. Bi-directional reasoning combines the approaches in the hope of gaining from the advantages of each.

Uncertainty exists in many aspects of everyday life. Many things are 'more or less' correct, or happen 'most of the time'. The ability to represent uncertainty within an expert system is important in some application areas where categoric rules and facts do not exist. Various techniques may be employed but all suffer from problems, in that the expert and user must somehow quantify their uncertainty. How certain they are about this quantification adds a new dimension and further difficulty to the approach.

Whilst AI systems have advanced capabilities for structuring, understanding and reasoning with information, the amount they can actually store and manipulate tends to be limited. Some AI techniques rely on holding all facts and rules in the main memory of the computer, others can take a great deal of processing time when the data volume becomes large. In either case improved techniques of information handling would be of great benefit in situations when AI is used with large data volumes.

CHAPTER 4

4. INTELLIGENT GEOGRAPHIC INFORMATION SYSTEMS

4.1 INTRODUCTION

In previous chapters the strengthsandweaknesses of artificial intelligence (AI) techniques and geographic information systems (GIS) have been discussed in isolation. AI provides complex reasoning and representation strategies, but lacks the ability to use large data volumes. GIS by contrast, store and manipulate large data volumes but lack a sophisticated reasoning capability. The potential of combining AI and GIS into a hybrid system is currently being investigated by research groups in both component disciplines. Such a hybrid, an *intelligent GIS*, has clear potential in overcoming the limitations of its individual parts.

This Chapter reviews existing projects that have attempted to marry AI and GIS. The projects cover a wide variety of specific application areas and disciplines. The aim of the text is to isolate salient factors used in the creation of AI/GIS hybrids, and highlight any negative aspects which could be avoided in future work.

This thesis aims to determine the potential of intelligent GIS as a tool for design applications. Prior to the more general review above, attention is paid to areas where AI has been used with *spatial,* as well as geographic, data for design. Spatial data may be of a more restricted form than that typically regarded as geographic data, the layout of cupboards in a kitchen, for example. The discussion leads into a more conceptual discourse on the philosophy behind spatial representation and reasoning, with the aim of finding a theoretical basis on which to build an intelligent GIS.

4.2 AI AND SPATIAL DATA IN DESIGN

Design using spatial data covers a multiplicity of disciplines. Of these, applications using geographic data are more limited. However, they include such diverse tasks as landscaping, site layout and positioning, the routing of pipelines, roads, cabling and other linear structures, and just about every design activity where a map is used.

A number of projects have considered automating or aiding the design function using AI with spatial, rather than strictly geographic, data. Bowen et al (1986) exemplify the approach with the BERT brickwork design expert system. BERT allows the engineer to produce elevation drawings of building faces on an AutoCAD computer aided draughting system. BERT then converts these drawings into a text based description, and this is used as input to the expert system. By analysing the description BERT is able to give counsel on the proposed layout with respect to it's efficiency when built in brick. It can advise the engineer, for example, to change the position of an expansion joint or window to minimise brick cutting. These comments are reported as text and the engineer is left to update the drawing himself.

BERT is able to function because it restricts the scope of the spatial information needed. Building elevations are two dimensional surfaces and BERT only allows certain component types to be used, for example windows, doors and joints. This restriction must exist to allow unambiguous translation from the drawing to the text based description which is needed before the expert system can work. The translator must explicitly recognise each component in the design before it can operate.

Coyne & Gero (1986) use the term *syntactic to semantic interpreter* to describe the program which translates from a spatial description to a text based description. They note that "points and lines can be regarded as syntactic representations, the semantic descriptions of which are the building components themselves. The building components may, in turn, be considered as syntactic elements and their meanings described in terms of compositions and performances". Essentially Coyne & Gero are stating that a spatial description alone is seldom enough for design decisions to be made. It is the meaning or function of the objects represented by the points and lines, *in the context of the design problem*, that is of fundamental importance to any decision. BERT translates the CAD drawings into a specific and restricted vocabulary of objects and relationships which are understandable to both the rules and the engineer. Logic within the expert system can then be used to infer any changes needed in the design. For example, if the distance between a wall and a window were significant, it could be semantically written

distance (wall, window, 150)

A rule requiring a minimum distance of 250mm could then recognise this fact and, using standard logic, infer that a problem exists.

Oxman & Gero (1987) illustrate the interpreter functioning in PREDKIT, an expert system for layout design in kitchens. This improves on BERT in that it can express its findings in graphical form, and can generate designs as well as diagnosing faults in existing designs.

These examples and others (Green, 1986) show that, whilst design expert systems using spatial data are possible, the spatial content of existing systems is restricted to items that can be readily translated into a semantic form. Thus the spatial objects described tend to be simple.

Geographic information, by contrast, is far from simple. The engineer using geographic information cannot restrict the number, location or complexity of the features which exist on the ground. The spatial interrelationships, which can fundamentally affect the design, are complex and ill defined. The relationships 'distance' and 'direction' lose their meaning when the objects they relate are irregularly shaped regions. There is no formal definition of such terms which can be used to categorically define such relationships.

It can be argued that geographic information is a superset of the spatial data commonly used in design. It includes but goes beyond the bounds of simple objects and relationships, at the same time making 'syntactic to semantic interpretation' a far from trivial matter.

Projects that have used unconstrained (ie. complete) geographic information and AI for design are rare. Chandra & Goran (1986) report on a system for site selection, but although this uses unconstrained data, it is essentially just searching the database for areas which match the required site selection criteria. The lack of an iterative sequence of changes would suggest that it is not a design system but merely a selection system.

Design problems that involve an iterative sequence of improvements leading to the eventual artifact, and these are more complex. Routing of linear structures and communications is an example where the efficiency of the design not only depends on route length, but on the wide ranging effects that different geographic features can have on a route. These would not only affect cost, but also the constructibility and operating efficiency. As such, routing typifies the complexity of design problems using geographic information. This was a major reason for its choice as an example application for the research in this thesis.

Routing has been a problem for civil engineers for centuries. The Romans took a rather dogmatic straight line approach, but since then engineers have assessed existing geography and moved the route around features where there was a clear advantage in doing so. In recent years computer programs have allowed highway engineers to rapidly assess cut and fill earthwork volumes for their proposed routes (Finniear, 1986a). Research is now in progress using GIS to itemise the type and extent of features crossed by pipelines, although AI is not being employed (Coleman, 1989). These help to provide the base data for design decision making, but fully automated route design is a far more difficult concept. Those attempts which have been made all show limitations in either the data they use or the quality of the results they achieve.

Route finding and optimisation have been an area of interest in disciplines outside engineering, including robotics and flight planning. Dobbs et al (1988) address the problem of routing a military aircraft through a hostile environment by using algorithms based on heuristics or 'rules of thumb'. The location of potential threats cannot be constrained, but the system classifies the threats themselves into basic severity categories and considers them to be points with an attached 'conflict cost' and a circle of influence. The heuristic search algorithm attempts to find routes which minimise the total conflict cost. By describing threats as they have Dobbs has constrained the problem to a two dimensional path generation and test method, where at each step the threat points near to the 'aircraft' can be considered in terms of the single 'conflict cost' variable. By contrast, designing routes for engineering works is generally a multivariate problem, which cannot be simplified to a single variable comparitor. Also the geographic features themselves cannot be simplified to points or circles.

Oshima et al (1986) propose using a GIS for automated routing by suggesting a stepwise procedure, allowing the route to 'grow' from start to end. At each step alternative routes (radiating at 10° intervals) are generated and tested. The most suitable is found by assessing its direction and a 'weighting factor'. The route then steps along to the next point. This is effectively a blind search, with the system having no forward looking capability and no understanding of the form and spatial layout of the geographic features in the region. Typically an initially favourable path could wind its way down a 'canyon' to a dead end, with no appreciation of its demise in advance. Deriving a realistic single weighting factor on which to base the step assessment is also limiting. In some applications, such as the cross country movement of military vehicles, multivariate data can be condensed into a single variable representing the maximum traversal speed at any point (Loomer, 1986). However, for engineering applications with high capital value, such simplification is seldom realistic.

Vehicle location and route selection are cited as being of great potential as a GIS application (DoE, 1987). Karimi et al (1987) propose a real time expert system to find the best route for a vehicle on a road network. The network constraint acts to make the number of possible solutions finite, as there are only a fixed number of choices at each road junction. Thus the selection of suitable routes is far easier than in the unconstrained case. A quoted advantage of this system is that information about the condition of the roads can be updated in real time, with new routes being suggested to cope with the changing situation. This is where the AI approach excels as it can be designed to deal with dynamic changes to facts in the knowledge base. Were it not for this capability the system would have little to distinguish it from conventional routing systems such as AutoGlide (DoE, 1987).

In all the above projects the unconstrained use of geographic information has either not been required, or has been compromised in order to make the application practical for automation. BERT and PREDKIT limit spatial objects to simple components that form a restricted language for use with design rules. Geographic design applications, which must use features in unconstrained locations, have had a choice of simplifying objects to plain geometric shapes, condensing multivariate data to univariate data, or blind path generation and testing. None have successfully tackled the problem of using unconstrained multivariate geographic data for fully automated design.

It is clear from BERT and PREDKIT that a language to express spatial objects and relationships, together with a logic to manipulate it, can make automated spatial design possible. Without this any automated spatial design system effectively becomes blind, the system probing for information in space without a global appreciation of the whole spatial picture. It is reasonable to assume therefore, that intelligent automated design using unconstrained geographic information would need a similar formal language and logic before it could be successful.

The scope of the research was widened with two major aims, to see if there was significant progress toward a formal language for geographic space, and to discover what efforts had been made to practically link AI and GIS to date.

4.3 Spatial Representation and Reasoning

Attempts have been made to define spatial languages and logics. In the main this effort has come from the pure AI research community where spatial reasoning has a far wider application than the use of geographic information alone. Popplestone (1979), Davies (1986), and Fisher (1987), typify the AI approach in the areas of robotics and scene analysis. Data volumes used, however, tend to be small enough to hold all data in the working memory of the computer. This permits AI techniques to be used which could not be employed with large geographic databases.

Peuquet (1987) has made significant efforts in the formal definition of geographic objects and relationships, with a view to creating a language to describe space. She notes that there are two types of query that can be put to a spatial database :-

- Given a specific object (or objects), what are its associated properties (one of which may be location) ?
- Given a location, what objects exist there ?

The paper argues that these are logical duals, and a dual structure for modelling spatial phenomena is suggested. The proposed structure consists of an object based description (with location as one of its properties) and a location based description (with objects existing at a location being properties). Spatial dualism forms an intrinsic part of a hybrid intelligent GIS called KBGIS, which is described in Section 4.4.3.

In an attempt to define spatial relationships Peuquet (1988) adapted the work of Jakendoff (1983) from its use for general word meanings into a specific spatial context. Jakendoff proposed that three conditions were sufficient to define the meaning of any word:-

Necessary Condition	things which MUST be true for a word to apply
Centrality Condition	A central or threshold value for variables within which a word applies
Typicality Condition	Features which would normally be associated with the application of a word

Figure 4.1 shows the Jakendoff conditions applied to the direction relationship. Despite this Peuquet failed to define a spatial algebra, observing that "there are a bewildering number of potential spatial relationships with seemingly infinite variations". After reviewing such diverse conceptual tools as the relational data model, directed hypergraphs, if-then-else rules, uncertainty and semantic/associative networks (such as frames), she suggests that "the most productive approach toward specifying the functional relationships between spatial elements would seem to be a combination of all these mechanisms, using each to its best advantage".

Other researchers have addressed the spatial language problem and trends are apparent. The use of first order predicate logic, the basis of the AI languages Lisp and PROLOG, has been recommended by several groups as a possible solution. These papers are reviewed in more detail in Section 4.4.3. However, it is clear that much fundamental work is needed before a language and logic for space can be said to exist.

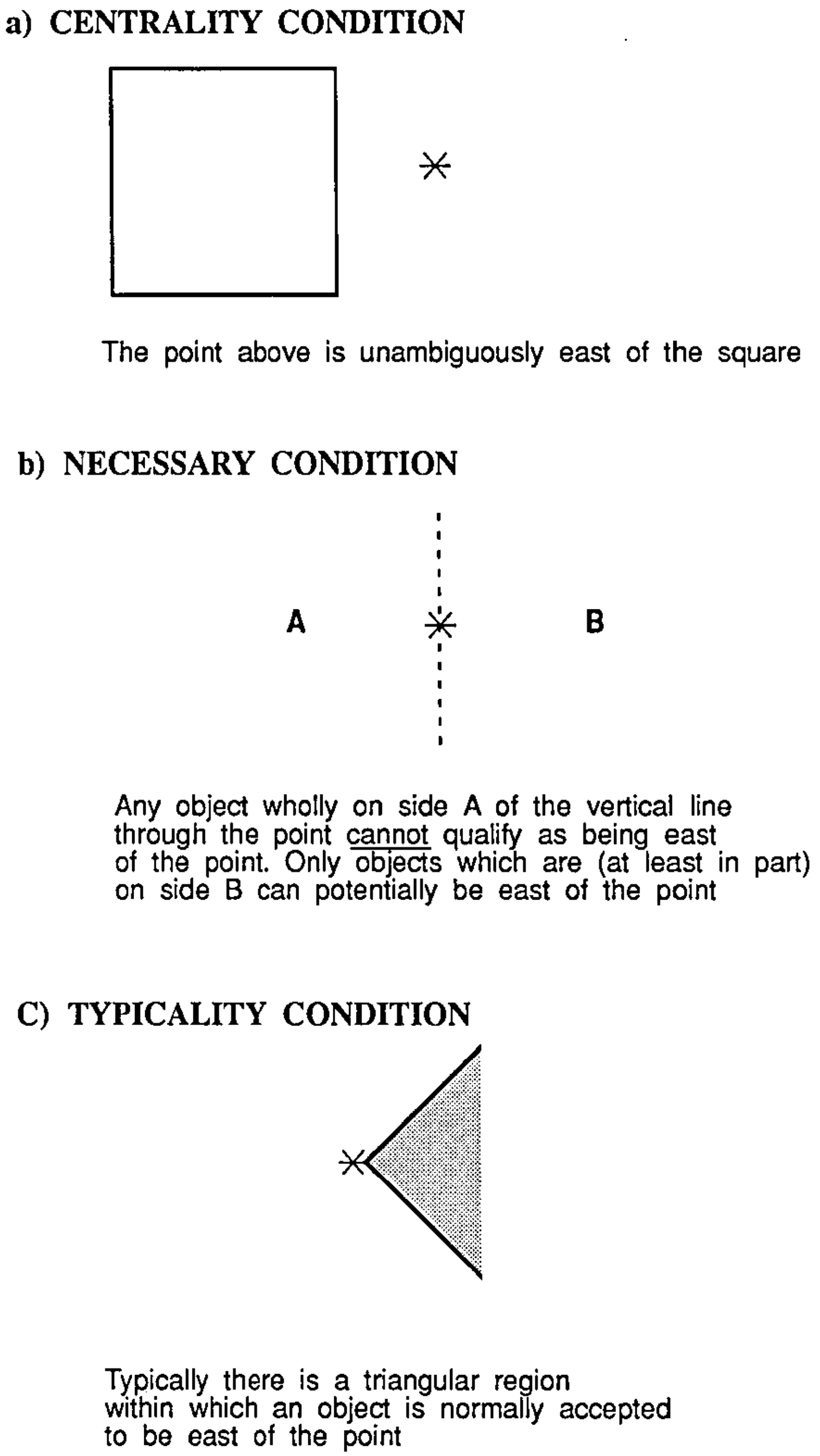
4.4 EXISTING PROJECTS

Artificial intelligence techniques have already been applied to GIS with varying degrees of successs for a variety of purposes. Reviews by Robinson, Frank and others identify four key areas of exploitation :-

- Map Design Systems
- Terrain Classification/Feature Extraction Systems
- Geographic Database Management Systems
- Geographic Decision Support Systems

(Robinson et al, 1986a,1986b; Robinson & Frank, 1987a,1987b; Egenhofer & Frank, 1990)

Ripple & Ulshoefer (1987) add a fifth category, intelligent user interfaces, and this is treated as an additional section in this chapter.



a) CENTRALITY CONDITION

The point above is unambiguously east of the square

b) NECESSARY CONDITION

Any object wholly on side A of the vertical line through the point cannot qualify as being east of the point. Only objects which are (at least in part) on side B can potentially be east of the point

C) TYPICALITY CONDITION

Typically there is a triangular region within which an object is normally accepted to be east of the point

Figure 4.1 JAKENDOFF CONDITIONS APPLIED TO THE DIRECTION RELATIONSHIP (after PEUQUET, 1988)

cd\5786T

4.4.1 Map Design Systems

One of the most common uses of GIS is in the production of maps. An agreed system of scales has made manual map creation a fairly standardised process, with skilled cartographers knowing what level of detail is required (or possible) at any particular scale. However, a certain amount of 'artistic license' has always been needed to decide where best to put text, how to generalise, include or omit features, and how to design the layout effectively. Adding AI to a GIS may enable these decisions to be made automatically.

Automatic generalisation of maps between scales is the aim of the OSGEN system (Robinson & Zaltash, 1989). Implemented in the Leonardo expert system shell, OSGEN attempts to model the reasoning of the cartographer. It is designed as a text based advisory system with no interface to a GIS. Rules were written to judge situations such as whether or not to include a building, whether to symbolise an object, and whether to combine, simplify or exaggerate features.

However, Robinson describes the rules as "hard" and "soft", the latter being inconsistent yet at times overriding the former. It is apparent from this that the OSGEN model does not contain enough "hard" knowledge to mimic the manual process, "soft" rules merely cover areas where knowledge acquisition is incomplete, and other rules should be acquired to state when a rule is applicable and when it is not. As a text based advisory system OSGEN relies on the user to describe the situation surrounding map items, and this can be a source of inconsistency between different operators. A more complete system would attempt to use the map data directly.

Fisher & Makanness (1987) ask the question "Are cartographic expert systems possible ?", concluding that they are provided cartographers "get in on the act" by adding their expertise to the projects. Robinson & Frank (1986a) reinforce this, noting that the MAPEX and AUTONAP systems were both built without consulting the appropriate experts in map generalisation.

Although map design appears closest to the general concept of design using AI/GIS, the problem is more like that addressed by PREDKIT, Section 4.2. Map text, for example, is essentially a box symbol which must be placed in such a way as to satisfy constraints caused by surrounding symbols. It does not involve the use of the geographic information at a semantic level, except when prioritising the relative importance of each symbol to the eventual use of the map. As such this subject area did not offer any significant pointers for this research.

4.4.2 Terrain Classification and Feature Recognition Systems

The data collected for a GIS does not always provide enough structure or detail for the intended application. This is particularly true of remotely sensed imagery where the data is either a photograph or a digital image of unconnected pixels, Section 2.6 . With the increasing availability of remotely sensed data, particularly from satellite (House of Lords, 1983), new methods for analysing the images using AI have been investigated.

Conventional image processing relies on statistical and arithmetic methods to stretch and stratify the image, and clustering techniques for identifying regional features (Bagot, 1985). A skilled human is required to operate the system and classification relies on the expertise of the operator in understanding the significance of the resulting data. The process can be time consuming and expensive.

Other methods have been tried, such as fractal analysis of images for land use assessment (DeCola, 1989). However, many researchers recognise the potential of using the AI technique of pattern recognition in analysing both remotely sensed images and other geographic data.

Palmer (1984) performed some inspiring early work on the analysis of terrain features. Starting with a triangular irregular network (TIN) of a terrain surface he used PROLOG rules to search for ridge lines, valleys, peaks etc. His hypothesis was clear: if a rule could be written to describe a feature in terms of its terrain representation in PROLOG, then the PROLOG search strategy should be able to find all occurrences of that feature. The TIN was defined in PROLOG as nodes, line segments connecting nodes, and cells corresponding to the triangles formed. A peak, for example, could then be found by searching for a node that had no connected node higher than itself.

Frank et al (1986) recognised the potential of this approach for the formal definition of geographical terms, such as "watershed" and "ridge line". He argued that this may provide a basis of a formal language for spatial description and reasoning.

In feature classification the expertise of the interpreter has been noted as a possible target for an expert system. Peacegood (1985) used a text based expert system to achieve precisely this. It relies on the analyst describing the shape and properties of the data of a potential feature, and then employs rules to suggest possible features to match the data given. Hadipriono et al (1990) found text based methods adequate in a project to analyse drainage patterns from remotely sensed data. However, such approaches still rely on the operator to correctly describe the images.

Identifying features direct from digital imagery is far more complex. Human operators often use other data (such as maps) to provide a contextual base which helps in the interpretation. Skidmore (1989) proved that a terrain model, geometrically co-registered with Landsat imagery, can help an expert system to classify forest cover types. Van Cleynenbreugel et al (1990) reinforce this in their system to recognise road structures. The system uses a terrain model and SPOT imagery as base data, and heuristics to reliably pick out roads and paths. A typical heuristic is "in mountainous regions roads tend to follow contours". These projects clearly demonstrate the need to integrate different data types within an intelligent GIS.

Image classification and feature recognition will continue to attract researchers as satellite data becomes more detailed and freely available (Estes et al, 1988). Recent work by Frankot & Chapella (1990) illustrate this continuing commitment with their computer vision approach to image analysis. The interpretation by the system of Venusian synthetic aperture radar images has produced maps of the surface topography of this shrouded planet.

4.4.3 Geographic Data Base Management Systems

Conventional GIS research has made a number of significant advances in data representation and handling, as was seen in Chapter 2. However, the definition of a formal spatial query language and the rapid search and manipulation of large heterogeneous spatial databases still need much research. AI techniques have been employed in several projects addressing these issues, and they give some useful ideas on potential structure and integration methods for intelligent GIS.

Traditional database query languages, such as SQL (Van der Lans,1989; ISO9075,1989), can certainly be used to access spatial objects stored in databases, but it is difficult for them to express queries which involve particular spatial properties. For example, requesting the distance between two objects is not possible with traditional query languages, which lack geometric concepts to perform the operations required.

First order predicate logic, in the form of the AI programming language PROLOG, was first proposed by Frank (1982) as an alternative for spatial query processing. Forbes (1984) reinforced this proposal, noting that PROLOG could provide a simple language for both data definition and query, and that "virtual relations" can be set up within the data that are far more powerful than conventional relations. Frank (1984) acknowledged that, as a purely memory based language, PROLOG was of limited use with large geographic data sets. He overcame this by incorporating DBMS extensions to create a *persistent PROLOG*. By manipulating data on disk rather than in memory, persistent PROLOG allows much greater data volumes to be handled. LOBSTER is a geographic database system written using persistent PROLOG (Frank, 1984). The work of Palmer (1984) on feature recognition has been implemented as an example of the functionality of LOBSTER (Eganhofer & Frank, 1990).

The continuing faith in the suitability of first order predicate logic as a spatial logic is exemplified by Menon & Smith (1989), who use Lisp as a base language for experimentation.

The most influential intelligent GIS project to date was first reported by Smith & Pazner (1984), Peuquet (1984) and Chen (1984). Called the Knowledge Based GIS (KBGIS), the stated aims for the system were to answer queries rapidly, and to learn about spatial objects. KBGIS is firmly geared toward providing fast access to data in large heterogeneous databases. Its database is conceptually split into dual location based and object based data structures to allow rapid access, advocated as spatial dualism by Peuquet (1987), Section 4.3. The location based representation is implemented as a quadtree, whilst objects are defined in Lisp in a frame-like tree structure. Crucial to the project is the ability of KBGIS to 'learn' through memorising examples of objects for which it has already searched, so that future searches can be answered directly. Smith et al (1987) describe this as "rote" learning, and also add inductive learning to the KBGIS, allowing the system to define new object types given a series of examples.

KBGIS has continued developing to the present day (Albert, 1988; Smith, 1988; Campbell & Goettsche, 1989). The full structure of the system is involved, and detail can be found in the referred papers. Key points which have been noted for the authors research are :-

- The use of location based and object based structures
- The use of hierarchical trees for objects
- The use of Lisp as a control language

Donna Peuquet, a principal researcher in the KBGIS group, was approached directly by the author in the hope that KBGIS could be obtained to act as a base system for the authors work. However, she stated that unfortunately KBGIS was not a coherent system, but a collection of individual programs that could not easily be transported or used (Peuquet, 1989). The implication was that whilst KBGIS is impressive in theory, the practical system is merely a collection of research prototype modules which do not form a reliable, integrated package.

4.4.4 Geographic Decision Support Systems

Decision support has been a principle use of GIS since their inception. It is a broad category, applying wherever data provided by a GIS forms the basis of some form of decision making. Typical applications include land use management, vehicle scheduling, supermarket site location and, of course, design.

Robinson & Frank (1987a,b) review several applications where AI has been used to aid or replace the decision maker using geographic information. Of these ASPENEX, a system for advising on the management of aspen forest, is perhaps the most successful (Morse, 1987). ASPENEX consists of a personal computer (PC) based expert system containing rules about aspen stand management. The GIS is based on a mainframe computer. Control software on the PC initiates ASPENEX by requesting a spatial analysis of a particular stand from the GIS. This is returned as a text file which the expert system uses as a basis for its management recommendations for the stand. ASPENEX has a one way control structure, and the expert system itself does not have direct access and control over the GIS during processing. In the context of the problem, however, this loose integration is adequate as the GIS data does not change after the initial consultation, and no iterative analysis is undertaken.

4.4.5 Intelligent User Interfaces

Existing GIS tend to require skilled operators, not only to drive the system, but also to formulate the types of query needed to produce the desired map or other information. AI techniques have been harnessed to reduce this complexity, with the aim of allowing the end user of the data to retrieve it himself.

Kubo (1986) describes TRINITY, a GIS with an intelligent interface which learns the individual habits and requirements of its users. On initialisation, TRINITY asks for the users name, and then loads the appropriate user knowledge base for the interface. The user can drive the system using natural language phrases or menus, and even a kanji (Chinese character) interpreter is provided. A thesaurus is used so that words other than those in the TRINITY command language can be understood.

For interrogating distributed geographic archives Stoms et al (1988) have created BROWSE, a text based advisory system written in VP Expert, Section 6.3.1. Although not of great technological significance BROWSE does show that even very simple (and inexpensive) expert system shells can be successful in appropriate geographic applications.

Other reported intelligent interfaces include an X-Windows based front end for the KBGIS. (Campbell & Goettsche, 1989). However, this is limited to the ability to define mouse sensitive buttons and menus on the screen which protect the user from having to use the KBGIS spatial object language to create queries. The project falls far short of the intelligence of the TRINITY interface. Robinson & Frank (1987a,b) and Ripple & Ulshoffer (1987) adequately review other contributions to this area.

It should be noted that engineers using an intelligent GIS for design will not necessarily be expert GIS users. Hence an intelligent, application oriented interface may be crucial to the eventual usability of a system.

4.5 Summary

Intelligent GIS have been created successfully to perform a variety of tasks, such as map generalisation, feature recognition, geographic database management and decision support. Design using unconstrained geographic information is not among these applications.

Automated design has been achieved using AI and spatial data. Spatial data in this context refers to data more simple that than the diverse, irregular objects commonly found in geographic data. The key factor appears to be the translation from a *syntactic* representation of points and lines, to a *semantic* representation. Essentially a semantic representation is a restricted language, by which the spatial situation is described in terms that are significant to the problem. Rules can then be written to recognise facts represented using this language, and standard logic can be used to infer new facts. Other systems have used geographic information for design, but have simplified the data so that either it can be interpreted semantically, or they have resorted to a blind 'generate and test' process of arriving at a solution.

Automated design using unconstrained geographic information has not been achieved. Unlike the situation with simple spatial data, geographic information has no restriction on the number, shape or types of object that exist. Relationships such as 'distance' and 'direction' lose formal definition with irregularly shaped objects. Efforts to produce a formal language for the definition of geographic objects and relationships have failed. Until such a language is found, fully automated design using unconstrained geographic information will not be viable.

Nevertheless, any intelligent GIS for design, fully automated or not, must represent geographic objects in some way and reason with those objects. Conclusions from those who tackled formal language research were felt to provide important guide-lines for geographic representation and reasoning.

Spatial dualism, the object based and location based representation of geographic features in a database, was shown to provide rapid data access.

A multi-paradigm approach, using a mixture of rules, frames or objects, and the relational DBMS data model, was suggested as potentially the most promising for the creation of a truly integrated intelligent GIS. First order predicate logic has also been used by several researchers who testify to it's virtues for geographic data representation and query processing. [The author notes that an artificial intelligence toolkit could provide an integrated environment for these paradigms].

Existing intelligent GIS, such as ASPENEX for example, generally rely on *attachment*, rather than integration, between the AI and GIS components. The KBGIS, by contrast, is an example where close integration has been achieved. KBGIS was built purely for making access and manipulation of large GIS databases more rapid. A system structure like KBGIS, displaying close integration, is more likely to succeed in design applications where an iterative sequence of events demands a close dialogue between the AI and GIS components.

CHAPTER 5

5. DEFINING AN INTELLIGENT GIS FOR DESIGN

5.1 INTRODUCTION

Previous chapters have shown that intelligent geographic information systems are possible and have been built to help with a number of tasks. However, an intelligent GIS has yet to be constructed which effectively demonstrates the potential of the technology as an environment for engineering design.

British Gas plc., the sponsors of this work, and J.P. Kenny & Partners, an off-shore design consultancy, share a common interest in the potential use of intelligent GIS for design. This common interest lies in the siting of pipelines on the ocean floor. The design of off-shore pipeline routes is both time consuming and tedious. The nature of the off-shore environment is such that data and design parameters are prone to change at late stages in the design. This demands a great deal of manual re-assessment, wasting weeks of design time and possibly disrupting construction schedules. Both companies were keen to see the possibilities of intelligent GIS explored as a tool for pipeline route design.

Pipeline routing is an ideal example for testing the concept of intelligent GIS for design. The location of the route is of paramount importance, and the optimal result depends not only on route length but on the geographic features that the pipeline traverses. The assessment of the effects of such features on the pipeline requires the expertise of the pipeline designer. This expertise would have to be represented within an intelligent GIS carrying out the route design function.

In Chapter 4 systems were discussed which attempted to perform routing fully automatically, but these were shown to be ineffective in complex engineering design situations. Initial discussions with the co-operating organisations revealed that, rather than a fully automated 'black box' producing routes, they saw the engineer's involvement in the routing process as crucial. Engineers are wary of 'black boxes', and probably wouldn't trust one. What was needed, it seemed, was a system which would take out all the time consuming, tedious work in the design, whilst still giving the engineer control over the routes.

It was decided to build an intelligent GIS for the routing of off-shore pipelines. To satisfy the broader aims of the research the system would be constructed such that its conceptual structure is independent of the specific application. However, knowledge about pipeline design would be included in the implementation. The system was given a name - PIRATE - the Pipeline Route Analysis and Testing Environment.

This Chapter explains how the PIRATE system specification was drawn up. Two themes run in parallel throughout the text, each inextricably linked with the other. The primary, though not the dominant, theme is the PIRATE specification itself, what facilities the system must have and what knowledge should be incorporated. The secondary, but dominant, theme discusses the method by which the PIRATE system specification was found. *Knowledge acquisition*, an ill defined technique of isolating knowledge for expert systems, was used to uncover the PIRATE specification, mainly by interviewing real pipeline engineers. Conclusions from the knowledge acquisition process are important as an addition to the small body of work in the discipline, although these conclusions are secondary to the main thrust of the thesis. The two themes are inextricable because the result of knowledge acquisition *is* the PIRATE specification. The chronologic sequence of events is also important. The author hopes that the Chapter itself is slightly easier to understand than the explanation just given.

5.2 System Design Strategies

A crucial part of conventional computer systems design is the systems analysis. According to Longley & Shain (1989) systems analysis is a technique involving the analysis of an activity or system, to determine if and how the system may be improved using computer systems. As a formal methodology it has been in use for many years, and is documented in various texts, for example Gane & Sarson (1979).

Knowledge acquisition is an analogous function used when assembling knowledge for an expert system. It is less well defined and less well documented, the technique only coming into existence with the advent of expert system tools. This immaturity is bourne out by the fact that the first book exclusively dealing with the subject was only published in 1986. Anna Hart, the author of this work, compares knowledge acquisition to systems analysis (Hart, 1986). She notes that in systems analysis it is the eventual system *user* who is the information source, and it is fairly clear what information is required before the analysis begins. By contrast, in knowledge acquisition it is the *expert* in the subject who is the knowledge provider, and it is *not* clear at the outset what information will be needed. Knowledge acquisition is therefore a less well defined and less straightforward methodology.

In the execution of engineering designs, engineers usually perform actions in a pre-defined sequence to achieve a result, even though they may be considering many alternative solutions to the design. It was made clear at the outset of the project that pipeline designers also follow a sequence of tasks to route a pipeline. At points along the sequence design judgements are made involving many alternative solutions.

Allwood (1989a) suggests that, in identifying a suitable task for expert systems, the over-riding characteristic to look for is whether the problem solving method is essentially sequential or not. Sequential methods, he notes, are likely to be best tackled using conventional programming technology rather than by using expert systems. If, instead of a definite sequence, the problem solving method involves the simultaneous consideration of many alternative solutions, any one of which may be the right answer, he suggests that databases or expert systems are likely to be the right implementation choice.

The spirit of Allwood's categorisation is fully supported by the authors experiences. However, the mutual exclusivity of the categories causes difficulty when attempting to classify pipeline routing, and to some extent design functions in general. The design function bridges the categories and in doing so fails to belong to either.

The question arises, then, as to whether an expert system could provide a suitable base for a pipeline route design system. Overall control in the manual design method appears procedural, yet the decisions made during the procedure can only be described as requiring expertise. The computer system design must reflect this. It was therefore envisaged at the outset of the project that a computer system for pipeline route design would consist of an overall procedural control structure, with access to knowledge bases for judgemental decision making at appropriate points in the procedure.

With the overall structure of the system appearing to be procedural, formal methods of systems analysis were first looked to in order to ascertain the work flow. However, the procedure for pipeline routing is not well documented, apparently tending to vary depending on the nature of the design. Conventional systems analysis, according to Hart (1986), would not offer the facilities to cope with this type of information extraction. Knowledge acquisition methods were therefore applied despite the analysis being apparently for a mainly procedural system structure.

5.3 KNOWLEDGE ACQUISITION

The aim of knowledge acquisition is to create a correct and complete description of the experts knowledge of his subject. At the start of this research the only authoritative work dedicated to knowledge acquisition was the book by Hart (1986), although others had contributed prior to this in more general works on expert systems and AI (Feigenbaum & McCorduck,1984; Hayes-Roth et al,1983). These works were used as the initial guide for the knowledge acquisition, with later techniques assessed and used if necessary when they became available (Anjewierden, 1987; Slatter, 1987; Greenwell, 1988; Schreiber et al, 1988; Breuker & Wielinga, 1987; Breuker & Wielinga, 1988).

Hart (1986) describes fact-finding by interview and machine induction as methods of knowledge elicitation. Machine induction is best suited to subjects, or *domains*, where large numbers of examples can be given which have different values for the same variables. The induction process builds rules using the examples. The pipeline routing domain lacks this body of examples and so machine induction would not be useful. Interview techniques were thus the main approach used in the project.

It is pertinent to note that Hart (1986) fails to mention the use of documented case studies, industrial standards or codes of practice as part of the knowledge acquisition process. Other authors distinguish between knowledge *acquisition* and knowledge *elicitation* in that the second exclusively deals with the extraction of knowledge from an expert, whereas the first allows other sources such as codes of practice to fall within it's remit (Greenwell,1988). This thesis will uphold the latter terminology.

Interviews can be divided into three types corresponding to levels of detail in the domain being discussed. These types are the unstructured, focussed and structured interviews. They are held between the person who is providing the knowledge, known as the *domain expert*, and the person who is trying to extract the knowledge and build it into an understandable structure, known as the *knowledge engineer*.

Unstructured interviews are the first to occur in knowledge elicitation, when the knowledge engineer has little or no understanding of the domain. The main objectives are to build a rapport with the expert, an overall appreciation of the domain, its main components and its vocabulary. This can be particularly testing for the knowledge engineer, who may have little experience of the domain under consideration.

Focussed interviews, as the name implies, focus on particular aspects of the domain. The technique is to build on the overall domain structure elicited in the unstructured interviews. The knowledge engineer uses individual areas of the domain as subjects for further exploration. Depth in the knowledge base is thus built.

Structured interviews are the most detailed of all, concentrating on particularly complex parts of the domain, reviewing knowledge elicited to date, and verifying the validity of the knowledge in a variety of test situations. Structured interviews can include demonstrations of prototype systems which the expert is invited to criticise. They may also include detailed or unusual case studies and the way the expert handles exceptions to his normal rules. Detailed knowledge on specific parts of the domain is usually discovered and verified in structured interviews.

Hart (1986) identifies the qualities a knowledge engineer should possess to ease interview problems. These include good communication skills, intelligence, tact and diplomacy, empathy and patience, persistence, logicality, versatility and inventiveness, self confidence, domain knowledge and programming knowledge. One wonders whether these mythical creatures exist, but Hart's general message is clear. The knowledge engineer must be able to accommodate the whims of the expert whilst still achieving the results he needs. A manner likely to elicit the friendship of the expert is probably the most powerful tool a knowledge engineer can have.

Each interview type is illustrated in the knowledge elicitation below. Within the interviews themselves a number of questioning techniques were used to control the direction, type and depth of the knowledge being given by the expert. Although used naturally by the author in his own use of the English language, many of these questioning methods have since been described by Greenwell (1988).

5.5 KNOWLEDGE ACQUISITION FOR THE PIRATE SYSTEM

Knowledge acquisition for PIRATE centred on the use of elicitation interviews, following the layout given in Figure 5.1. The information gained from the interviews lead to case studies and other documentation which furthered the understanding of the domain. The experts in pipeline route design were provided by J.P. Kenny & Partners (JPK). The interview process began in July, 1988 and extended through to final structured interviews in January, 1990.

5.5.1 Initial Meeting

Once involvement with JPK had been approved, company management assigned a pipeline design engineer to act as the *domain expert* for the project. The initial meeting was designed to 'sell' the project to the expert, acquiring his confidence and building a rapport between him and the author, hereafter referred to as the *knowledge engineer.* His colleagues were also curious and so were accommodated during initial explanations of the project aims. Although it was commented that "this artificial intelligence business" sounded rather far fetched, the expert seemed happy with his continued involvement. A provisional programme of interviews was agreed before the meeting closed. No elicitation was attempted during this session.

5.5.2 Unstructured Interview Programme

Two unstructured interviews were timetabled to take place soon after the initial meeting. The author had no previous experience of knowledge engineering, and so the difficulties of understanding the domain were expected to be compounded by the authors amateur status as a knowledge engineer.

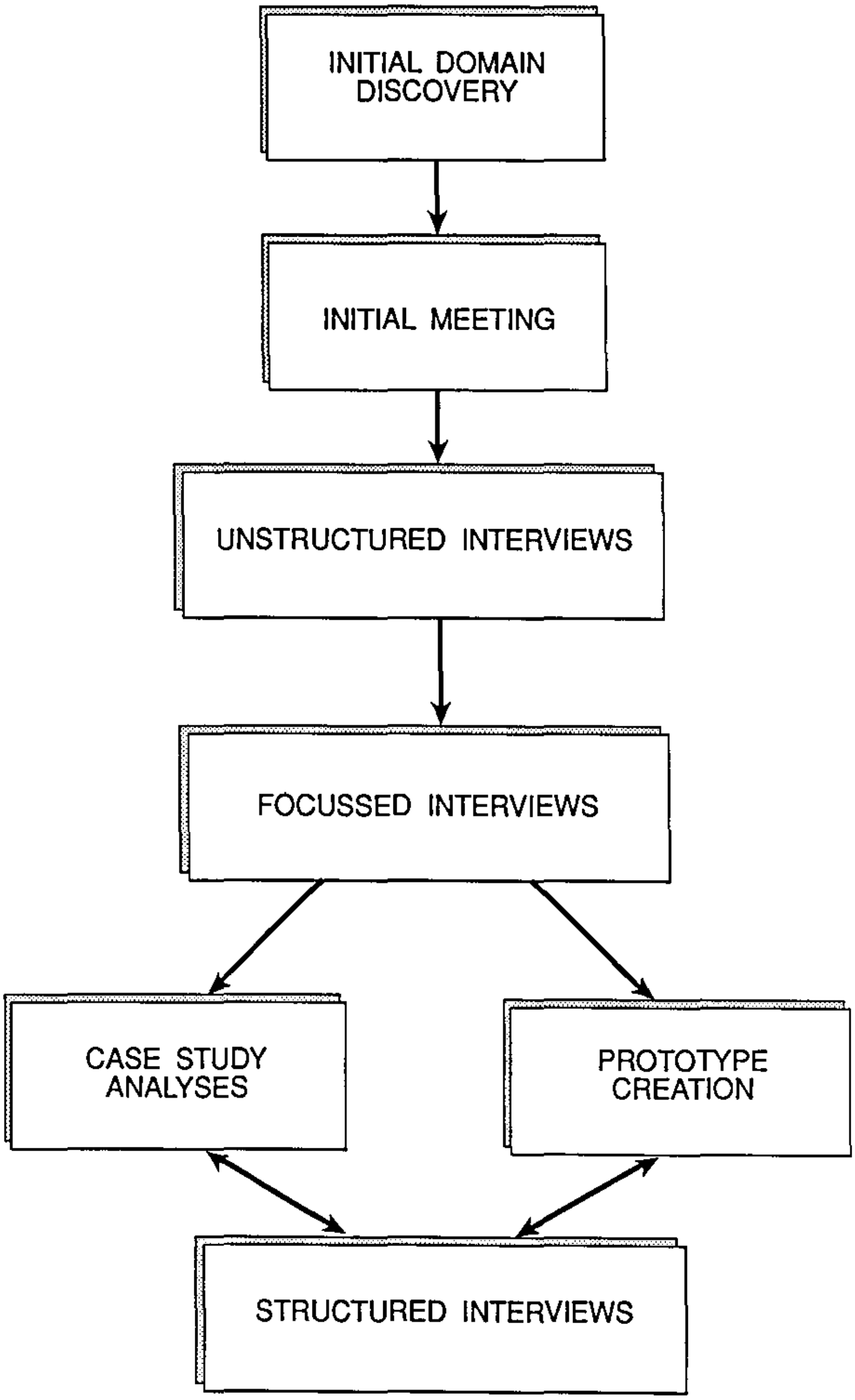


Figure 5.1 THE FLOW OF KNOWLEDGE ELICITATION FOR PIRATE

CD\5786U

5.5.2.1 Techniques Used

It was decided to follow Hart's (1986) advice for the format of the first unstructured interviews, in the absence of any preferred method. She stated the following basic principles :

- Encourage specific case discussions rather than using generalities.
- Do not impose alien tools or representations onto the expert - allow him to express himself naturally.
- Do not interrupt - allow the expert to digress and repeat himself as he pleases.
- Record information - audio recording is recommended
- Listen to the way the expert uses knowledge - it may provide clues to the underlying structure of the domain.

Each interview was split into two hour sessions, two per visit, Both the expert and the knowledge engineer found concentration difficult over such lengthy periods, but available meeting times were few and had to be used as effectively as possible. The interviews were recorded on audio tape.

5.5.2.2 Findings

Although Hart's general principles are presumably based on sound research, when applied in the first interview the result was little short of disastrous. With the knowledge engineer effectively muted by the second and third principles, and with the expert encouraged to use specific cases by first, the digression that occurred was extreme. Rather than choosing representative cases depicting typical pipeline routing projects, the expert chose the most obscure he could find, concentrating on small aspects of the domain which he found interesting or on instances that deviated from the norm. As the knowledge engineer knew nothing of normal pipeline routing, information on exceptions or microcosms of the process were extremely hard to place in perspective.

Additionally the knowledge engineer was expected to take note of how the expert was using his knowledge, in the hope that this would give further clues to underlying domain structure. On top of all this he had to appear reasonably intelligent and 'on the ball' when directing the interview and making comments on the expert's discussion. The author found this an extremely taxing experience, leaving the interview totally bewildered. Appendix A gives a transcript of one of the unstructured interviews, which lucidly illustrates the point.

The approach to the second unstructured interview was modified in the light of experience. The expert was asked in advance to find *typical* pipeline routing case studies on which to base his explanation, and to avoid detailed subjects representing only a small part of the overall domain. He accepted this once he realised that he could concentrate on more detailed knowledge in later sessions.

The result of the second interview was an approximate work flow that pipeline engineers followed in typical situations. This is shown in Figure 5.2., and provided the basis for focussing later interviews.

5.5.2.3 Conclusions

The first interview showed a total mismatch between the expectations of the knowledge engineer and the understanding of the domain expert. The expert simply took for granted much of the domain, which he considered to be common knowledge simply because it seemed so obvious to him. The situation was not helped by the Hart principle of not constraining the interview subject.

In hindsight the author would suggest a far more structured approach. The expert should be asked to prepare a talk on his area of expertise. By specifying a time limit and an audience the scope, language and content can be constrained. A fifteen minute speech to a group of sixteen year olds would be a suitable example. Instead of belittling the domain this is actually forcing the expert to isolate the salient features of his expertise, expressing them in a form devoid of unnecessary jargon and irrelevances. By studying the form of the talk the knowledge engineer may gain an understanding of how the expert views the domain structure, without having to struggle to control the elicitation.

A talk of a similar standard should also be prepared by the knowledge engineer, covering AI, expert systems and methods of knowledge acquisition. Chung & Kamur (1987) maintain that "trying to convince the expert about AI" should be avoided, but it is apparent that their conviction is based on experiences when they did not pre-plan the nature of their approach. The author believes that, if suitably structured, a talk will educate the expert in the basic concepts of the subject, allowing him to have his own mental picture of the progress being made and what is expected of him. The expert would then be more likely to realise the fundamental nature of the knowledge required at the first interview stage, avoiding the problems exhibited above. It would also help to negate the lack of understanding highlighted by the quote on "this artificial intelligence business" at the initial meeting.

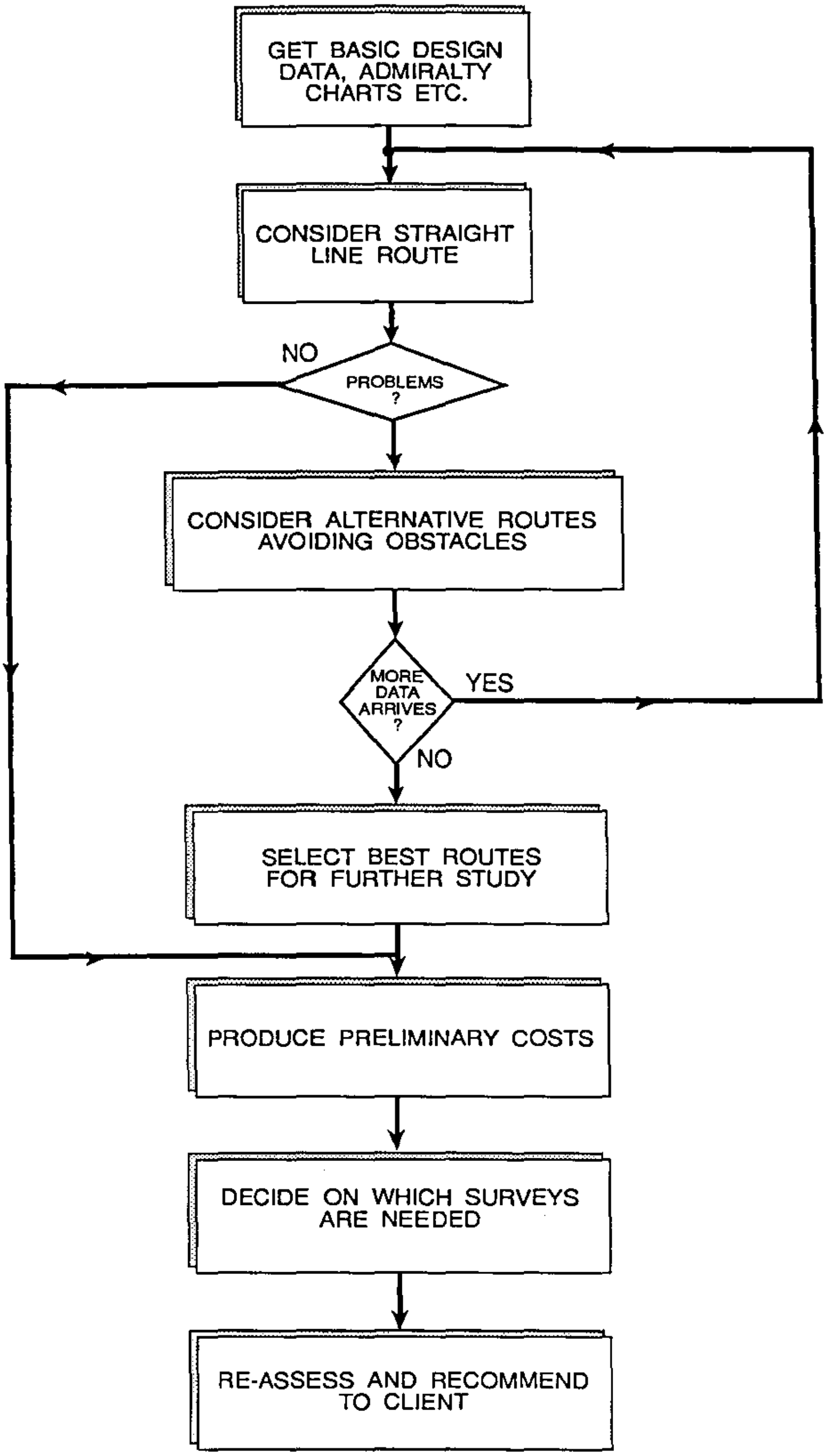

GET BASIC DESIGN DATA, ADMIRALTY CHARTS ETC.
CONSIDER STRAIGHT LINE ROUTE
NO
PROBLEMS ?
CONSIDER ALTERNATIVE ROUTES AVOIDING OBSTACLES
MORE DATA ARRIVES ?
YES
NO
SELECT BEST ROUTES FOR FURTHER STUDY
PRODUCE PRELIMINARY COSTS
DECIDE ON WHICH SURVEYS ARE NEEDED
RE-ASSESS AND RECOMMEND TO CLIENT

Figure 5.2 APPROXIMATE WORK FLOW FOR PIPELINE DESIGN: AFTER UNSTRUCTURED INTERVIEWS

CD\5786V

The results that were obtained from the interviews were not encouraging simply because more progress had been expected. The work flow that was found at least provided focal points for the next programme of interviews.

Audio tape recording of the interviews were not without drawbacks either. Transcription of four hours of tape took several days, and while a much more thorough analysis of the information was possible, there were problems. In particular, difficulties occurred when the expert made reference to a spatial location on a map or diagram he was using in his explanation. The knowledge engineer had anticipated this and verbally referenced the map number whenever it was used. However, when the expert used such phrases as "From here to the jack up rig there are .....", there is no way of placing the comment into context. Appendix A gives many other examples of such comments. Clearly a more reliable method had to be found.

5.5.3 Focussed Interview Programme

Between the unstructured and focussed interviews there was a gap of about four months, allowing for an assessment of the lessons to be learned from initial experiences, and to undertake further reading of texts related to the domain in general (Hydrocarbons(GB), 1986; Det Norske Veritas, 1974). A total of four focussed interviews were planned.

5.5.3.1 Techniques Used

Learning from the experiences of earlier sessions, the expert was fully briefed, in writing, prior to each interview. This allowed him enough time to prepare any case material and make comments. The four interviews each had one of the following focal themes :

- Data availability, significance and preparation
- Creating a route, major factors affecting the choice
- Analysis of chosen routes, accuracy of estimates, cost, plant and material requirements.
- Presentation of route choice and what happens afterwards.

Case studies were used extensively to allow the expert to express his knowledge. At this stage in the interviews basic case studies were more acceptable, as the knowledge engineer had more of an overall understanding of the domain structure. At times it also helped if a hypothetical route were proposed by the knowledge engineer across real sub-sea data, allowing the expert to criticise the choice. Judicial use of this technique allowed particular avenues of the pipeline problem to be explored. The interviews were once again recorded on audio tape.

5.5.3.2 Findings

By focussing the interviews into four themes a more detailed assessment of each could be undertaken, and this successfully pulled out substantive information on the routing process.

Data availability in pipeline design was found to be erratic, disturbing the sequential nature of data gathering then routing. The first available data tended to be the most unreliable, such as admiralty charts of the region. More accurate and detailed data is usually the result of other construction work in the area and is often privately owned. Access to this data may not be possible until after the initial route analysis has taken place. The additional information results in a route re-assessment being needed.

Potential routes are decided upon by the engineer using the data he has available to him. This data is pre-prepared in the form of constraint charts and the engineer must thread the pipeline through the constraints to minimise costs, maximise pipeline safety and constructability, and ensure long term stability and maintainability. Figure 8.1 is a typical constraint chart that was used in the case study, Chapter 8. Placing the route involves a high degree of spatial cognition, something which humans excel at when compared to computers. However, the engineer only uses a few major constraint variables in the consideration of a route choice. Once the line of the route is decided a more complete analysis is undertaken to determine its exact implications.

Full route analysis considers all major factors contributing to the overall route cost. The method used for this involves the initial charting of the route across a number of maps depicting different geographic features and constraints. The chainage distance over which the pipeline traverses each feature is used to estimate the impact of the feature on the route cost. These costs are built from estimates of required construction plant and extra materials needed to overcome the effects of a feature. Plant costs depend on the output rate, the mobilisation fee and the severity of the remedial actions for the feature.

A typical cost analysis for a route is given in Appendix D. Combining component costs is not straightforward, as overlapping features may need conflicting or similar remedial actions, and a compromise action must be reached.

Reporting of the route analysis is extensive, with large numbers of charts augmented by volumes of calculation and detailed written assessment. A typical preliminary costing for a route may take up to two weeks. However, even at this stage the base variables can be subject to change. The expert quoted examples where the diameter of the pipeline was altered at a late stage, necessitating a complete cost re-assessment. The case study, Chapter 8, had its routes re-assessed because of the discovery of a new gas field, which affected the implications of each route. During the focussed interviews the work flow was re-assessed in more detail using a specific case as an example. The results are given in Appendix B.

In the discussions the expert highlighted the following time consuming tasks which hampered progress during the design :-

- Initial collation, storage and comparison of maps, charts and other data concerning ocean floor characteristics in the proposed region.
- Preparation of overlay maps showing features which could affect a pipeline. Several are typically needed.
- Analysis of proposed routes, calculating the distances a route travels through relevant features.
- Subsequent analysis of the effect of each feature or combination of features on the pipeline route.
- Assessing plant and material requirements and hence the cost of crossing each feature, and producing an overall pipeline cost estimate.

It is clear from the above that a suitably designed GIS could assist immensely with the first three tasks. the last two, however, require more expertise than would normally be expected of a standard GIS. It is here that PIRATE would come into its own by using AI in the form of an expert system to make the appropriate decisions.

Audio tape analysis was still proving to be problematic, and it was decided to experiment with video recording before the next interview programme.

5.5.3.3 Conclusions

The lack of sequence caused by erratic data gathering in the routing process indicated that a sequential structure for PIRATE would be unduly limiting. Instead it was decided that PIRATE must be a pipeline design *environment*, a place where the pipeline engineer can develop his design without necessarily maintaining a rigid sequence, where he can add data whenever it suits him, and make changes whenever necessary.

A summary list of the features for PIRATE was compiled with the expert, and this was used as a basis of the implementation of the first prototype :

- Input via a CAD medium or digitiser, or the ability to use existing digital data in a rapid and simple manner.
- The storage and manipulation of spatial data of any kind, allowing production of spatial overlays of feature types, and providing facilities for associated non-spatial properties of features to be held.
- A graphic user interface (GUI), to provide map, chart and other spatial data display, with interactive interrogation capabilities.
- Interactive pipeline route design at the GUI
- Calculation of chainages where routes cross features.
- Automated judgement of the remedial actions needed to overcome problems caused by the pipeline crossing features.
- Automated re-evaluation on addition of new data.
- Facilities to store and compare alternative routes.
- Ability to integrate plant and material specifications into the system, to allow the implications of remedial actions to be assessed and costed.
- Production of maps and reports in an understandable form for incorporation into design documentation.
- Ability to allow changes to the parameters of the design, the pipe diameter for example, to see the effect changes have on the evaluation.

With the above specification there was enough information to begin a prototype implementation of PIRATE. The result would effectively be a geographic expert system shell, into which design rules can be placed. These rules should result from detailed case study analyses and structured interviews.

Video tape recording experiments were initiated at Loughborough to see whether the spatial referencing problems of audio recording could be overcome. Mock interviews proved that there are significant advantages to be gained. Hart (1986) and Allwood (1989a) both note that video recording can be intrusive, inhibiting the expert. However, the tests indicated that, after the first few minutes, the 'stand-in' expert became so involved with the interview that he was unaffected by the presence of the video camera. Part of this may be due to the camera having been set up on a tripod above and behind the expert, and being unmanned throughout the interview. Apart for the level of intrusion being low, there was the additional advantage using this set-up that the documents referred to could be seen in the correct orientation. Whilst, as expected, detail on charts and diagrams could not be made out, a wealth of contextual information could be inferred from the hand movements of the expert. This, incidentally, further justifies the use of a GUI with a mouse, as its use emulates the designers natural descriptive movements.

5.5.4 Case Study Analyses

At the end of the focussed interviews the expert was asked to search for a representative case study, a project he felt was substantial, yet typical of pipeline route design. It was requested for two reasons. First, it would provide confirmation of the design knowledge and work flows elicited to date. Secondly, the PIRATE system would need testing on a real pipeline project if it was to prove it's worth as a practical design tool, and it was hoped that the case study would provide the base data for this.

In December, 1988, J.P. Kenny & Partners kindly allowed the release of a full set of design documentation and charts for the Sable Island Gas Pipeline project, a difficult and extensive pipeline in complex terrain off the coast of Nova Scotia, Canada. The project is described in substantial detail at the beginning of Chapter 8.

In addition to it's verifying role, the case study directly contributed to the knowledge acquisition effort as the organisation of the documents into chapters and sections illustrated the detailed structure the designers felt was behind the route justification. This allowed the domain to be split further, yielding the major geographic features and other object types, such as categories of dredging vessel, to be isolated. These object classes are too numerous to be gathered effectively during interview, but once listed from the cases study analyses the expert can criticise them more easily.

An example set of rules that act on each geographic feature type were collated from the case study, the intention being to provoke the expert into further rule definitions during later structured interviews.

This process has shown that case studies provide an invaluable source of information. However, the author would not advocate their immediate introduction into the acquisition process *other than as a general vehicle to aid expert explanations.* Until the domain is appreciated by the knowledge engineer at a general level he is in no position to judge whether the case is typical. An unrecognised bias in any case studied in detail can affect the structuring of the whole expert system, perhaps with irreparable results. It is for this reason that the Sable Island case was only studied once the focussed interview programme had been completed.

5.5.5 Structured Interview Programme

The structured interview programme began in October, 1989. This was after the first version of the PIRATE prototype had been completed. A total of three days were spent at interview, concentrating initially on a review of knowledge elicited and a critique of the prototype, before continuing to explore deep areas of the domain.

5.5.5.1 Techniques Used

The first structured interview began with a review of knowledge elicited to date. This provided a base from which to continue the work, as it had been many months since the last meeting. The prototype was then introduced to the expert. At this stage most of the major modules existed, including the graphic user interface (GUI) and a GIS capable of accepting spatial feature data and attaching associated non-spatial properties. The pipe design ability was present, though only in a restricted form, and extensive interrogation facilities were available from the menus of the GUI. After a thorough explanation the expert was allowed to try the system and make comments.

The second interview concentrated on the Sable Island case study, and the information extracted from it by the knowledge engineer. The domain object list was assessed, and the rules collated from the study were put to the expert for comment on their truth and scope beyond the Sable Island region. The rule and object lists effectively constituted a *paper model* of domain expertise (Allwood, 1989a). The final rule list is given in Appendix C in abbreviated natural English. From this list the domain objects are also apparent. Hypothetical route analysis was found to be a useful technique during this interview to help judge the applicability of rules in different potential situations.

The final interview was used to verify information gleaned from transcripts of the earlier interviews, and to once again assess the object and rule lists. Each object and rule was formally agreed or rejected before the interview ended. Also the modified PIRATE prototype was shown, which had been changed to accommodate the modifications requested by the expert. The prototype included a small example knowledge base to illustrate the function of the 'intelligent' parts of the system.

Interviews were recorded on videotape, using the same positional arrangements as in the trials, Section 5.5.3.3.

5.5.5.2 Findings

The critique of the prototype yielded a number of pointers to revisions needed in the software. In particular, the expert noted the following additional needs :-

- A digital terrain model (DTM) for interrogation, sectional profiles and the analysis of sea bed slope characteristics.
- Plant and materials specifications in an easily updatable form.
- A locational grid over the map representation on the GUI.

Rule and object list reviews eventually led to an agreed paper model. The consensus between the expert and the knowledge engineer was that in the time available the whole domain could not be elicited, but that the information in the model was a representative sub-set. It is reasonable to assume that further elicitation would yield more detailed knowledge, but the expert agreed that no major areas of pipeline routing in PIRATE had been overlooked. The completed paper model is given in Appendices B and C.

As expected from the trials, video recording did not significantly affect the performance of the expert. It was extremely useful during interview analysis, particularly when discussing the case study and hypothetical route locations. The video was tripod mounted without an operator, so a space had to be marked on the interview desk which was wholly within the camera's field of view. As long as this operating space was used the interview could proceed as normal. The resolution of the image was such that major features on the diagrams could be seen, and the experts hand movements gave indispensable information on the tape.

5.5.5.3 Conclusions

The PIRATE prototype received a positive response from the domain expert. He said that he had imagined an interface that was less interactive, and praised the mouse and menu type operation as well as the geographic display and manipulation facilities. The changes he wished to see were built into the system before the project finished.

The paper model of elicited knowledge was useful as a planning and referencing tool during the structured interviews, and enabled specific agreements to be reached on specific items rather than more vague generalities. The final agreement of the complete paper model heralded the end of knowledge elicitation for the project.

Video recording for the structured interviews was found to be a major asset. A summary list of advantages are:

- Spatial referencing during elicitation is easily discernible
- The video provides a complete image of the interview from an observers stand-point, it is less tedious to transcribe.
- It is far easier to follow which documents being referred to at any time.
- An on-screen date/time stamp allows easy scanning and replay
- Body language, an important indicator of emphasis, is fully recorded.
- The 'shelf life' of the recording before analysis is greater as the knowledge engineer has to recall less about the interview from memory. Visual images are also a more powerful 'memory jogger' for the knowledge engineer than the equivalent audio recording alone.

5.6 SUMMARY

This chapter has documented the method by which the PIRATE system requirements were formulated, and the resulting points that make up the system specification.

Interviews with a practicing pipeline engineer revealed a number of time consuming tasks in pipeline route design. These included data collection, the production of overlay maps, assessing proposed routes for feature crossing, and deciding the effects and costs relating to route position. A specification for PIRATE was produced, such that it would ease the burden of these tasks. This system specification is summarised in the bulleted lists of Sections 5.5.3.3 and 5.5.5.2.

In addition to the system specification, a detailed study was carried out of the knowledge the engineer uses when he makes pipeline design judgements. This knowledge is given in Appendices B and C.

The overriding reason for carrying out the work in this Chapter was to arrive at a PIRATE system specification, the method by which it was achieved was not, at the time, the focus of the effort. The author was simply looking for the most effective way of getting the results he needed. However, the methods did yield a number of *secondary* conclusions, which are of interest in the context of knowledge acquisition as a discipline. These in the main refer to the methods used for controlling and recording interviews with a domain expert. They include the following points, which were more fully described in the Chapter :

- *Pre-planned talks* by both the expert and the knowledge engineer in their respective disciplines is felt to be an important precursor to the interviews themselves. In this way each can gain an idea of the others aims and function.
- *Paper models* of work flow and knowledge elicited make an ideal focus for future discussion and interview structuring.
- *Video recording* excels as a form of data capture during interviews where the expert refers to maps, charts and other documents.
- *Case studies* may be used by the expert to help his explanation, but should be treated with caution until enough of the domain structure is known to judge if the cases are typical or not.
- *Prototype systems* are valuable way to attract expert criticism and views. Timing of the introduction should be considered, however. The prototype must be advanced enough to be impressive, but still capable of being changed radically should the expert wish it.

It was found that knowledge acquisition methods were suitable for discovering not only the judgemental knowledge that the engineer uses during pipeline design, but also for itemising the design work flow and PIRATE system requirements.

CHAPTER 6

6. SOFTWARE TOOLS FOR BUILDING PIRATE

6.1 INTRODUCTION

Earlier Chapters have outlined the theoretical basis and system specification for PIRATE, a practical intelligent geographic information system for design. The implementation of such a system, using commercially available hardware and software tools, was expected to push those tools to the limit of their applicability. A failure in any of the tools used could propagate to the failure of the whole project. Naturally, suitable tools must be extremely carefully selected.

The two major components of an intelligent GIS are the GIS itself and the AI or expert system component. Each are considered separately in the following sections.

6.2 A GIS FOR PIRATE

To allow an intelligent system to reason effectively with geographic information it must be able to access and control any part of the information. Conventional AI tools and GIS are discrete systems, and an interface would be needed to allow one to control the other. Demonstrations of existing commercial GIS revealed that in general they were tailored to interfacing with a human operator rather than another system.

The concept of having an expert system emulating the human operator, giving commands in the same way, was mooted. However, physically interfacing the systems would undoubtedly give problems using currently available technology. Getting the two systems running together in the same machine, and then attempting to get them 'talking' to each other appeared to offer a path fraught with difficulty.

What such a dual system could actually achieve is also open to question. In design an engineer uses a map to access information at two levels, *globally* for an overall appreciation of layout, and *locally* to pick up specific detail. The philosophy behind PIRATE still requires the user to obtain the global appreciation, but PIRATE itself is intended to perform the reasoning which requires local detail. Rules may typically ask for individual items of data from the GIS to prove their consequents. A large knowledge base may result in vast numbers of individual GIS database enquiries being needed.

The external interface approach between two separate systems was not thought to be appropriate for three reasons. First, an external interface would probably be slow, and if large numbers of rules request information the system would be continually swapping control between the two modules, and would no longer be interactive. Second, many commercial GIS do not allow direct access to individual spatial data items in the database, a limitation on the flexibility of the implementation. Third, the designer ideally needs to work at the graphics screen with the map displayed at all times. Writing an interface to allow the designer to work with the map whilst the expert system is advising him in the background appeared to be virtually impossible. Commercial GIS tend to require the exclusive control of the screen and processor when in use.

The problems inherent in using a commercial GIS product were compared with the difficulties of producing bespoke software to fulfil the GIS function for PIRATE. This latter option would give much greater flexibility during PIRATE system design, and would allow a transparent interface between the GIS and the expert system if structured correctly. However, the writing of a GIS is not a trivial task. A bespoke system would need careful design and implementation if it was to operate effectively with the vast quantities of data expected from practical pipeline projects. It would also open the author to accusations of 're-inventing the wheel'.

It was decided that using a commercial GIS would limit the PIRATE system design to the extent that the whole project might fail. A bespoke GIS was therefore needed.

To minimise the amount of unnecessary program coding, a relational database implementation language called Nantucket Clipper was chosen to encode the main GIS structure and functions (Nantucket, 1988). This provided a range of functions which could operate directly on relational tables. Conventional programming languages would not support such functions unless the programmer wrote them himself.

Another advantage of Clipper was that it's relational tables that were compatible with the dBase industry standard (Ashton Tate, 1986). Many other systems can interface with this form of table, including the AI toolkit eventually chosen for the implementation, Section 6.3.5. This common interface would allow the AI toolkit to access the GIS tables directly, and the GIS functions to communicate with the AI system by creating tables which the AI system could read. Finally, Clipper was fast. In initial tests comparing the creation of a 500,000 record relational table, Clipper took seven minutes whilst the dBase interpreter took fifteen *hours*. This was not considered a comprehensive benchmark, but was an indication that Clipper did have advantages over it's major rival.

6.3 AI TOOLS FOR PIRATE

Tools for knowledge base system implementation fall into three broad categories, AI languages, expert system shells and AI toolkits. Systems from each category were considered for the implementation of the PIRATE prototype.

AI languages are the least specialised and most flexible of the category types. They are essentially programming languages, but specialise in the manipulation of symbols rather than numbers. Two of the most popular languages are Lisp and PROLOG (Winston & Horn,1984 Clocksin & Mellish,1987).

Expert system shells are the most easy to use yet restrictive of the categories. They allow the user to concentrate on entering the domain knowledge, leaving the interfaces and inference engine to be provided by the shell. The simplicity of expert system design using a shell means that prototypes can be developed rapidly by relatively inexperienced computer users. However, many shells have inference engines that can only perform certain types of inferencing, backward chaining for example. Also the domain knowledge must be moulded into the knowledge base format used by the shell, which may not be suitable for the type of information being stored. The interfaces provided with the shell may also prove unsuitable or restrictive for the application.

AI toolkits attempt to provide the flexibility of the AI language and the ease of use of the expert system shell. Usually based on a version of an AI language, toolkits invariably have a library of inference strategies, knowledge representations and interface facilities. Rather than restricting the designer the toolkit principle allows him a choice of a wide variety of modules, giving the opportunity to select which are best for his particular application. The modules can often be combined with AI language code if additional functions are needed. AI toolkits are expensive and tend to require high capacity, high performance hardware. However, in cases needing flexibility combined with speed of development they do provide an ideal prototyping environment.

Which tool to choose for a project is a complex decision which will significantly affect the success or failure of the development. Often, however, the project cannot be defined clearly enough prior to prototyping to ensure that the tool chosen is the most appropriate.

For PIRATE, tools from each of the three categories were tested for suitability. The criteria for the comparison rested largely on preliminary knowledge elicitation with pipeline experts, Chapter 5. Key criteria included access to interactive graphics, interfaces to databases and flexibility in inference strategies and knowledge representation. Clearly the dangers of limiting development by the choice of an over restrictive tool had to be avoided as at this stage the PIRATE specification was itself only vaguely defined.

Initial tests at Loughborough were augmented by using the facilities of the Artificial Intelligence Applications Institute (AIAI) in Edinburgh. AIAI is considered to be the leading AI institute in the United Kingdom, and it offers a wide range of courses and research facilities to both industry and academia. At AIAI an introductory course was taken to gain an overview of the tools available (AIAI, 1987c), and this was followed by hands on experience with each tool.

Five systems were chosen for further investigation based on initial findings, and each is given a section below.

6.3.1 VP-Expert

VP-Expert is a simple backward chaining expert system shell written for MS-DOS machines by Paperback Software (1987). It was chosen for further investigation because of its reasonable price (£100) and the flexible interfaces it offers to databases, spreadsheets and external programs. It was purchased early in the project to give the author a means to experiment with expert systems and their capabilities.

A prototype automatic routing system was built using VP-Expert to test whether such a simple shell could cope with choosing routes in a geographic context. A dBase database was used as the basis for a small quadtree encoded grid of geographic cells. Each cell was assigned a simple weighting factor, and the task of system was to optimise the route between two points.

The prototype took three months to develop, but eventually failed due to limitations in the VP-Expert shell. Although interfaces to the databases worked perfectly, backward chaining hit an apparently unavoidable limit when an individual chain reached 17 rules in depth. The system could not function within this restriction and so the prototype was abandoned. A full report on the development is given by the author (Finniear, 1987d).

There was always some doubt as to whether VP-Expert could be made suitable for the PIRATE implementation. However, the prototype highlighted where the shell was inadequate and this was invaluable experience when assessing other AI tools for suitability as a PIRATE host system.

6.3.2 Inference ART

Inference ART is a comprehensive AI toolkit introduced to the author at the AIAI introductory course. A further two day course on ART was taken at AIAI to look in more depth at the system capabilities (AIAI, 1987a). Although the toolkit provided all common AI paradigms, it had no interfaces to external databases. The possibility of creating a database within the working memory of ART or through a Lisp interface to physical storage was investigated. However, the effort required to achieve a working solution would have been too great. Also ART only works on UNIX workstations or specialised Lisp machines, which are expensive.

The lack of database access and high cost of hardware and software lead to the rejection of ART as an environment for PIRATE.

6.3.3 AutoCAD and AutoLISP

AutoCAD is the most popular CAD system in the UK (AutoDesk, 1988). It was investigated as a possible tool for PIRATE development because its macro language AutoLISP is a derivative of Lisp (Head, 1987). As such it could perhaps offer an AI capability together with full AutoCAD graphics integration. This would avoid the problems of graphical integration common in other environments.

AutoLisp shares with Lisp the disadvantage of not having a pre-defined inference engine, rule or object forms. It was felt, however, that the inherent graphic capability could offset the development effort needed to build the additional AI structures needed for PIRATE requirements.

Investigations into the power of AutoLisp as an AI development environment were undertaken. Crucial to any inference engine is the recursive capability of the implementation language, as this can place a depth limit on search through a knowledge base. The *stack space*, allocated by the language in computer memory, stores intermediate stages of recursion and is the key to recursive capability. AutoLisp documentation failed to give a figure for the stack space so AutoDesk, the publishers of AutoCAD, were contacted directly. They stated that although AutoCAD release 10 supported the use of extended memory, it could not support full recursion as it only allowed a 25kByte stack space. Experience with recursive limitations in the VP-Expert prototype, which caused total system failure, together with the lack of in-built AI paradigms lead to the rejection of AutoCAD.

It should be noted here that other expert systems have used AutoCAD successfully as a graphics front end. The BERT expert system for brickwork design is a notable example (Bowen et al,1986). However, BERT translates all the graphics in AutoCAD into a text based description before transferring this to the expert system for analysis. The MUFL expert system language used by BERT is not dependent on the AutoLisp stack space, and so problems of recursion limits do not arise. The BERT methodology relies on the graphics input by the user being composed of simple primitives which can be translated into text. Geographic information in PIRATE could not be translated in this way, so the method could not be adopted.

6.3.4 IntelliCorp KEE

KEE is an AI toolkit based on UNIX workstation platforms, offering rule based, frame based and object oriented paradigms (Sobel, 1988, IntelliCorp, 1987a). Based in Common Lisp it also allows the flexibility of incorporating user defined Lisp code into an application. A two day course on KEE was taken at AIAI (1987b), and this was followed by a three week trial using KEE as a development tool. The main aims of this trial were to gain an appreciation of how KEE worked in practice, and to discover whether the facilities provided would offer enough flexibility for the PIRATE development. It must be remembered that at this stage the structure of PIRATE had not been clarified in detail, so flexibility to alter the PIRATE structure within the chosen toolkit was important.

As a result of these trials KEE was put forward as a possible PIRATE development environment. In short the reasons for the recommendation were that it :-

- Provides all the major AI paradigms in one unified environment
- Has a Lisp language base allowing the programmer to build any additional functionality he needs in the system
- Has a graphic image capability for the representation of objects in the system (with the possibility of building an interactive graphic interface for pipeline design).
- Provides extensive debugging and knowledge base display facilities, specifically designed to allow rapid knowledge base development and testing.
- Can access relational databases using SQL by using the KEEConnection interface module (so giving the possibility of direct access to existing GIS written using the relational model).

PIRATE naturally needs a GIS database to provide the core data on which to assess a pipeline design. Negotiations were entered into with Edinburgh University Department of Geography for the use of the Geoview GIS (Sinah & Waugh, 1988). Based on the Oracle relational DBMS it seemed to offer the functionality and low level access needed by PIRATE whilst maintaining compatibility with KEE through the KEEConnection interface (IntelliCorp, 1987b).

Initial dialogue with Edinburgh about Geoview was positive, though further arrangements were cut short as the research budget was too restricted to purchase KEE.

6.3.5 GoldWorks

At the time of the investigation (mid-1988) GoldWorks was available in its first version (GoldHill, 1987). GoldWorks version 1 was a text based AI toolkit, and whilst it provided the multi-paradigm approach characteristic of ART and KEE, it did not have the interactive graphics and debugging facilities which were seen as a necessity for the PIRATE implementation. However, it did run on personal computers under MS-DOS, giving it a significant financial advantage over other toolkits.

Other researchers at Loughborough University had already gained considerable experience using GoldWorks on a number of projects, and they were an important source of practical knowledge on the strengths and limitations of the system (Allwood, 1989b). One project had even included a graphic display facility, developed through the use of FORTRAN and C interfaces. This was used for displaying machine vibration analysis graphs. While this interface was adequate for this application, it still fell short of the interactive requirements envisaged for PIRATE.

Meetings with Artificial Intelligence (AI) Limited, the UK distributors of GoldWorks, were undertaken to see whether any improvements to the software were scheduled which would make it more suited to PIRATE. AI Limited stated that GoldWorks version 2 was due for release, and that this version would run under the Microsoft Windows graphics environment on MS-DOS machines. The intention, according to AI Ltd., was to offer facilities of the same calibre as ART and KEE, but on the less expensive MS-DOS platform. They noted that interfaces to the Oracle DBMS were not envisaged, however, and so the use of the Geoview GIS would not be a possibility.

An arrangement was negotiated with AI Ltd. in which GoldWorks version 1 would be provided until GoldWorksII became available, whereupon an upgrade would be made.

Likely difficulties in taking on GoldWorks were apparent when negotiations were underway. GoldWorksII was an un-released and unproven package, likely to have many bugs and teething troubles which would aggravate the PIRATE development process. The release date was also vague.

Despite these difficulties PIRATE was judged to need the facilities of an AI toolkit. Without the financial support needed for KEE, the only plausible choice for specification and price was GoldWorks.

6.4 SUMMARY

The choice of a software and hardware to host an implementation of PIRATE has been made. The AI and GIS components were considered separately although the links between them were taken into account.

The GIS choice hinged on the decision either to use an existing commercial GIS package, or to write a bespoke GIS. Commercial packages appeared to offer no realistic possibility of integration with an AI system at the level required. The lack of flexibility in commercial packages, bourne out by difficulties experienced by other researchers, was also thought to be a possible problem. Therefore, reluctantly because of the amount of extra software writing required, it was decided to write a bespoke GIS database and manipulating functions for PIRATE.

The AI component was also a difficult choice. A wide range of systems were put on extensive trial, from simple expert system shells to advanced AI toolkits. The choice was eventually GoldWorksII, an AI toolkit that could run on an MS-DOS based personal computer with a minimum of 10 Megabytes of main memory. In addition to promising the required flexibility for knowledge representation, interfaces to relational tables and graphics facilities, it also fell within the bounds of the finances available for the research.

CHAPTER 7

7. A TECHNICAL DESCRIPTION OF PIRATE

7.1 INTRODUCTION

This chapter gives a full technical description of the Pipeline Route Analysis and Testing Environment (PIRATE). The system combines artificial intelligence (AI) techniques with a geographic information system (GIS) to assess whether an intelligent GIS could indeed make a practical, worthwhile contribution to design. The system was written by the author and has been applied to off-shore pipeline route design.

In a series of interviews with practicing pipeline engineers, Chapter 5, it was revealed that a computer system was needed not only to help with data collation and analysis, but also to automate the design process itself. Ideally the engineer should be able to sit at a computer screen and have immediate access to all his project data, geographic or otherwise. By simply sketching a route onto an on-screen map, the engineers wished to be presented with a full route analysis, incorporating remedial actions, plant and material requirements and preliminary costings. It was hoped that using such a system routes could be optimised in hours rather than weeks. A detailed list of system requirements can be found in Sections 5.5.3.3 and 5.5.5.2.

It was decided at an early stage that PIRATE would *not* attempt fully automatic pipeline routing without the intervention of the engineer. Engineers are inherently suspicious of 'black boxes', and prefer to be aware and in control of the routing process. Also, humans are far better at assimilating spatial pattern than computers. The theoretical basis for fully automated design using unconstrained geographic information is still unclear, Section 4.3. It is therefore sensible and desirable to give the engineer all the spatial and related information he needs quickly and allow him to place a number of routes. The time consuming part of the design, according to the engineers, is not route placement but the initial data collation and subsequent analysis.

In addition to the obvious advantage of speeding up the design process, PIRATE was expected to yield a number of other advantages. First, by encapsulating the knowledge of pipeline design engineers it may allow less experienced staff to perform pipeline design. Second, the application of the same knowledge to each route ensures that any route comparison is completely free of bias. Third, PIRATE can provide a complete and up-to-date information source, a *management information system* which can be interrogated throughout the life of the project.

7.2 THE PIRATE SYSTEM STRUCTURE

The close integration of GIS and AI is the key to providing the functionality needed for engineering design. During the design process geographic and other project data can change unpredictably at any stage. To achieve its aims the AI system must be sensitive to those changes as they occur, updating its contribution to the design accordingly.

PIRATE attains this integration by using a specially written GIS data model, part of which actually exists within the environment of an *AI toolkit*, called GoldWorksII. The overall structure of PIRATE is illustrated in Figure 7.1. This tangled web is actually rather simple. Essentially PIRATE has two major components, a GIS and GoldWorksII. These are delimited by the dashed boxes in the Figure. The detail within the boxes is explained later, but the overall information flow can be seen.

Essentially geographic information is entered through a CAD interface, whereupon the GIS builds the appropriate description. Non-geographic project information, such as pipeline parameters, construction plant and material specifications, are entered directly or from spreadsheets into GoldWorksII. Output is via the graphic user interface (GUI), which taps straight into both the AI toolkit and the GIS.

Pipeline design takes place at the GUI, which is written within the AI toolkit. The user places pipelines as a connected network over a base map, having used overlay and interrogation facilities to satisfy himself of the spatial layout. The user then chooses a pipeline for analysis, and PIRATE GIS functions assess what geographic features the route crosses, and the crossing distances. This data provides the basis for route assessment by the PIRATE AI component.

With a pipeline route assessed for feature crossing PIRATE has access to all the information a human engineer would have at the same stage of the design. It knows not only *what* features the pipeline crosses and the crossing distance, but all the properties of the features. Within GoldWorksII the *pipeline rule base* adds the final touch, giving PIRATE knowledge about the *significance* of crossing those features.

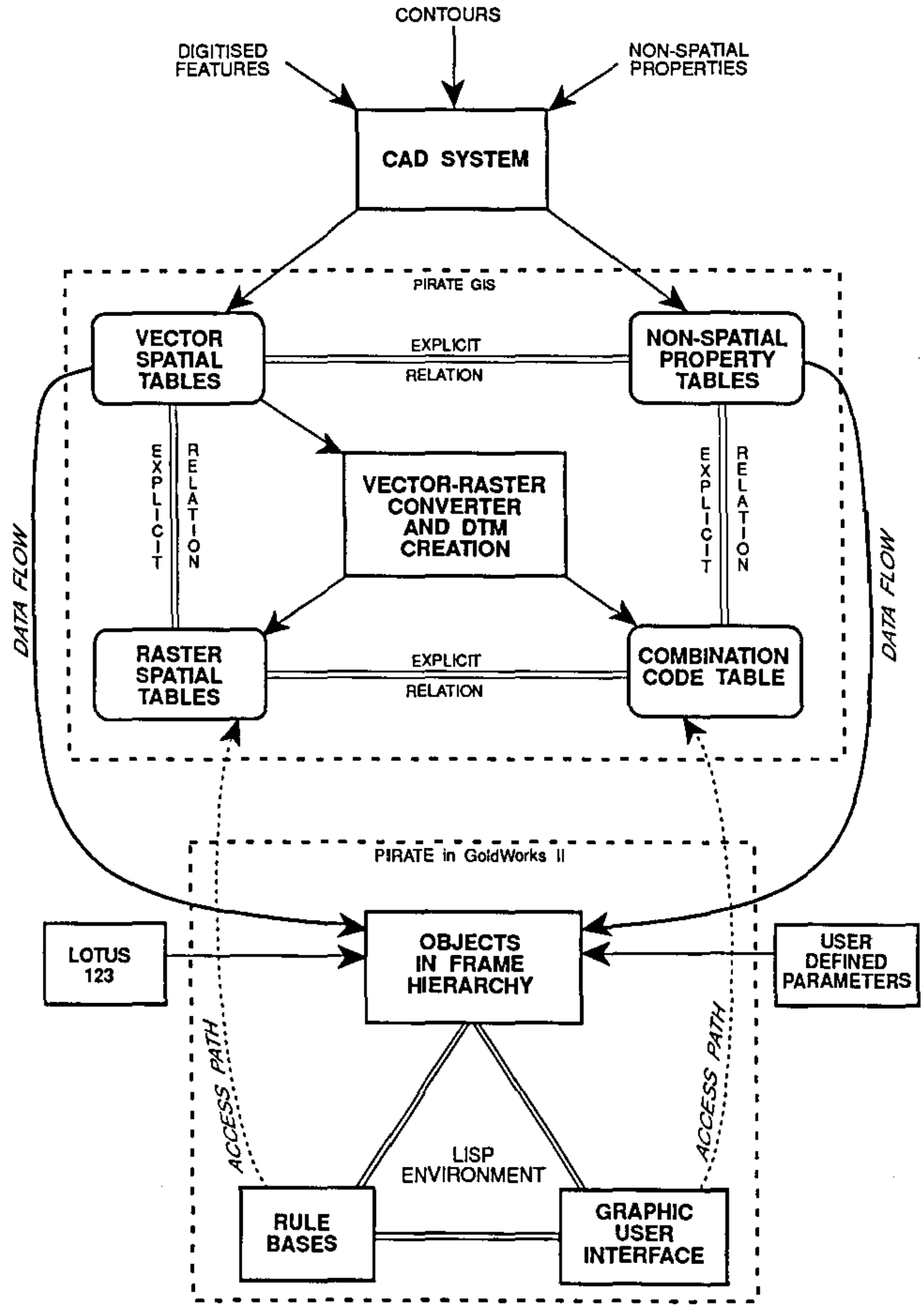


Figure 7.1 A SCHEMATIC OF PIRATE

CD\5786A

PIRATE is built using over 10,000 lines of program code, written by the author. These break down into the following components :

Lisp Code (PIRATE in GoldWorks)	: 7318 lines
Clipper Code (PIRATE GIS)	: 2014 lines
Expert Series Macro Code (CAD interface)	: 1004 lines
TOTAL	: 10,336 lines

Note that of the Lisp code, approximately one quarter deals with the graphic user interface and associated functions.

The following sections take the GIS and AI parts of PIRATE in isolation, describing precisely how they operate and the reasons why they have been designed the way they have. *At all times the reader is asked to bear in mind the position of the components within the overall PIRATE structure, as the depth of the following explanation is such that the overall context can be easily lost.*

7.3 THE PIRATE GEOGRAPHIC INFORMATION SYSTEM

To give a rapid, interactive response at the GUI with potentially vast quantities of sub-sea geographic data, the PIRATE GIS must be *fast.* The implementation hardware, an 80386 PC, would not make up for poor database design.

It was noted in Section 4.3 that GIS queries break down into two basic forms:

What objects (properties) exist at this location ?

Which location(s) has this particular object (property)?

Spatial dualism, having both location based and object based representations in a GIS database, has been shown to be a key factor in rapid access for both types of query, Section 4.3. PIRATE uses spatial dualism as the hub of its GIS data model.

In PIRATE the location based component is provided by both raster and vector data structures implemented as relational tables. The raster structure provides rapid locational indexing, thus allowing instantaneous access to information about any point in space. It is derived from vector information that was originally input. The vector database is maintained for fast graphic display. The GIS also contains a grid based digital terrain model (DTM) for slope and section analyses. The data structures and GIS functions are of the author's design and are detailed in subsequent sections. At this stage of development the PIRATE GIS deals with area features only. Point and linear features are intended as extensions to the vector data structure.

The object based component of the GIS data model is actually built within the AI toolkit. It is this which ensures close AI/GIS integration. Within the AI toolkit each geographic feature is considered as an individual object, and is stored as an instance of a *frame*. The object based component is discussed in Section 7.5.1

7.3.1 Vector Tables

The vector form of the PIRATE GIS consists of two relational tables. Geographic feature boundaries occupy the vector boundary table and contour vectors are placed in a separate contour table. Non-spatial properties of features are stored in a separate series of tables, one for each feature class.

Features are input as closed areas with distinct boundaries. The boundaries themselves are made up of a connected chain of straight line *segments*. To store these the feature table has four fields. ATTRIB_ID stores a unique code for the feature, LINE_ID holds a unique code for each boundary segment, and the X and Y fields hold the end point coordinates of the segment. Figure 7.2 shows the representation graphically. The start of any segment is assumed to be the end point of the previous segment, and it is left to the input software to check that each polygon boundary closes. The CAD system interface is written to ensure this.

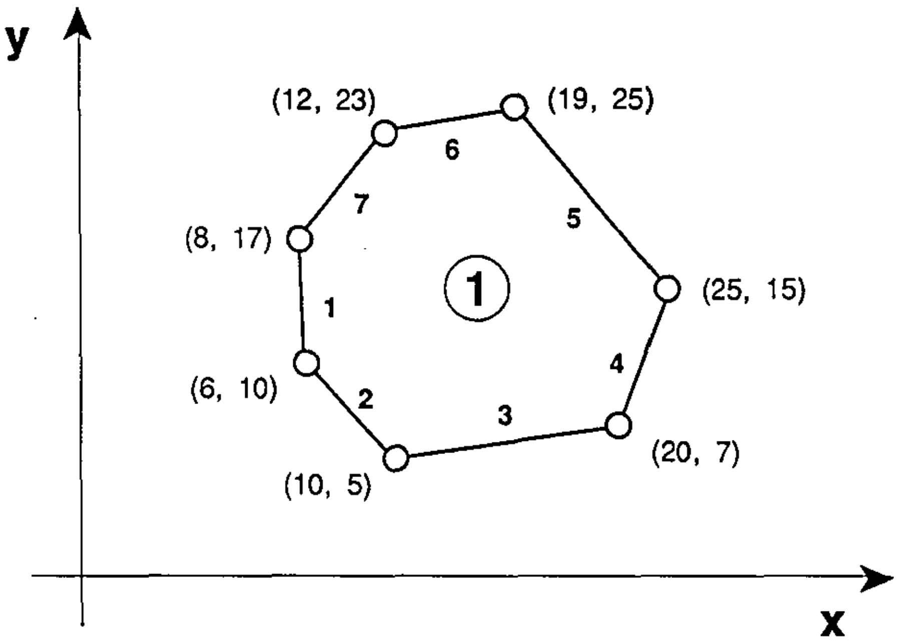
Note that during PIRATE development 'features' were commonly referred to as 'attributes' of the geographic model, hence the ATTRIB_ID *field label in most GIS tables. This apparent inconsistency is merely a product of early research thinking which proved persistent in the program coding. In this context it should be remembered that the terms 'attribute' and 'feature' are synonymous.*

Contours are stored in a similar manner to features. In this case, however, a Z field is also provided to store the elevation of each contour. A full schema of the tables in the GIS is given in Appendix F.

At this stage features in the GIS are no more than a shape in space with a unique code identifier. PIRATE uses the relational architecture to relate these codes to non-spatial data in other tables, such as the boulder field and megaripple tables shown in Figure 7.3. Each feature class has it's own table with fields denoting specific properties relevant to the feature. Each record in these tables represents an individual feature polygon, with the relation to the spatial description maintained through the shared feature code in the ATT_ID field. In the implementation all feature codes consist of two text characters, employing the extended ASCII character set.

7.3.2 Raster Tables

Different forms of raster data structure were discussed in Section 2.3.2. Common to all is the notion of a grid of regular cells covering a region. Each cell has a value assigned to it depending on the value of a certain parameter at that point in the region. Three issues are important in raster database design; the coding of features within a cell, how the cells themselves are represented, and how the data is input and manipulated. A standard raster grid was used for PIRATE rather than a hierarchical structure, for reasons explained later.



x	y	Line_ID	Attrib_ID
8	17	0	1
6	10	1	1
10	5	2	1
20	7	3	1
25	15	4	1
19	25	5	1
12	23	6	1
8	17	7	1
"	"	"	"
"	"	"	"

Figure 7.2 REPRESENTATION OF FEATURES IN THE VECTOR BOUNDARY TABLE

CD/5786B

ATT_ID	DENSITY	ATT_NAME
#1	27	Boulder_field_1
#2	36	Boulder_field_2
#4	10	Boulder_field_3
#9	37	Boulder_field_4
"	"	"
"	"	"

BOULDER FIELD FEATURE PROPERTY TABLE

ATT_ID	WAVE_DIR	WAVE_HT	WAVE_LEN	ATT_NAME
#3	271	2.67	50	Megaripple_1
(-	345	1.2	175	Megaripple_2
(A	27	3.2	75	Megaripple_3
"	"	"	"	"
"	"	"	"	"

MEGARIPPLE FEATURE PROPERTY TABLE

Figure 7.3 EXAMPLES OF FEATURE PROPERTY TABLES

CO/5798C

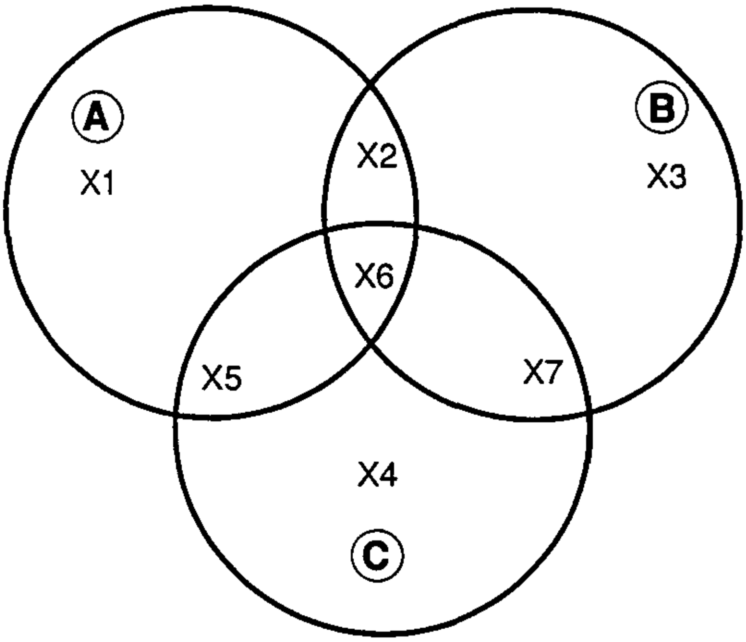
The main raster database in PIRATE is implemented as a relational table where every record represents a grid cell. The position of a record in the table indicates it's cell location in the geographic area. The content of each record is a 2 character (2 byte) text code. To access the code at a particular geographic location the cartesian coordinates of the location are used to calculate a single value representing the position of the cell in the table. This value is known as the pointer, or address, of the cell. A typical raster table is shown in Figure 7.4. Note that in addition to the CELL_CODE field, the table also has an ELEVATION field for holding height values for a grid based DTM.

The two characters of text that are stored in each cell record have a special significance. The code is called a *combination code*, and represents the entire combination of feature polygons that exist at the cell. During the conversion from vector to raster, described in Section 7.3.3, a separate *combination table* is built which can be used to decompose a combination code into any number of individual feature codes.

Figure 7.4 shows three overlapping feature polygons, A, B and C. It also shows a part of the raster table and the combination table. If the features shown were converted to the PIRATE GIS raster form, the feature codes A,B and C would *not* be stored in the raster table. Instead the raster table would hold combination codes, shown in the figure as codes X1 to X7. One code exists for each different combination of features.

Referring to figure 7.4, if a query to the raster table yielded combination code X6, for example, the individual features could be found. First, the code X6 is matched with an entry in the CURR_CODE field of the combination table. This matching is rapid because the field is indexed. The record also gives the latest feature code to be added to the cell, in the ATT_CODE field, and the old combination code that existed before the latest feature was added, in the PREV_CODE field. For code X6, feature code C and previous code X2 are found.

The previous combination code, X2, can then be matched against the CURR_CODE field to find the next oldest feature addition, and a still older combination code. The arrows on the combination code table in Figure 7.4 show how the decomposition continues until all features at a cell are discovered.



RASTER TABLE

CELL_CODE	ELEVATION
"	"
"	"
"	"
X2	0.00
X2	0.00
"	"
X6	0.00
X6	0.00
X6	0.00
X5	0.00
"	"
"	"
"	"

COMBINATION TABLE

CURR_CODE	PREV_CODE	ATT_CODE
X1	"	A → A
X2	X1	B → B
X3	"	B
X4	"	C
X5	X1	C
X6	X2	C → C
X7	X3	C

List of Features at any
Cell with Code X6 is:
(A B C)

Figure 7.4 RASTER AND COMBINATION TABLES SHOWING CODE DECOMPOSITION

CD/5786D

The method of using combination codes allows the variable number of features existing at any point to be accommodated within the fixed field lengths of relational tables. In this manner the technique overcomes some of the problems of variable length cartographic data highlighted in Section 2.4.

The actual representation of the cells in a raster database can either follow a standard grid pattern with all cells the same size, or be hierarchical, with cells formed by the recursive subdivision of the whole region, Section 2.3.3. Quadtrees, a type of hierarchical data structure, were initially considered for the PIRATE GIS. They appeared to offer advantages in minimising database size and providing rapid access speeds. However, they were eventually rejected for the following reasons:-

- Quadtree creation from vector data is not simple as the pointer order follows the convoluted Morton Order curve. This is not suited to linear 'scanning' methods.
- Quadtrees do not fit well into the relational data model as the fixed field lengths inhibit quadtree growth.
- To allow the creation of an object model in the AI toolkit, PIRATE represents each feature polygon separately. Quadtrees are better suited to representing data which is grouped by class, rather than individual polygons which would need a separate quadtree each.
- Pipeline clash analysis, where each straight line segment of a proposed pipeline is 'scanned' to find out when it enters and exits feature polygons, is more complicated with quadtrees because of the Morton Order curve and differing cell sizes.
- A DTM, needed for PIRATE, does not gain any advantage from quadtree encoding. To save space quadtrees amalgamate adjacent cells with the same property value(s). Elevation changes continuously, so little space could be saved.

A standard grid was therefore chosen for the PIRATE raster data structure.

The order of the records in the raster table has to be interpreted spatially so that the position of any record indicates its location in geographic space. The Row Order, Figure 7.5(a), was chosen because it allows straightforward 'scanning' of vector data such that the encoding functions do not have to backtrack up the main raster database. This makes the encoding more efficient. Also it is simple to derive the pointer, or position, of any cell in the raster table from the cartesian coordinates of a point location. The equation is given with a diagrammatic explanation in Figure 7.5(b).

6	12	18	24	30	36
5	11	17	23	29	35
4	10	16	22	28	34
3	9	15	21	27	33
2	8	14	20	26	32
1	7	13	19	25	31

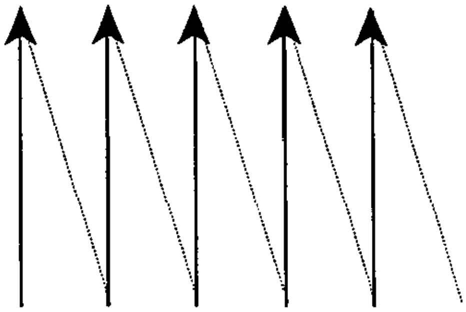
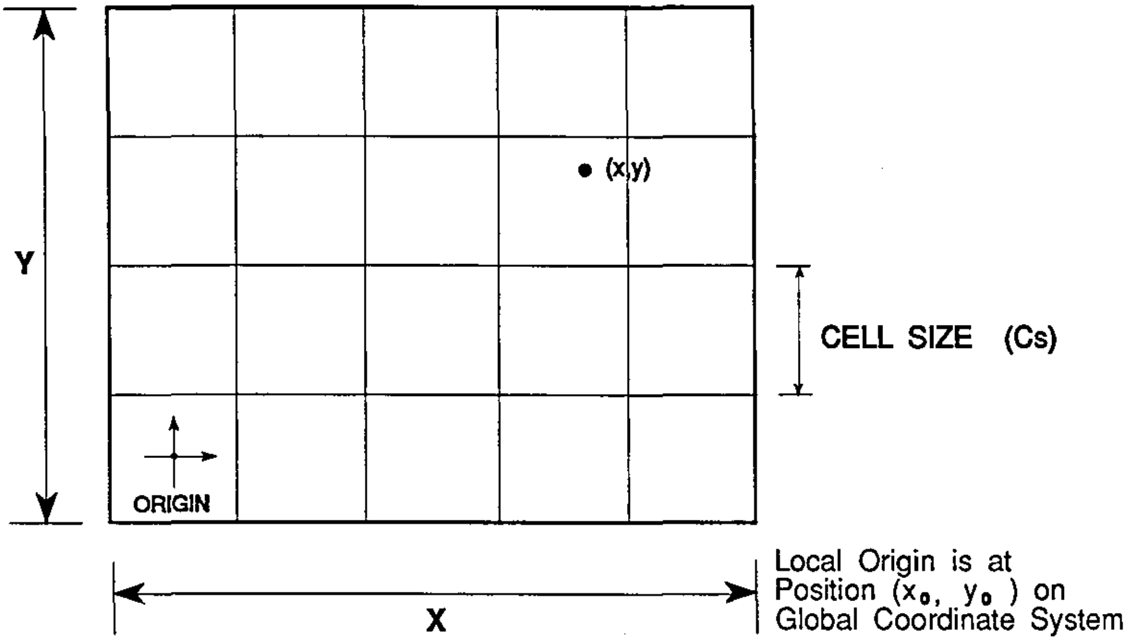


Figure 7.5(a) ROW ORDERING OF POINTERS AND THE CURVE DEPICTING IT



Y
• (x,y)
CELL SIZE (Cs)
ORIGIN
X
Local Origin is at Position (x_0, y_0) on Global Coordinate System

THE POINTER AT POSITION (x, y) IS DEFINED BY:

$P = Y\ (\text{Round}[(x - x_0)\ /Cs])$ ..... Component from Columns to left of Cell

$+\ (\text{Round}\ [(y - y_0)\ /Cs])$ ..... Component from Current Column

$+1$ ..... Correction for Origin being in Cell 1

Figure 7.5(b) EQUATIONS FOR THE DERIVATION OF POINTERS IN THE PIRATE GIS

7.3.3 Vector to Raster Conversion

Raster conversion takes place by isolating each feature polygon in the vector table, then 'scanning' it in row order, Figure 7.6(a). The purpose of scanning is to identify which raster cells lie inside a feature polygon. Each scan line is in the cartesian Y direction along cell centroids, starting from the leftmost part of the feature. When the scan line hits the first feature boundary, subsequent cell centroids are known to be inside the feature and so raster cell recoding is 'turned on'. The recoding generates and inserts new combination codes. Crossing the second boundary turns the recoding off. Subsequent boundaries continue the on/off switching. Closed polygons always have an even number of scan intersections, no matter how convoluted they become. Centroidal sampling is used to centre the raster 'saw tooth' boundary around the original vector boundary, thus minimising errors in the representation, Figure 7.6(b).

In practice the conversion program first splits the main vector polygon database into separate tables, one for each component polygon. For each table the boundaries are taken in turn and all scan line crossing points are found. The resulting table is a long list of coordinates of scanline/boundary intersections for the polygon. This is indexed on the X value field, and then each set with identical X values is taken as a separate table. These tables represent the crossing points for an individual scan line on each individual polygon (because a scan line has a constant X value). Each table is then re-indexed on the Y value, such that an *ordered* list of points along the scan line is produced. Row order pointers are calculated for each of these in turn, and the cells between the pointers are thus classified as inside or outside the polygon. The cells inside have their codes updated to reflect the existence of the new feature.

If a cell is found to be within a new feature the following steps occur. First, the old combination code is checked to see if it has been found already during this polygon scan. If it has a new combination code will have already been generated, and this is inserted into the cell. If not, a new combination code is generated (two characters using ASCII codes 30 to 255). This new code is inserted into the cell and a new record is added to the combination table, giving the new code, the old code and the feature code. The links in the combination code illustrated in Figure 7.4 are thus automatically built in.

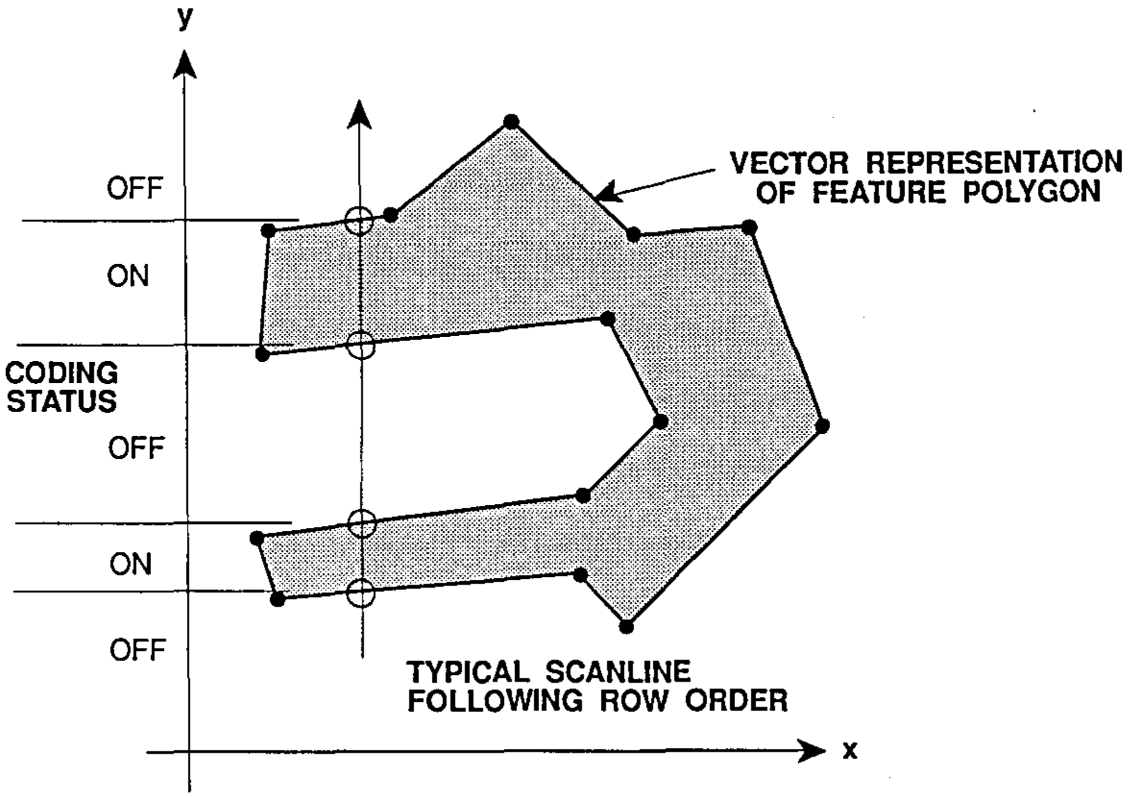

y
VECTOR REPRESENTATION OF FEATURE POLYGON
OFF
ON
CODING STATUS
OFF
ON
OFF
TYPICAL SCANLINE FOLLOWING ROW ORDER
x

Figure 7.6(a) SCANNING FEATURE BOUNDARIES INTO THE RASTER GIS

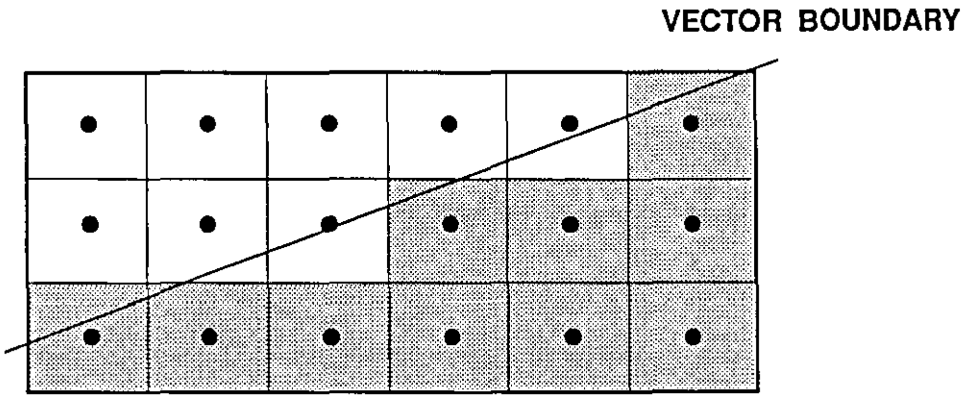


Figure 7.6(b) THE EFFECT OF CENTROIDAL SAMPLING

7.3.4 Terrain Model Generation

A digital terrain model is built into the PIRATE GIS. The DTM is grid based at the same cell resolution as the raster database cells. Storage is an extra field on the main raster table which holds the height at each cell, Figure 7.4.

Creation of the DTM is only possible from digitised contours at present. The same scanning technique is used as with feature encoding, with the scan lines crossing contours instead of feature boundaries. Any two crossing points are noted and linear interpolation is used to find the heights of all cell centroids between them.

7.3.5 Storing and Analysing Pipeline Routes

Any judgement on the suitability of a pipeline route depends on the geographic features it encounters. This data is derived by the PIRATE GIS using routines which scan each pipe segment to find where it enters and exits feature polygons. These are called *clash* routines.

Pipelines can be specified either initially at the CAD interface when entering contours and features, Section 7.4, or later at the PIRATE graphic user interface (GUI), Section 7.5.4. In either case the pipe is stored as straight line segments in vector form. Curves are not supported at this time though they could be included in later versions.

PIRATE needs to know not only *what* features the pipeline traverses, but where they are encountered and *the distance* through which they are traversed. The clash function provides this data by scanning along each pipe segment in turn, and checking each cell that is crossed, Figure 7.7(a). The exact distance through each cell is calculated by boundary/pipe intersection and the combination code found. Successive cells with the same code are amalgamated as one longer distance.

Results take the form of a list of combination codes crossed by each pipe segment, with the start and finish chainages of each occurrence. Figure 7.7(b) shows two typical pipe segments and the results table from clash analysis. The first segment starts in code c0, then crosses c2 before returning to c0. It thus gets 3 results records. The second segment travels through c0 throughout its length, and so gets only one result record. The chainages are local to the segment, i.e. they start from zero at the start of each pipe segment. This makes the table longer, but ensures that the figures are always referenced to a particular segment. In some cases the effect of a feature on a pipeline depends on the direction that the pipeline travels through it. Only pipe segments have constant direction, so local referencing must be used. Combination codes are later decomposed to feature codes in the AI toolkit.

When pipeline analysis is performed pipeline segments are pre-ordered so that the first segment to be considered crosses cells nearest the top of the main raster table. The direction in which the function scans along the pipe is also chosen so that access moves successively down the raster table. Every effort has been made to minimise backtracking and database enquiries so that the minimum number of disk page accesses on the raster table are required. With large database systems this is often a primary factor in the overall operating speed.

In the same scanning process a sectional terrain profile of the pipeline is built using the elevation figure for each cell rather than the combination code. This is stored in a separate table for section display and analysis at a later stage.

The use of exact cell crossing distances in the clash routines ensures that accumulative errors are minimised in the aggregate distances. The accuracy depends on the precision of real number arithmetic in the computer processor and the original precision of the raster representation. The latter is a function of the cell resolution chosen by the user. The only significant inaccuracy in clash distance is at the boundary of a feature, where the 'saw tooth' effect of rasterising can give a +/- 1/2 cell error in the worst case.

Figure 7.8 shows the true boundary and raster boundary of a feature, with the 'saw tooth' profile clearly visible. Pipeline A shows an example error in crossing distance. Pipeline B illustrates how the 'saw tooth' can cause small areas of feature to be shown as on a pipeline even though in truth the pipeline avoids it. In the case of pipeline routing these errors are tiny compared with the sweeping assumptions which have to be made about the character of the ocean floor. The errors obviously reduce if a higher cell resolution is used, and this is the users decision. PIRATE can use any cell resolution.

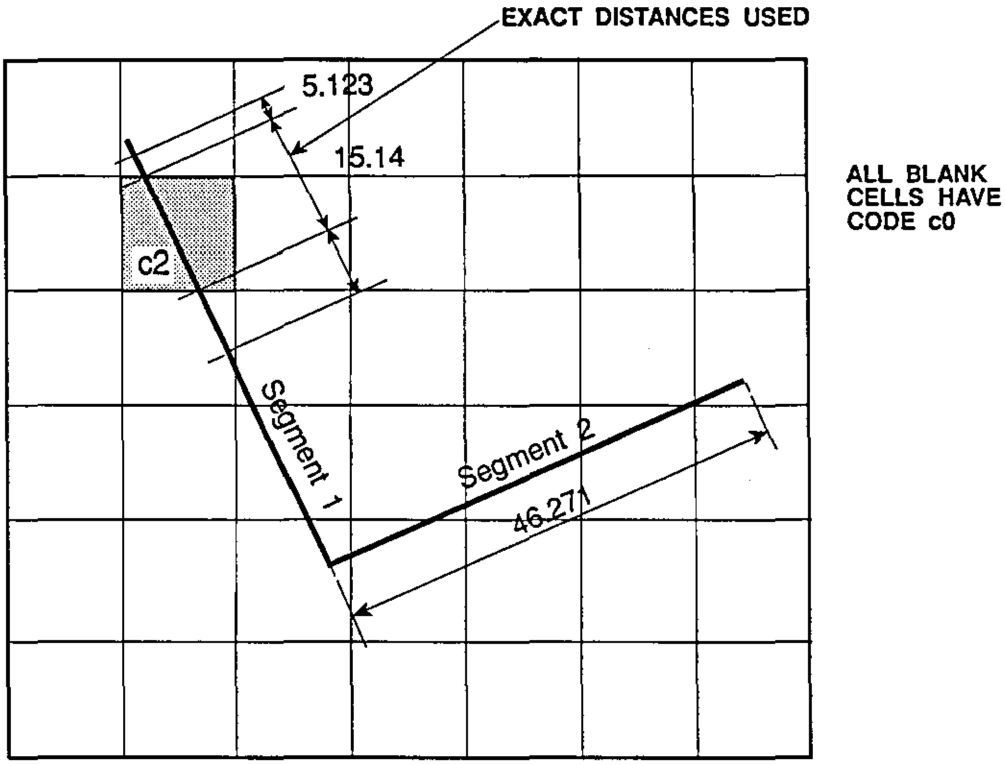


Figure 7.7(a)

PIPELINE CROSSING DISTANCES THROUGH THE RASTER GRID

PIPE_ID	SEG_ID	ATTRIB_ID	START_CH	FINISH_CH
1	1	c0	0.000	5.128
1	1	c2	5.128	20.271
1	1	c0	20.271	43.561
1	2	c0	0.000	46.271

Figure 7.7(b)

THE RESULT TABLE FOR THE ABOVE PIPELINE SHOWING EXACT DISTANCE CALCULATIONS

cd/5786H

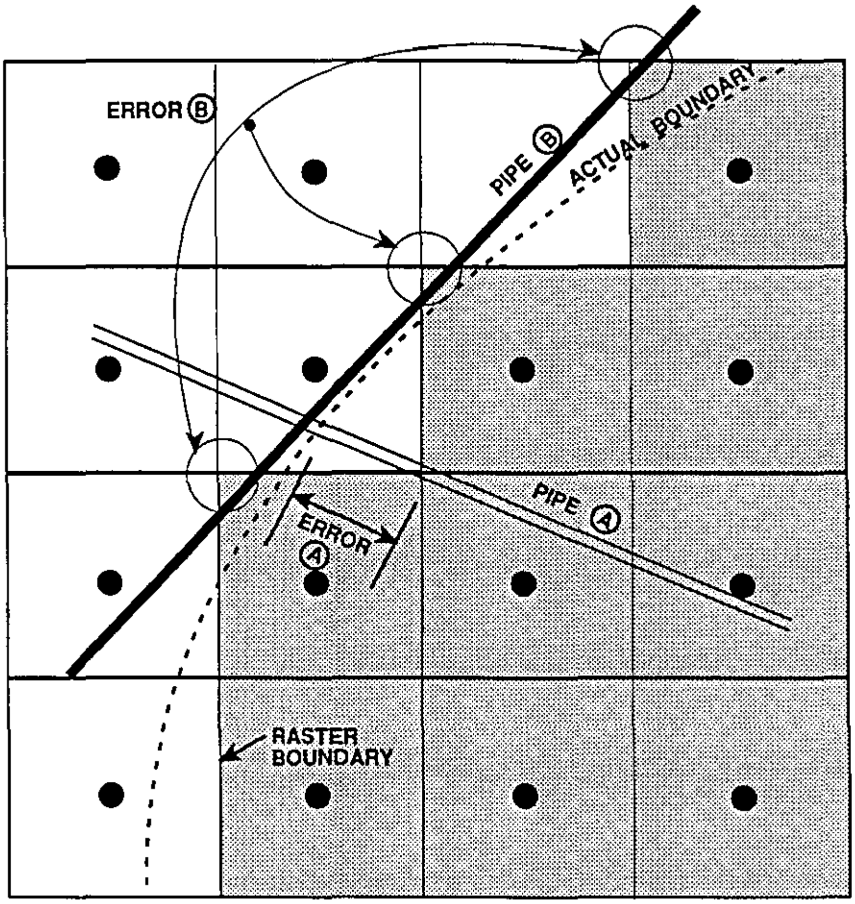


Figure 7.8

ERRORS IN PIPE ANALYSIS CAUSED BY THE RASTER REPRESENTATION

CD/57861

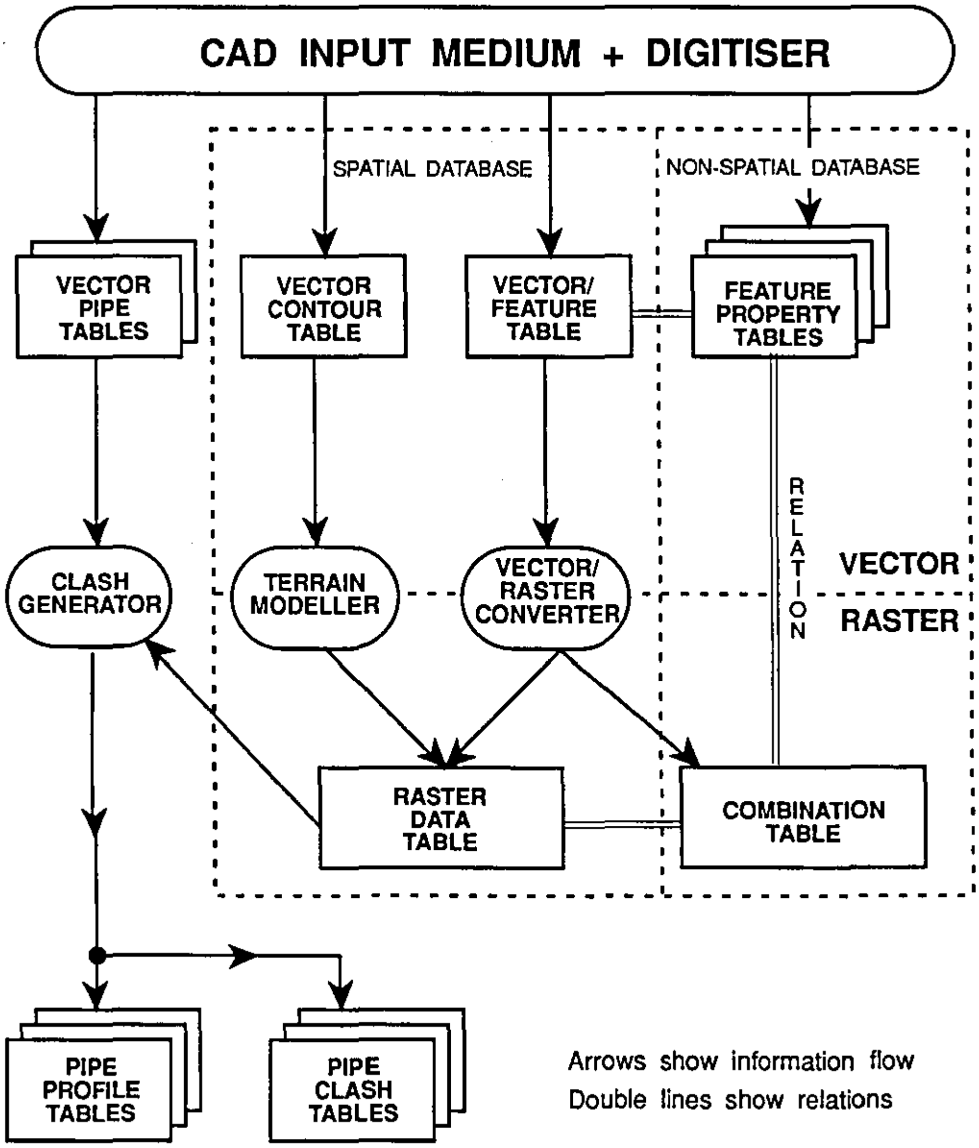
7.3.6 Implications of the PIRATE GIS

So far the text has described the raster and vector data structures of the PIRATE GIS, and how they are used for pipeline analysis. The GIS data model is of the author's design, having some unusual qualities that are particularly relevant to PIRATE. It is appropriate, then, to discuss these qualities (and difficulties), before going on to the 'intelligent' part of the system under GoldWorksII.

The PIRATE GIS data model is shown in Figure 7.9. Integrity maintenance becomes an important issue in cases where both raster and vector representations must agree. The PIRATE prototype by-passes many of these issues by constraining the user to a standard one way work flow (vector -> raster). Changes to the database can only be made by editing the vector representation and passing these through to the raster. A practical system could not limit the operator in this manner. It would have to ensure, for example, that when a feature is deleted from either part both representations of it are removed. With the PIRATE GIS this would be possible as there is a unique feature code which relates all images of the feature. However, no procedures have as yet been written to exploit this.

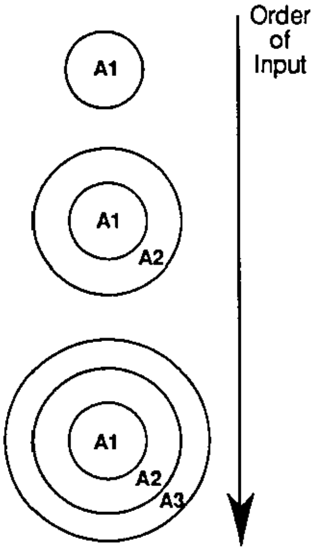
The raster and combination code structure is perhaps the most interesting and unique part of the PIRATE GIS. The single 2 byte combination codes (with the ASCII characters 30-255) allow 50,256 combinations before hitting a limit. An extra character would increase this to 11,390,625. The fixed field length and constant number of cells means that the main raster data table never changes size, no matter how complex the geographic area becomes. As this table is by far the largest in the GIS, its predictable size is useful in a storage management context.

The combination code table is the only one which changes size in the raster part of the GIS. It is interesting to note that the *order* of feature input can affect its size. Figure 7.10 shows two situations where three concentric features are being converted into the PIRATE raster database. In 7.10(a) the smallest feature, A1, is scanned first, followed by successively larger ones. Figure 7.10(b) shows the opposite. To the right of each figure is the resulting combination table as each feature is added. In 7.10(a) the combination table has redundances, codes which no longer exist in the raster table but are needed to maintain the linkage to the component features. Figure 7.10(b) stores the same amount of useful information, but in a non-redundant form.



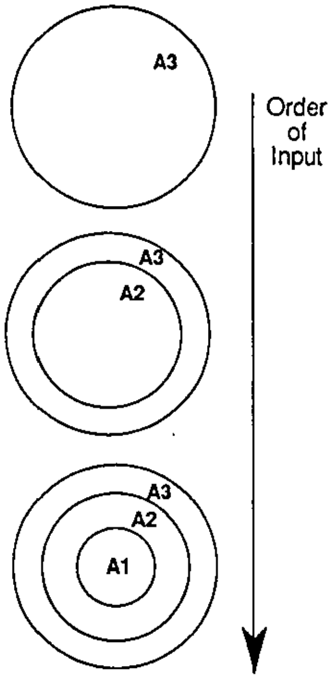
Arrows show information flow
Double lines show relations

Figure 7.9 A FULL SCHEMATIC OF THE PIRATE GIS DATA MODEL



	CURR_CODE	PREV_CODE	ATT_CODE
Stage 1	C1	-	A1
	C1	"	A1
Stage 2	C2	"	A2
	C3	C1	A2
	C1	"	A1
	C2	"	A2
Stage 3	C3	C1	A2
	C4	"	A3
	C5	C2	A3
	C6	C3	A3

(a) SMALLEST ENCLOSED FEATURE SCANNED FIRST



	CURR_CODE	PREV_CODE	ATT_CODE
Stage 1	C1	"	A3
Stage 2	C1	"	A3
	C2	C1	A2
	C1	"	A3
Stage 3	C2	C1	A2
	C3	C2	A1

(b) LARGEST ENCLOSING FEATURE SCANNED FIRST

Figure 7.10 HOW SCAN ORDER AFFECTS THE COMBINATION TABLE SIZE

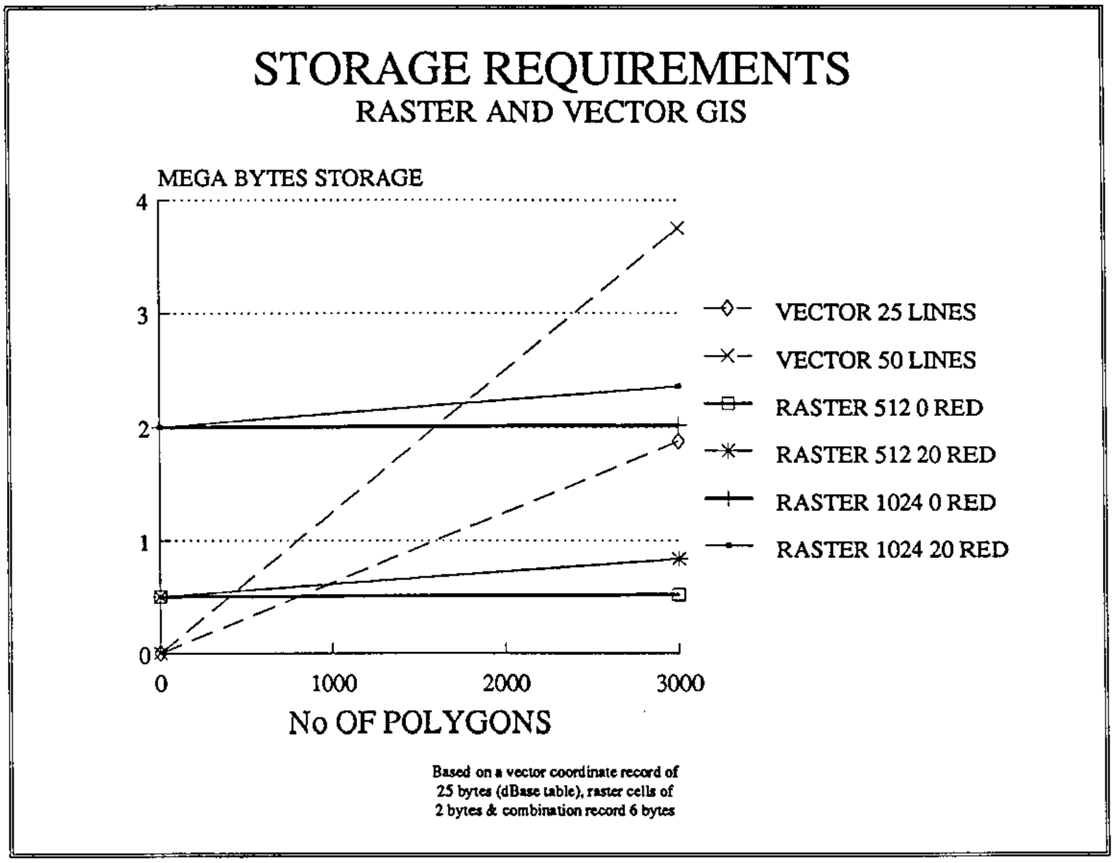
Potentially, the number of records in the combination table can equal 2^n-1, where *n* is the number of cells in the grid. If redundancy were avoided this number would simply be *n*. The first case is when the initial polygon is a single cell, the next is 2 cells enclosing the first, the following is three cells enclosing the first two polygons etc. The latter case is when the initial polygon is the entire region, the next is one cell smaller and enclosed by the first etc.

In the second case any feature polygon is completely spatially represented by a single record addition of 6 bytes. Hence the combination table would have the same number of records as the number of feature areas input. This is an extremely low storage addition. With practical projects there will be redundancy, but as redundant elements depend on the number of overlapping polygons at any point, normal data sets will tend to the lower bound rather than the upper. In complex data sets there may be some benefit in re-ordering polygons by size prior to rasterising, as this will reduce redundant levels. The saving is not predictable, however, due to the random nature of the data sets.

One can visualise the size of the combination code table for any data set, if redundancies were avoided. It would equal the number of polygons created if all feature boundaries were overlaid as a skeleton. Also, for any individual feature, the number of records created (including redundancies) is equal to the number of skeletal polygons existing within the new feature boundary just prior to placement.

Arguments are often put forward stating that the storage requirement of raster GIS is extremely high compared with a similar vector system. The raster technique developed for PIRATE, using combination codes, does not support this argument. Example situations have been examined, and the results are illustrated in the graph in Figure 7.11.

The test involves modelling a region with 0 to 3000 polygons present. Two cases are explored, with polygons having an average of 25 and 50 sides respectively. A vector model, such as that in PIRATE, requires 25 bytes per polygon boundary segment (2 coordinates, a line id and a polygon id). This is a 'spaghetti' structure with no locational indexing or specific topology. Two lines are plotted on the graph showing the overall vector storage requirements with increasing numbers of polygons.



Based on a vector coordinate record of 25 bytes (dBase table), raster cells of 2 bytes & combination record 6 bytes

Figure 7.11 A GRAPH OF STORAGE REQUIREMENT FOR VECTOR AND PIRATE RASTER DATA STRUCTURES

The PIRATE raster model, by comparison, requires 2 bytes per grid cell. Additionally it needs a minimum of 6 bytes per polygon in the combination table. Overlapping and redundancy may require additional entries at 6 bytes each. Two model resolutions have been assessed, 512x512 (equivalent to a 200m resolution of a 10,000 square kilometre area), and 1024x1024 (100m resolution of 10,000 square kilometres). Four lines are plotted, showing storage when one combination entry (0 RED) and twenty combination entries are, on average, needed for each polygon. As stated before, the number of combination entries depends on the type of data being modelled. Land ownership coverages, for example, do not overlap and so would need only one entry each. Multiple coverages of that type are unlikely to build that requirement significantly.

The graph line crossing points indicate that, especially at lower resolutions or at high data volumes, the PIRATE raster structure can compete with vector representations. Obviously these results are for a specific situation. Graph line positions would vary with number of boundaries per polygon and number of bytes per boundary in the vector structure, and the complexity of the region in terms of overlapping polygons and redundancy.

Locational data access is the key strength of the PIRATE raster structure. Once the cell pointer is calculated from point coordinates, itself a simple operation, the number of database accesses needed is equal to the number of overlapping features at the point plus one (the initial direct access to the cell code). This is independent of any redundancy in the combination table. All data table accesses are either direct or on indexed fields.

In a practical GIS point access and data input are only a small part of the library of functions available. Although not implemented in PIRATE at this stage the potential in the data model for other essential GIS functions must be investigated.

Deletion of feature polygons can be accomplished simply by setting all entries of the feature code in the combination table to null. Leaving the records in place maintains the connectivity for other codes, but the record no longer indicates that that particular feature is present. This leads to an ever increasing combination table size and a lengthening of access times due to extra links. If this is becomes unacceptable a separate compaction process could redirect the links in the table around the null members, which could then be deleted.

Moving/editing existing polygons would be a process of deletion and re-assertion. The raster database can be continually added to by scanning individual polygons at any time. The additive nature of the scanning process means that the raster table never needs to be re-created from scratch. It is recognised that editing the raster representation in this manner is not a trivial, or computationally inexpensive process.

Polygon overlay can be achieved by initially obtaining a list of feature classes or required properties from the user. The system must then find which features match the query, either using GIS feature property tables or the object representation in the AI toolkit, Section 7.5.1. The combination table can then be used to build a list of combination codes that contain features of interest. Scanning the raster table could give a display of cells with these combination codes, thus yielding regions where the required parameters exist.

Burrough (1989,p169) noted that "it is probably better to acquire a number of specialised modules that do a limited number of tasks well and link them together so they can make use of common data sources, than to attempt to find a single universal system that can do everything". The PIRATE GIS is designed in the spirit of this statement.

7.4 THE CAD SYSTEM INTERFACE

Getting data into PIRATE has been achieved by customising a CAD system. By using this approach the engineer can effect GIS data input as a by-product of creating CAD drawings. This point is significant as an argument supporting the cost effectiveness of any commercial application of PIRATE.

The system chosen was a pre-released version of the ProCAD Expert Series (AiC, 1989). Reasons for the choice included that it was written under Microsoft Windows version 2, and it had a macro language and a dBase interface. The Windows environment should have allowed it to run together with GoldWorksII, thus providing a full GUI for all aspects of PIRATE operation.

Unfortunately, despite assurances to the contrary from both GoldHill and ProCAD, the two systems would not reside in computer memory together. Using the Expert Series as the PIRATE GUI therefore proved impossible.

Despite this the Expert Series was still customised for initial data input. This satisfied the aim of creating PIRATE data as a by-product of normal drawing. The system enjoyed limited success with freehand drawing of polygons and the input of associated feature properties. However, the macro language was somewhat limiting, with only 50 numeric registers to store variables, a maximum of three dBase files open at one time, no local variables and a very sedate operating speed. Furthering the Expert Series development beyond data input would have been impractical in the light of these limitations.

The Expert Series not only proved that a CAD system could be used with PIRATE, but also lucidly illustrated that any system choice must be based on extremely detailed specifications, and possibly by acceptance tests prior to purchase.

7.5 PIRATE WITHIN GOLDWORKS II

The GIS described so far is essentially preparing and storing data for the main part of PIRATE, which runs under the GoldWorksII AI toolkit. GoldWorksII was chosen for reasons given in Chapter 6. In short it provides the knowledge formalisms needed to implement the PIRATE blueprint, it has satisfactory external interfaces, and it runs under Microsoft Windows.

For brevity the part of PIRATE in GoldWorksII will be abbreviated to PiG, with apologies for its rather inappropriate phonetic meaning.

When PiG is initialised all the information it needs to make decisions must get into the system. Section 7.5.1 deals with the initial data input, where PiG generates within itself the object based representation of the GIS, and reads spreadsheets to get other project data.

The passive part of the GUI is then described in subsequent sections, before pipeline design within PiG is outlined. Right up to this point the *intelligence* of the system has not been mentioned. Now, at last, *with all the information an engineer normally has to assess a pipeline available to the system,* the intelligent part of PIRATE is finally laid out. The processes and form of the PIRATE rule base and inference engine are given in Section 7.5.5. Finally the mechanism which allows PIRATE to act as a *geographic design spreadsheet* is outlined in Section 7.5.6.

7.5.1 Frames for Geographic and Other Data

PiG needs to have information about geographic features, design parameters, available construction plant and materials in order to assess any pipeline route. The frame hierarchy is used to hold all this data in a form the rules can access easily. (Frames were described in Section 3.3.2).

At the core of PiG is a frame based representation of geographic feature classes. Plate 7.1 shows these united under a parent frame MODEL_ATTRIBUTE. Each leaf frame in the tree is a geographic feature class. The parent frames in the tree allow inheritance of common slots between similar feature types.

In the PIRATE GIS each geographic feature class has an associated relational table. Fields in this table hold non-spatial feature properties, and individual feature polygons have one record each. PiG uses these tables to create instances for each polygon in the frame hierarchy, one instance per record, one slot entry for each field value. Instances are always created under the appropriate frame. Figure 7.12 illustrates the process of taking a record from a feature property table and converting it to an instance of a frame. PiG also reads the GIS vector table, storing the boundary coordinates of each feature in a slot of its instance.

These frames and instances make up the object-based PIRATE GIS representation. Spatial dualism is thus created, Section 7.4, giving PiG rapid locational and object-based interrogation of GIS data. The GIS description of the geographic region is now complete.

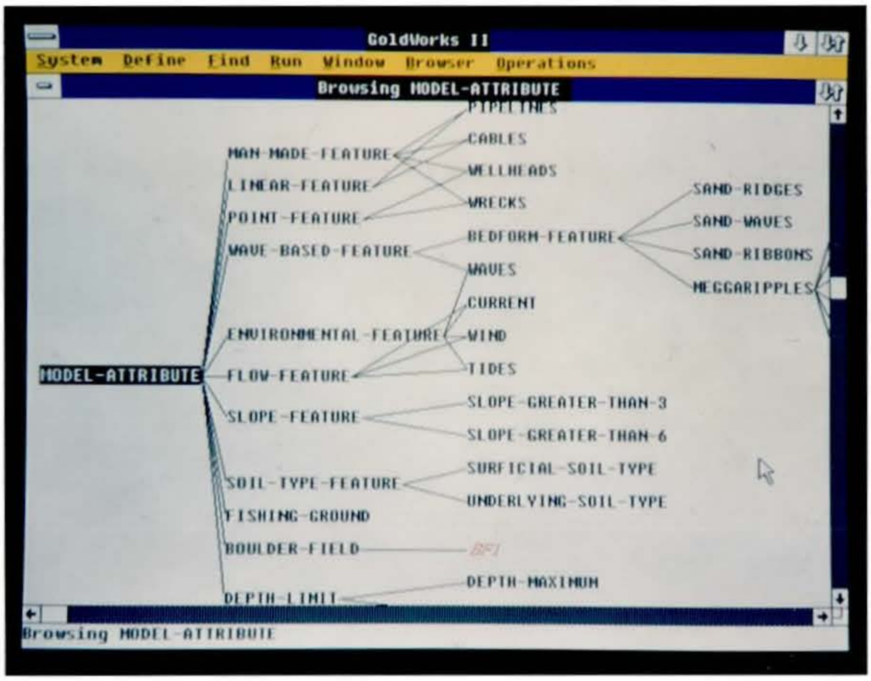


Plate 7.1 The Frame Hierarchy For Geographic Feature Classes

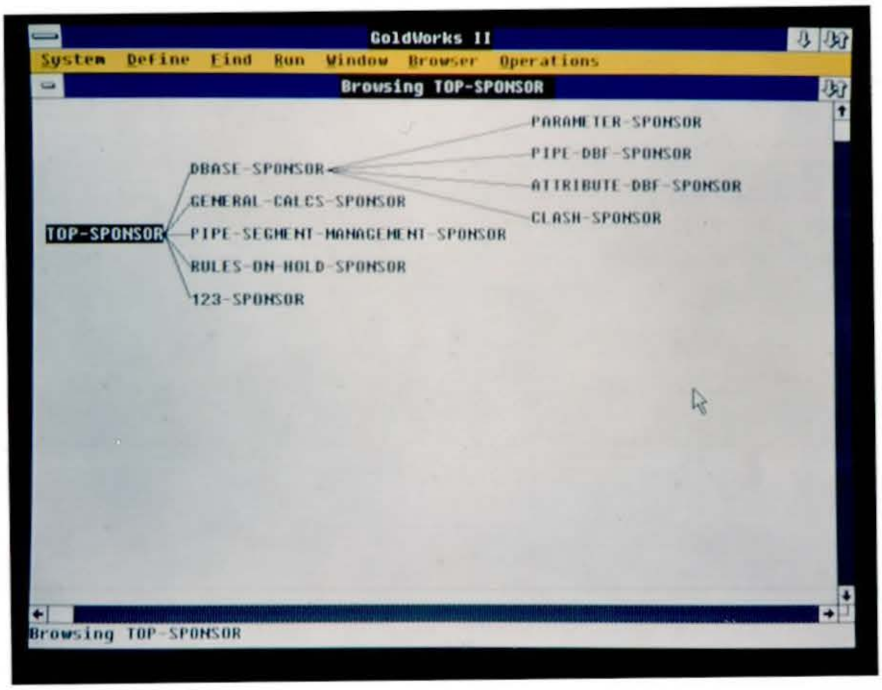
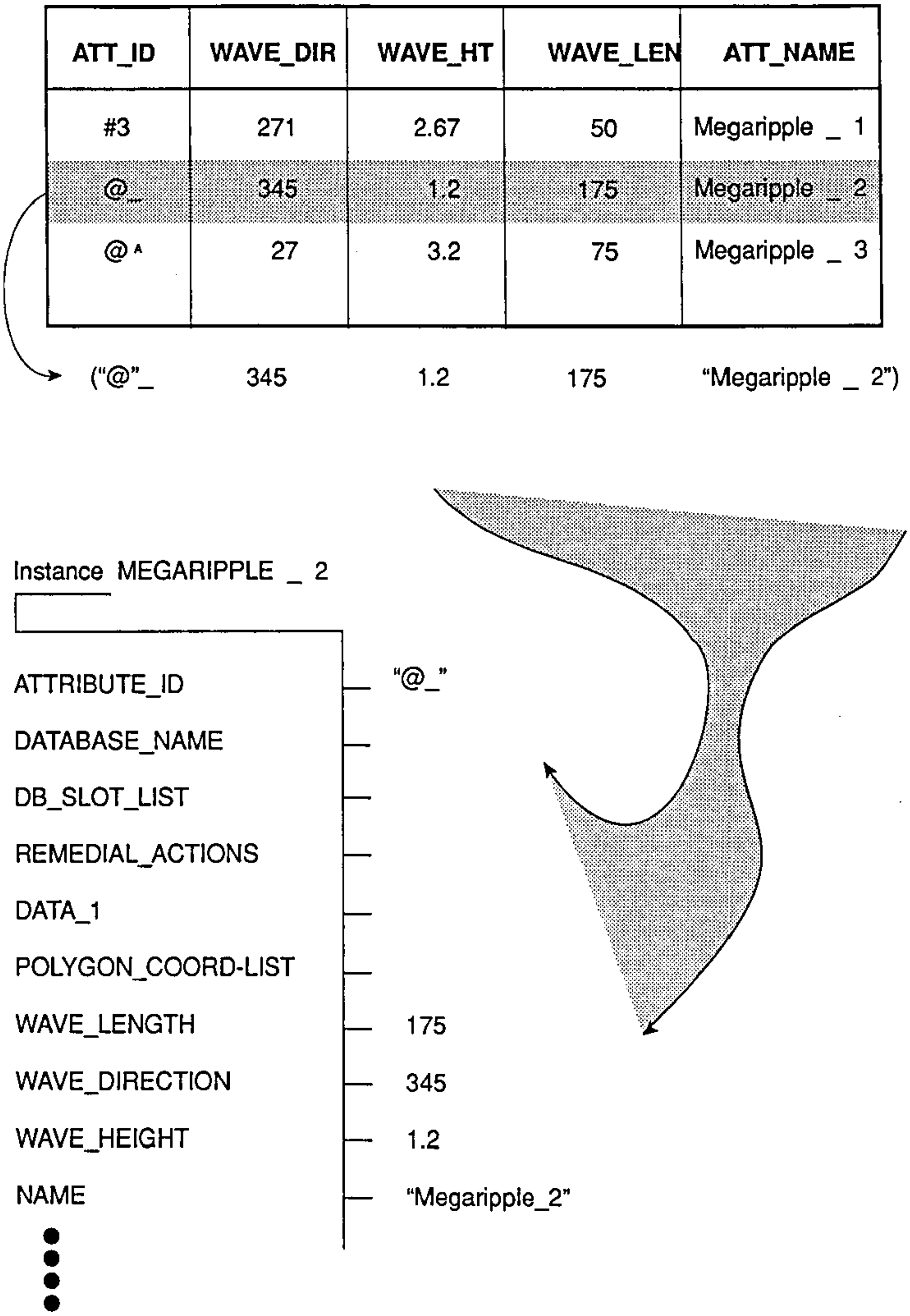


Plate 7.2 The Sponsor Hierarchy For The Forward Chaining Rules



ATT_ID	WAVE_DIR	WAVE_HT	WAVE_LEN	ATT_NAME
#3	271	2.67	50	Megaripple _ 1
@_	345	1.2	175	Megaripple _ 2
@^	27	3.2	75	Megaripple _ 3

Figure 7.12 CREATION OF FEATURE INSTANCES FROM GIS TABLES

cd\5786K

Other information needed to assess a pipeline route includes the available plant and materials, their specification and their cost. General data of different available items will be needed to compare pipelines using alternative plant or materials specifications. This data is subject to rapid change and is therefore not suited to being encoded directly in rules or frame instances. The maintenance of these facts in the knowledge base would simply take too much time. Instead PiG interfaces with spreadsheets in Lotus 123 format, from which it can draw data to produce instances that are up to date. The rationale behind this is that spreadsheet information can be regularly maintained by clerical staff, and may even be supplied by certain material vendors or plant hire companies. Figure 7.1, Page 105, shows the interfaces as a part of the general schematic of PIRATE.

PiG creates all of the above instances when it is first initialised. Forward chaining rules use database access functions to carry this out. The rules are grouped under sponsors in a hierarchy according to their function. The sponsor hierarchy is shown in Plate 7.2, page 130. When creating feature instances the rules refer to *base instances* of each feature class. Each contains the relevant database filename of the GIS feature property table, the number of instances created so far under this frame, and the text name to give to new instances. Each base instance sits below its parent feature frame, and new feature instances will join it as they are created.

In addition PiG uses a series of database control frames to monitor the status of each database accessed. Typically a database control frame has slots for database filename, current status (OPEN, CLOSED, FINISHED), and record status, showing how the latest record is progressing on its path to becoming an instance. Each database that is opened acquires an instance of a control frame. Rules use these control frames as a 'cue' so that they know when they must fire. Plate 7.3, page 135, gives a typical rule which takes part in creating feature instances.

PiG also uses control frames to govern the creation of plant and material instances from Lotus spreadsheets. However, as well as filename and status, these store the spreadsheet range coordinates of the data block to be read. The data arrives in PiG as a Lisp list, which is decomposed by Lisp functions prior to the creation of instances of plant or materials. The frame hierarchy for plant and materials sections of the knowledge base are shown in Plate 7.4, page 136, and Plate 7.5, page 136. Typical plant and material instances are given in Plate 7.6, page 137, and 7.7, page 137. Naturally slots in these frames are user definable as long as the user remembers to modify the spreadsheet and the control frame information accordingly.

Data which does not fit into any of the above categories includes the design information for the particular pipeline being planned. This would include the chosen pipeline diameter, the nation claiming ownership or jurisdiction over the region, and the code of practice used for the design, for example. The DESIGN_PARAMETERS frame is set aside for this type of data.

Many of the other frames in the overall hierarchy, Appendix G, are *system frames*. These control the creation of graphics, dialog boxes, and interfaces to databases and spreadsheets. The reader can find out more about these from the appropriate manuals (Gold Hill, 1989)

Finally, no mention has been made of the definition of pipeline routes themselves. These are a special case and as such will be dealt with in later sections.

7.5.2 The Graphic User Interface

Geographic features are passive elements in PiG. The system assumes that the PIRATE GIS has been used to ensure the data is complete and correct. Thus the PIRATE GUI provides no facilities for the editing or manipulation of feature data. 'Read only' operations are the only ones permitted, such as display and interrogation.

GUI development was prompted by the failure of the ProCAD Expert Series to reside with GoldWorksII. Built by the author using Golden Common Lisp and the Gold Hill Windows function library, the GUI is fully integrated into the PiG environment. The interface takes the form of a window, having the facilities of expanding, iconising and scrolling one would normally expect from Microsoft Windows.

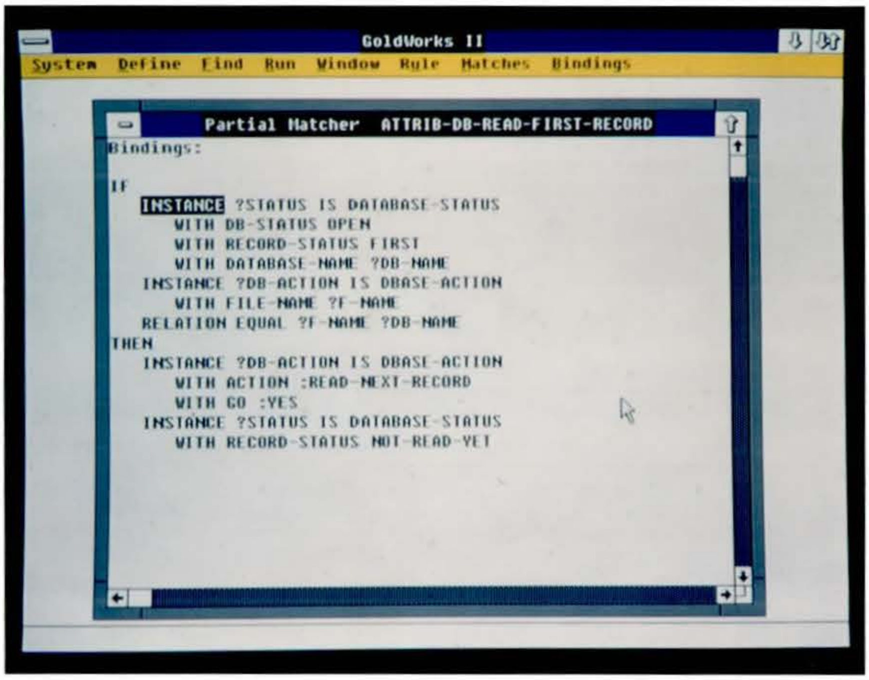
Any GUI must provide an easy to use and comprehensive interface for the operator. When using PIRATE the operator should not require access to the GoldWorksII frame interface, which can be unnecessarily complex. This puts the onus on the GUI to provide all the functionality the user needs. The PIRATE GUI aims to provide the following :

- A map display of all features in the GIS.
- The ability to overlay and highlight features in a composite image
- Interrogation of points and polygons for properties
- Utilities for zooming, panning, grid overlays etc.
- Control of all aspects of PIRATE operation
- Interactive pipeline design
- Provision of depth information and long section profiles of pipeline routes.
- Full reporting of pipeline analyses.

The first four aims concern the manipulation of passive geographic feature data and are achieved in two ways. Firstly the vector boundary data stored in the instance of each feature is used for drawing on-screen maps. A menu allows the user to pick which feature classes he wishes to display or highlight from a list of those known to exist in the frame hierarchy. Object-based overlay mapping is thus achieved, as illustrated in Plate 7.8, page 138.

Location-based queries are catered for by converting the co-ordinates of the queried point into a raster database pointer. PiG then directly accesses the GIS database to ascertain the combination code of the enclosing cell. Using the combination table this is un-nested into component feature codes. These are matched with the codes in feature instances in the frame hierarchy. Once feature instance identities are known more detailed reporting is a simple matter. Elevation values can also be obtained using this direct access approach.

Map grid overlay, full zooming and panning facilities are provided as menu choices in the GUI. The zoom scale can be set to the users preference, and fast zooming using a rubber band window is also offered. All these facilities were written in Lisp by the author.



Partial Matcher ATTRIB-DB-READ-FIRST-RECORD

```
Bindings:

IF
   INSTANCE ?STATUS IS DATABASE-STATUS
      WITH DB-STATUS OPEN
      WITH RECORD-STATUS FIRST
      WITH DATABASE-NAME ?DB-NAME
   INSTANCE ?DB-ACTION IS DBASE-ACTION
      WITH FILE-NAME ?F-NAME
   RELATION EQUAL ?F-NAME ?DB-NAME
THEN
   INSTANCE ?DB-ACTION IS DBASE-ACTION
      WITH ACTION :READ-NEXT-RECORD
      WITH GO :YES
   INSTANCE ?STATUS IS DATABASE-STATUS
      WITH RECORD-STATUS NOT-READ-YET
```

Plate 7.3 A Typical Rule For Accessing All GIS Databases

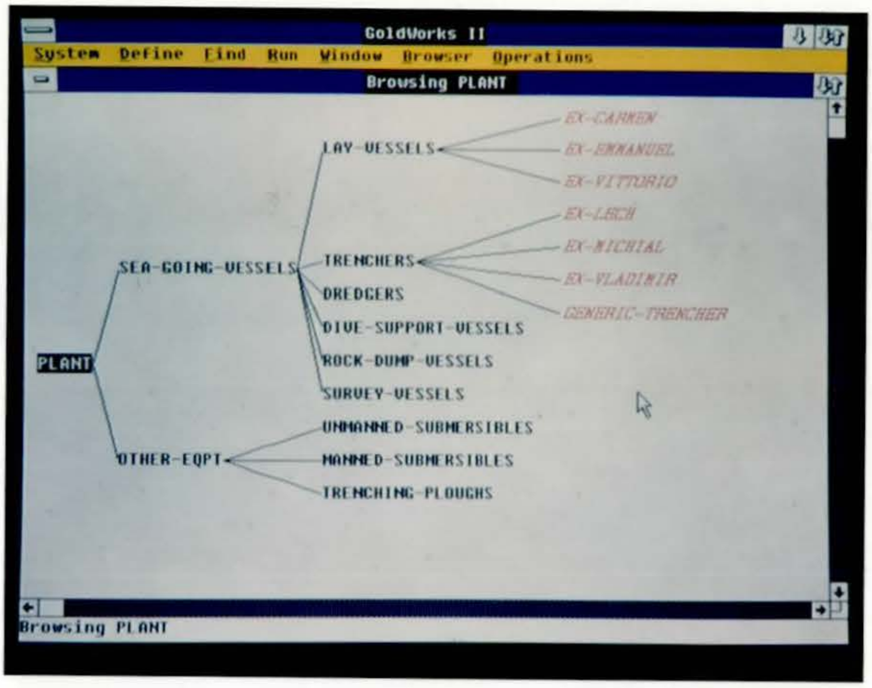


Plate 7.4 THE FRAME HIERARCHY FOR PLANT

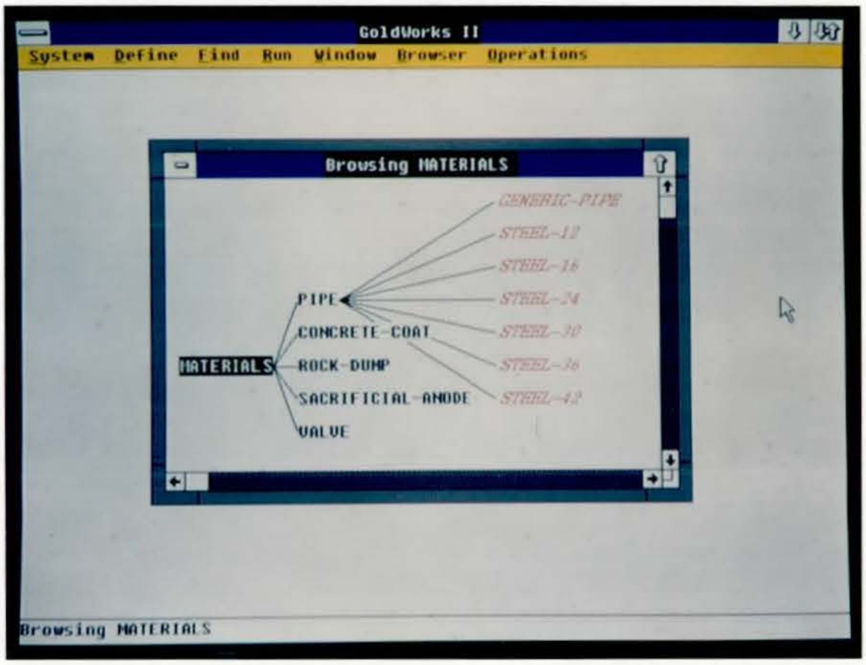
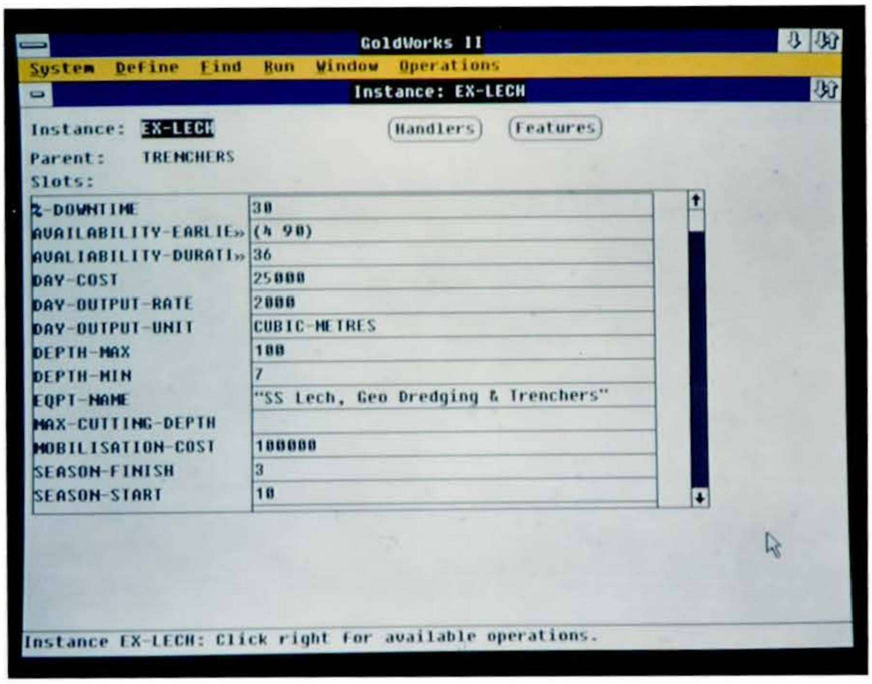


Plate 7.5 THE FRAME HIERARCHY FOR MATERIALS



Instance: EX-LECH

Instance: EX-LECH

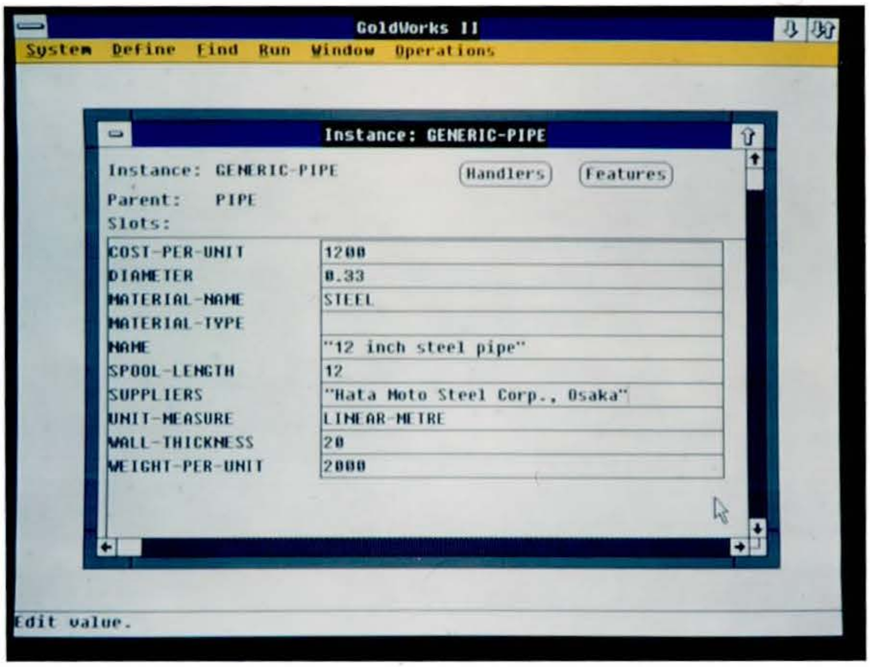
Parent: TRENCHERS

Slots:

%-DOWNTIME	30
AVAILABILITY-EARLIE»	(4 90)
AVALIABILITY-DURATI»	36
DAY-COST	25000
DAY-OUTPUT-RATE	2000
DAY-OUTPUT-UNIT	CUBIC-METRES
DEPTH-MAX	100
DEPTH-MIN	7
EQPT-NAME	"SS Lech, Geo Dredging & Trenchers"
MAX-CUTTING-DEPTH	
MOBILISATION-COST	100000
SEASON-FINISH	3
SEASON-START	10

Instance EX-LECH: Click right for available operations.

Plate 7.6 A TYPICAL PLANT INSTANCE



Instance: GENERIC-PIPE

Instance: GENERIC-PIPE

Parent: PIPE

Slots:

COST-PER-UNIT	1200
DIAMETER	0.33
MATERIAL-NAME	STEEL
MATERIAL-TYPE	
NAME	"12 inch steel pipe"
SPOOL-LENGTH	12
SUPPLIERS	"Hata Moto Steel Corp., Osaka"
UNIT-MEASURE	LINEAR-METRE
WALL-THICKNESS	20
WEIGHT-PER-UNIT	2000

Edit value.

Plate 7.7 A TYPICAL MATERIAL INSTANCE

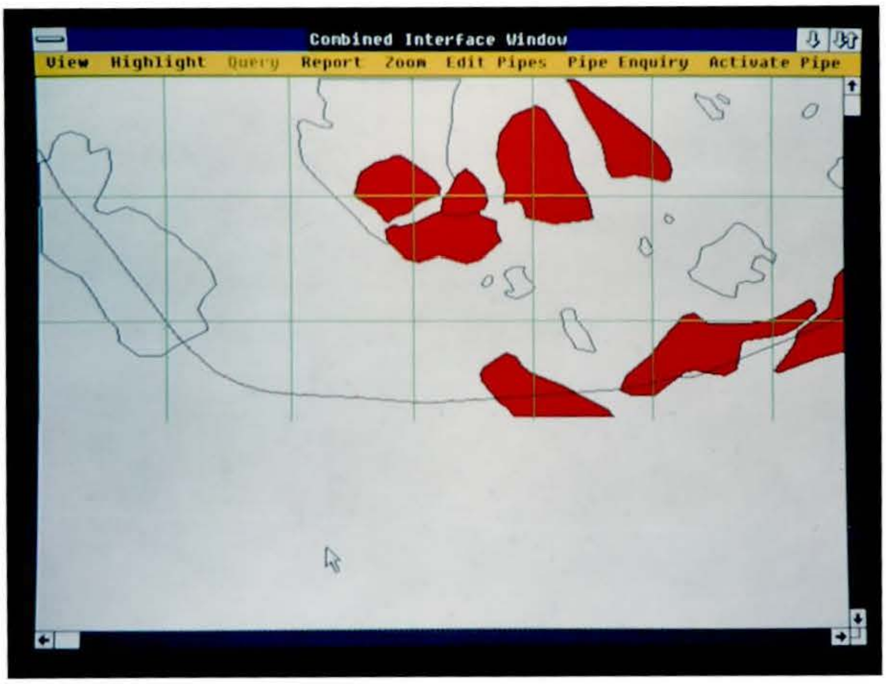


Plate 7.8 OBJECT BASED OVERLAY MAPPING AT THE PIRATE GRAPHIC USER INTERFACE

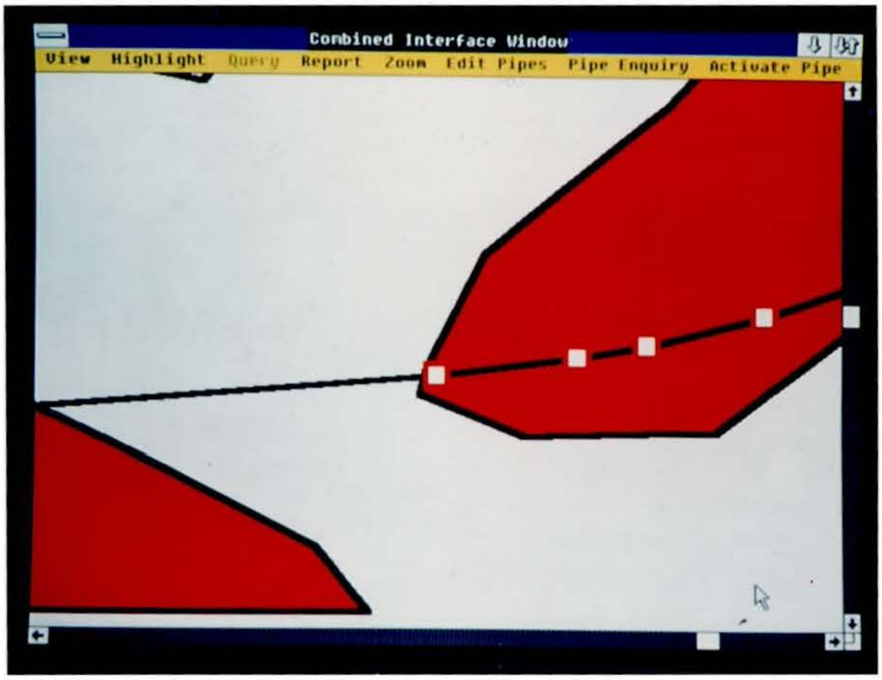


Plate 7.9 AN ENLARGED 'ZOOM' IMAGE SHOWING THE RASTER BASIS FOR THE PIRATE GUI

There are two constraints caused by the use of Gold Hill Windows in the feature display part of the GUI. Firstly, the zoom facility is raster based, as illustrated in Plate 7.9, page 138. This causes lines to thicken when zoomed in. Secondly, coordinates used for display have to be integer. This causes slight errors in feature display positions, though all PIRATE calculations are safely carried out using original data in the GIS.

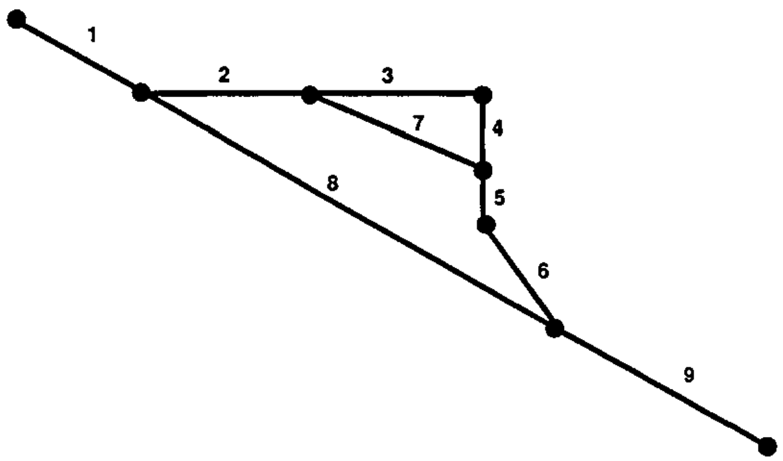
GUI images are created directly from values in slots, and these are accessed when any screen refresh is needed. This ensures that when any change is made to the information in the frame hierarchy the update of the graphic images is fully automatic. Changing a coordinate in a slot will automatically move the appropriate image, for example. This simplifies integrity maintenance between the graphics of the GUI and the knowledge base.

Other GUI functions, concerned with pipeline placement, interrogation and analysis, will be dealt with after the pipelines themselves have been discussed.

7.5.3 Modelling Pipelines

In the PIRATE GIS pipelines are stored and assessed by the GIS in terms of the individual straight line segments which approximate the true curved line, Section 7.3.5. This artificial division is decided by the user during pipeline input. Within PiG all pipelines are held in the frame hierarchy. Each pipe segment becomes an instance, whilst the pipeline as a whole gets a separate instance. The relationship of pipelines to segments is not simply *one* to *many*. More than one pipeline route can share segments, as Figure 7.13 explains.

Pipelines can either be created within the PIRATE GIS, or from the GUI. In the former case the pipeline is entered freehand or by digitising via the CAD input medium. The route is stored in a pipe database table, with fields for PIPE_ID, SEGMENT_ID, X and Y coordinates. On initialisation PiG reads this table by the same technique as the other databases are read, and instances of the PIPE_SEGMENT frame are created. If the route has already been assessed by the GIS clash routines this data is also read. Alternatively, pipelines can be designed interactively using the GUI, and this is recommended as the most productive method.



PIPELINES DEFINED BY CHOOSING MEMBER SEGMENTS:

PIPELINE A: (1 8 9)

PIPELINE B: (1 2 7 5 6 9)

PIPELINE C: (1 2 3 4 5 6 9)

NOTE HOW MORE THAN ONE PIPELINE CAN SHARE SEGMENT INSTANCES

Figure 7.13 DEFINING PIPELINES THROUGH PIPE SEGMENT NETWORKS

The PIPELINE_SEGMENT frame description is given in Figure 7.14, page 142. Essential data in the description are the start and finish coordinates. The frame, however, carries far more information than this. Many of the slots are for the benefit of the GUI, and these will be described later. Others store the results of analysis from the GIS clash routines, or from the reasoning mechanism of the expert system. Reasons for their existence should become clear in the following text.

7.5.4 Interactive Pipeline Design and Analysis

The PIRATE GUI considers geographic features to be passive, incapable of spatial change within PiG. Pipelines, conversely, are active and facilities are provided for their creation, editing, movement, and deletion. Without this PIRATE would be impotent.

This active status caused many problems during GUI design. The most severe concerned the need for pipe segment images to be 'mouse sensitive', allowing them to be pointed to and moved graphically. Gold Hill Windows cannot furnish mouse sensitivity except in the form of *hotspots*. These are user defined areas of the window which can detect the presence of a mouse pointer. They are normally used for creating mouse sensitive 'buttons' in user-friendly graphic interfaces. Pipe segments have been made sensitive by positioning small hotspots at each end point. Connected segments have to share a hotspot, otherwise shadowing occurs with one hotspot overlaying another.

Hotspot references are stored in the GRAPHIC_HOTSPOT slots of pipe segment instances, Figure 7.14, Page 142. When a hotspot is 'probed' using the mouse, lisp functions create a list of all segment instances which have the hotspot in their hotspot slot. If a particular segment is required it's two end point hotspots are probed. The two resulting segment lists can then be tested for joint membership, yielding the chosen segment.

Hotspots allow the user to create a connected pipe network over the map backdrop. Plate 7.10, page 144, shows a pipe network placed around some feature polygons. Both the black lines and the blue dotted lines are pipes, the blue dotted ones illustrating the effect of 'probing' a hotspot. Adding segments to this network is simple for the user through the GUI. Pipe creation functions ensure the hotspots of joined end points (network nodes) are found and shared, whilst new nodes have hotspots created for them.

```
(DEFINE-FRAME PIPELINE-SEGMENT
   ()
   (PIPE-ID :DEFAULT-VALUES (1))
   (SEGMENT-NUMBER-WITHIN-PIPELINE)
   (START-COORD :DEFAULT-VALUES (NIL))
   (FINISH-COORD :DEFAULT-VALUES (NIL))
   (START-CHAINAGE)
   (FINISH-CHAINAGE)
   (START-CHAINAGE-OF-ATTRIB-HIT
     :CONSTRAINTS (:LISP-TYPE LIST))
   (FINISH-CHAINAGE-OF-ATTRIB-HIT
     :CONSTRAINTS (:LISP-TYPE LIST))
   (DISTANCE-OF-ATTRIB-HIT :CONSTRAINTS (:LISP-TYPE LIST))
   (ATTRIB-HIT :EXPLANATION-STRING "This slot gives a list
of attributes in the order they are hit along the pipeline.
The other slots of start/finish chainages etc relate to
this list"
    :CONSTRAINTS (:LISP-TYPE LIST))
   (SEGMENT-ID)
   (GRAPHIC-HOTSPOT-1)
   (GRAPHIC-HOTSPOT-2)
   (GRAPHIC-WINDOW)
   (ORIGIN
     :CONSTRAINTS
          (:ONE-OF (CAD-DATABASE GW-INTERACTIVE BOTH)))
   (ORIGINAL-START-COORD)
   (ORIGINAL-FINISH-COORD)
   (ATTRIB-VALIDITY
     :DEFAULT-VALUES (NOT-GENERATED)
     :CONSTRAINTS (:ONE-OF (VALID NOT-VALID NOT-GENERATED))
     :when-modified (segment-dependency-daemon))
   (PIPELINE-MEMBERS
     :DEFAULT-VALUES (NIL)
     :CONSTRAINTS (:LISP-TYPE LIST))
   (A-LIST)
   (A-START)
   (A-FINISH)
   (A-DIST)
   (ATTRIB-MULTI-INST
     :CONSTRAINTS (:INSTANCE-OF MODEL-ATTRIBUTE)
     :MULTIVALUED T)
   (CONSOLIDATED-REMEDIAL-ACTIONS)
   (LONG-SECTION-LIST))
```

Figure 7.14 **THE PIPE SEGMENT FRAME**

Any node may be picked up and moved during the design, the shared hotspot ensuring that all associated segments are moved with it.

Once the network is drawn individual pipeline routes may be created by grouping segments together. The user traces the path of his chosen routes through the network by 'probing' successive network nodes. Each pipeline is given its own instance under the PROPOSED_PIPELINE frame, typified in Plate 7.11, Page 144. Slots in this frame detail the segments that the pipeline uses and their orientation, among other things.

Once a proposed pipeline route has been identified it can be assessed for feature clashes by the PIRATE GIS. On submission for GIS analysis, PiG creates a new pipe database table, using coordinates of the component pipeline segments. It then 'pushes out' to the DOS environment, first initialising new results tables, then running GIS clash procedures using the new pipe table. Results are posted to two database tables, one containing the feature clash data and the other holding a long section profile derived from the terrain model.

Returning to GoldWorksII on completion, PiG reads each combination code clash, together with its start and finish chainages. These are placed into the appropriate pipe segment instances as three lists. A-LIST stores combination codes, whilst A-START and A-FINISH store the chainages. The long section profile is read into each segment instance as a single list in the LONG-SECTION-LIST slot (see Figure 7.14, Page 142).

Combination codes have little purpose in pipe analysis if most of the rules refer to individual features. Lisp functions are applied to decompose the combination codes and chainages into their corresponding feature code lists. This is far from straightforward, as each combination code has an unknown number of features, and each feature can belong to an unknown number of combinations. Compiling chainages for features in this situation proved one of the most testing single Lisp programming problems of PiG. The solution incorporates a triple branching recursive function (a function which calls itself three times within itself). The function is given with its supporting functions and notes in Appendix H. The resulting feature code and chainage lists replace the combination code lists in their slots.

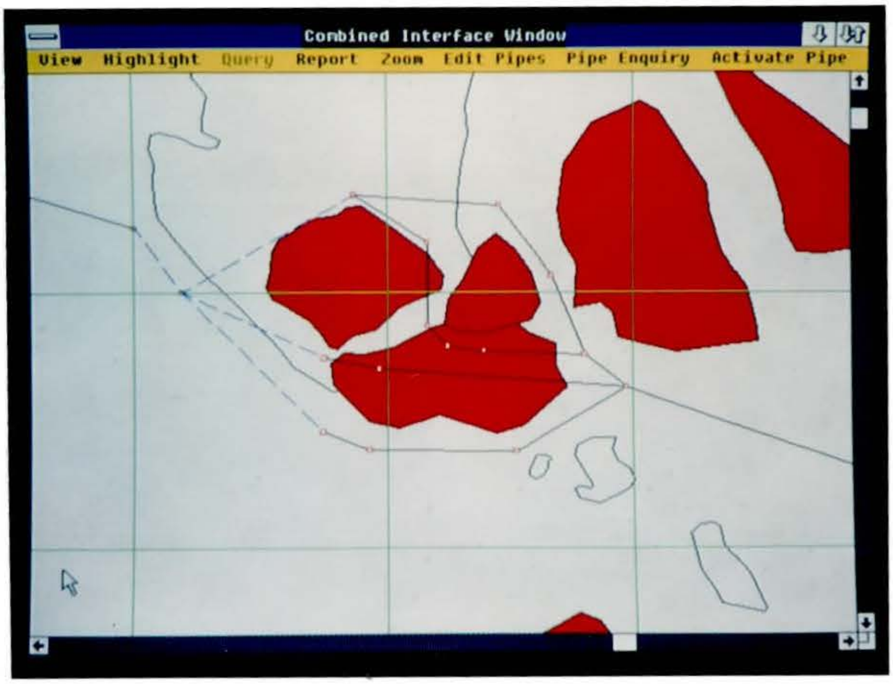
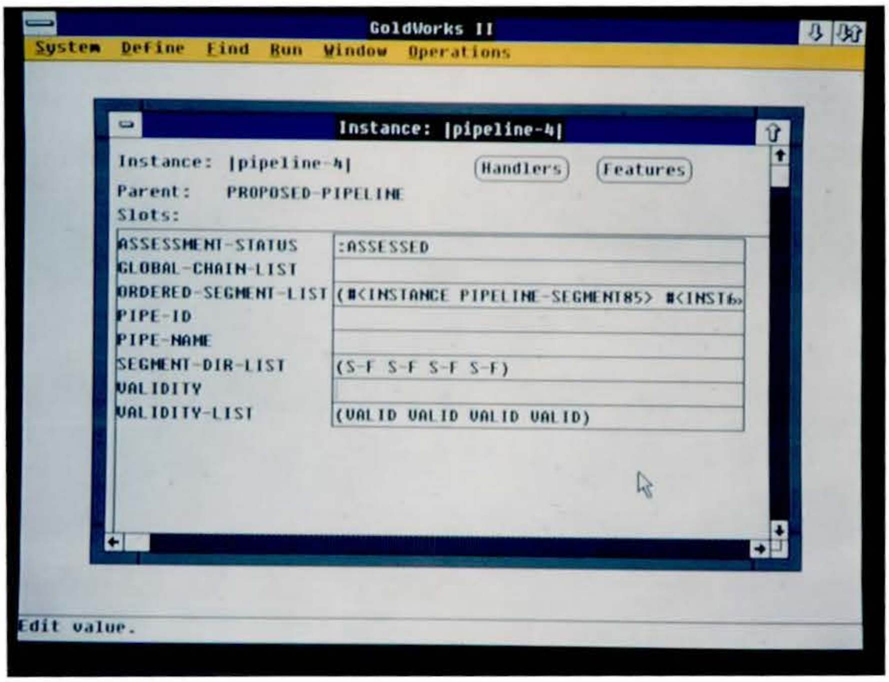


Plate 7.10 A TYPICAL CONNECTED PIPE SEGMENT NETWORK OVER A MAP BACKDROP AT THE PIRATE GUI



Instance: |pipeline-4|

Parent: PROPOSED-PIPELINE

Slots:

ASSESSMENT-STATUS	:ASSESSED
GLOBAL-CHAIN-LIST	
ORDERED-SEGMENT-LIST	(#<INSTANCE PIPELINE-SEGMENT85> #<INST6»
PIPE-ID	
PIPE-NAME	
SEGMENT-DIR-LIST	(S-F S-F S-F S-F)
VALIDITY	
VALIDITY-LIST	(VALID VALID VALID VALID)

Edit value.

Plate 7.11 A TYPICAL INSTANCE FOR A PROPOSED PIPELINE

7.5.5 Knowledge Based Route Judgements

The text to this point has described a system which is still preparing information. In its final form, all necessary information needed to judge a pipeline has been found and is accessible throughout GoldWorksII. The GoldWorksII inference mechanism naturally has access to all this data, and can use these facts as a basis for matching rules.

This state is THE crucial raison d'etre of PIRATE. The inference engine at last has FULL access to ALL data relevant the pipeline design, including all relevant geographic data.

The inference engine is also free to request additional geographic information from the GIS should it so wish. It can do this without curtailing inferencing.

In short, the inference engine has complete control over and direct access to the GIS, the GUI and all information pertaining to the proposed routes.

This sets PIRATE apart from many other intelligent systems using spatial information for design, which generally have only limited control over and access to any GIS they may use.

To judge the effects of geographic features on a pipeline PIRATE uses backward chaining rules. These rules were found during interviews with practicing pipeline engineers, described in Chapter 5. An example knowledge base was encoded in the prototype, and these rules are given in Appendix E. Other rules are still in natural English, and are detailed in Appendix C.

Some of the rules relate to individual geographic feature types. These tell the system what effect crossing a feature will have, for example crossing a megaripple field means that the pipe must be trenched. However, in this example the depth of trenching will depend not only on the wave height of the megaripples and the pipe diameter, but also the minimum cover requirement of the code of practice being used. Other rules take part in the backward chaining to provide such values.

Backward chaining was chosen for pipeline analysis because very few facts have to be found compared with the number of possible facts that could be inferred. If forward chaining were used the rules would assess the effect of *every* geographic feature, irrespective of whether or not it was on the pipeline. This is hopeless waste in processing irrelevant data. Backward chaining, by contrast, can only be invoked by issuing a *query* about a specific fact that needs to be found. A typical query might be :

> "What are the remedial actions that would be needed if a pipe crosses MEGARIPPLE-8 ?"

The queries are generated by the system as Lisp statements when required.

A typical backward chaining rule which relates to remedial actions associated with features is given in Figure 7.15, Page 149. This rule deals with trenching through megaripple fields, and results not only in trenching being recommended, but also calculates the depth of trench needed and the number of passes the GENERIC_TRENCHER would have to make to dig a trench that deep. The matching pattern, which the query is looking for, comes just after the THEN of the IF....THEN structure. If the matching pattern satisfies the query, the rule is used for backward chaining. The explanation strings and comments in the rules help the user to understand what the rules are used for, as the Lisp itself can be hard to read.

The examination of pipeline routes is triggered by the user from GUI menus. The user activates a route by probing all its segment end points. Provided the segments of the route have already been sent for clash analysis he can request one of a series of reports from the menu. There are four types of report, giving clash chainages, technical analyses, plant and material recommendations and cost estimates. Each can be requested for a single segment or a whole pipeline.

The clash report is the most straightforward, merely listing feature instances that the pipe crosses and the chainages of those crossings.

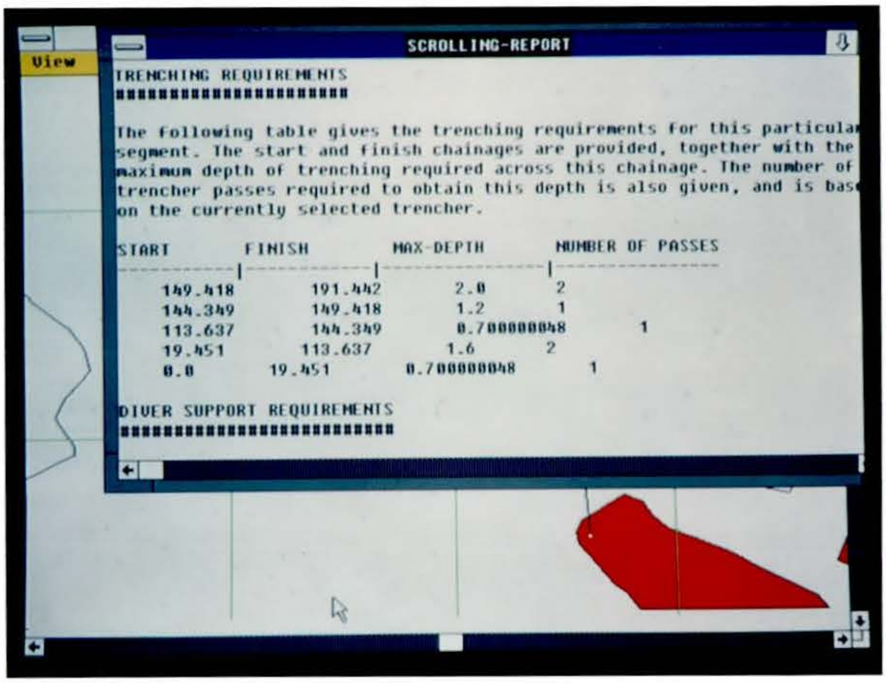
The technical report builds on this by judging the implications of the clashes in terms of the design, constructability and safety of the route. Remedial actions are suggested which would render the route safe to build.

The plant and materials report suggests equipment and materials which would be best suited to the construction of the pipe, whilst the cost report compiles a preliminary cost analysis based on results from the other reports. Part of a typical technical report is given in Plate 7.12, page 148. Currently only the segment clash and technical reports are fully implemented in the PIRATE prototype.

Control of the reporting process is not trivial. When a report is requested, the GUI sends the instance names of the chosen pipeline or segment to the REPORT-CONTROLLER. This frame has a slot for each report type, and a daemon that overlooks the value in each slot. A report is requested by setting the appropriate slot value to REQUIRED. The daemon, sensing this, initiates a number of relevant actions, including the generation of backward chaining queries if needed.

The technical report daemon is a suitable example of this process in operation. Each feature instance has a multivalued slot available to store remedial actions. The daemon asks each feature on the pipeline to provide the remedial actions needed for the pipe to overcome it. The demand is made by invoking a query. Thus, if no values are present in the REMEDIAL-ACTION slot of the feature, PIRATE automatically begins backward chaining to discover if any values can be inferred from the knowledge base. Once required remedial actions are known for each feature clash, the daemon can go on produce its report.

Engineers are inherently curious people, their position makes them legally liable for negligent errors in their design. It is often the case then, that a PIRATE user would want to know *why* a decision has been made by the system, or *how* PIRATE achieved a result. A quoted advantage of expert systems is that they can explain their reasoning, and PIRATE is no exception. Using the GoldWorksII explanation facilities the user can obtain a report on how any fact was discovered. Plate 7.13, Page 148, shows a typical explanation. Unfortunately the photograph is static, whereas the screen can be scrolled to see the rest of the text. Nevertheless it should give a clear enough indication of what to expect from the system. GoldWorksII uses the explanation text attached to each rule to try to make the trace more readable, with moderate success.



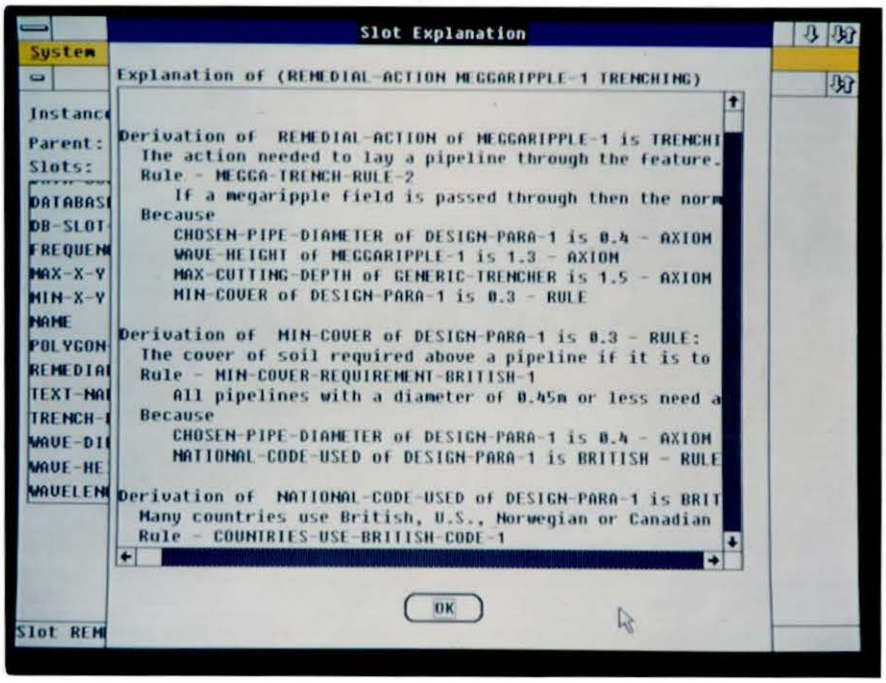
```
TRENCHING REQUIREMENTS
########################

The following table gives the trenching requirements for this particula
segment. The start and finish chainages are provided, together with the
maximum depth of trenching required across this chainage. The number of
trencher passes required to obtain this depth is also given, and is bas
on the currently selected trencher.

START         FINISH         MAX-DEPTH         NUMBER OF PASSES
------------|--------------|-----------------|----------------
    149.418        191.442          2.0        2
    144.349        149.418          1.2        1
    113.637        144.349          0.700000048          1
    19.451        113.637         1.6        2
    0.0         19.451         0.700000048         1

DIVER SUPPORT REQUIREMENTS
############################
```

Plate 7.12 A Typical Technical Report Giving The Results of Pipeline Segment Analysis



```
Explanation of (REMEDIAL-ACTION MEGGARIPPLE-1 TRENCHING)

Derivation of  REMEDIAL-ACTION of MEGGARIPPLE-1 is TRENCHI
  The action needed to lay a pipeline through the feature.
  Rule - MEGGA-TRENCH-RULE-2
     If a megaripple field is passed through then the norm
  Because
     CHOSEN-PIPE-DIAMETER of DESIGN-PARA-1 is 0.4 - AXIOM
     WAVE-HEIGHT of MEGGARIPPLE-1 is 1.3 - AXIOM
     MAX-CUTTING-DEPTH of GENERIC-TRENCHER is 1.5 - AXIOM
     MIN-COVER of DESIGN-PARA-1 is 0.3 - RULE

Derivation of  MIN-COVER of DESIGN-PARA-1 is 0.3 - RULE:
  The cover of soil required above a pipeline if it is to
  Rule - MIN-COVER-REQUIREMENT-BRITISH-1
     All pipelines with a diameter of 0.45m or less need a
  Because
     CHOSEN-PIPE-DIAMETER of DESIGN-PARA-1 is 0.4 - AXIOM
     NATIONAL-CODE-USED of DESIGN-PARA-1 is BRITISH - RULE

Derivation of  NATIONAL-CODE-USED of DESIGN-PARA-1 is BRIT
  Many countries use British, U.S., Norwegian or Canadian
  Rule - COUNTRIES-USE-BRITISH-CODE-1
```

Plate 7.13 The GoldWorksII Explanation Facility

```
(define-rule megga-trench-rule-2
         (
    :explanation-string "If a megaripple field is passed
through then the normal remedial action is trenching. The
trench depth is the sum of the pipe diameter, the minimum
cover required above the pipeline, and the wave height of
the megaripples in the field. The pipe has to be trenched
to avoid spanning, vortex shedding and possible collision
damage with anchors, trawl boards etc. In this case the
trench requires more than one pass of the trencher,
according to the maximum cutting depth of GENERIC-TRENCHER.
The number of passes is given in the third element of the
data-1 list"
    :direction :backward
    :dependency t)
  (instance ?attrib is meggaripples
         with wave-height ?ht)
  (instance generic-trencher is trenchers
         with max-cutting-depth ?cut-d)
 (instance design-para-1 is design-parameters
         with chosen-pipe-diameter ?dia)
  (instance design-para-1 is design-parameters
         with min-cover ?cover)
  (bind ?reqd-cut
     (+ ?cover ?dia ?ht))
  (> ?reqd-cut ?cut-d)
  (bind ?no-of-cuts
     (+ .999 (/ ?reqd-cut ?cut-d)))
  (bind ?int-no-of-cuts
     (truncate ?no-of-cuts))
  (bind ?data-list (list 'trenching ?reqd-cut ?int-no-of-
cuts))
 then
  (instance ?attrib is meggaripples
         with remedial-action trenching)
  and-then
  (instance ?attrib is meggaripples
         with data-1 ?data-list)
  (comment
    "In the case of trenching, data-1 elem 2 =depth of reqd
cut, data-1 elem 3 is number of passes required with the
generic trencher"))
```

Figure 7.15 **A Typical Backward Chaining Rule For Finding Remedial Actions**

7.5.6 Logical Dependency

If a pipeline is moved during the design the clash data associated with it becomes invalid. Every fact inferred from this data also loses justification.

GoldWorksII provides a system of logical dependency by which it can sense when supporting evidence for any fact is changed or becomes invalid. It is able to trace from the invalidated evidence through the knowledge base, retracting every dependent fact as it proceeds, until integrity is restored.

PIRATE uses the GoldWorksII logical dependency structure, but has to add significantly to it. GoldWorksII can only trace dependency when the facts were inferred through rules. PIRATE, however, uses many lisp functions during chaining. Any assertions made directly from these functions are lost to the dependency mechanism.

PIRATE overcomes this by mounting daemons on slots containing supporting evidence used by lisp functions. The daemons sense when slot retraction occurs and 'bridge the gap' in the dependency structure by retracting the result of the lisp function directly. GoldWorksII logical dependency may then continue the retraction through to conclusion. A single daemon function may be named to guard over any number of slots, and thus only one deamon function is normally required for every lisp function needing a 'dependency bridge'.

Once logical dependency has retracted information, backward chaining is not automatically invoked to replace the values. Rather, they are left uninstantiated until the user requests a report requiring them again. This 'request driven' approach minimises inferencing and adds to the overall speed of the system.

The logical dependency mechanism allows PIRATE to act as a *geographic design spreadsheet.* The user is free to change any data and thus observe the effects that such a change would have on the design.

7.5.7 Multiple Remedial Actions

For the technical report backward chaining produces a list of one or more remedial actions for each feature crossed. However, in cases where more than one feature occurs on a single length of the pipeline, an interaction between suggested remedial actions is inevitable. This is no problem if the remedial actions do not interfere with each other, diver support and trenching for example. In other situations, such as where two trenches of different depths are recommended for different features on the same pipe length, the system must find an overall plan which will ensure the safe installation and operation of the pipeline.

PIRATE has functions and rules which attempt to manage remedial actions where they do interfere. Two non-specific heuristics are used. In repetitive cases, such as when intensive diver support has been recommended twice, only one of the recommendations passes forward to be available for costing and reporting.

The other heuristic is applied in severity difference cases. In this situation the most severe of the operations is recommended. For example with trenches of different depths, the single deepest trench is put forward as the recommended action. These heuristics could not be implemented using GoldWorksII rules as they are cumbersome at manipulating lists recursively. Lisp functions similar to those used for decomposing combination codes, Appendix H, were applied instead.

7.6 SUMMARY

This Chapter has outlined a conceptual structure for an intelligent GIS for design. Further, it has described in detail the implementation of PIRATE, an intelligent GIS for off-shore pipeline route design.

PIRATE consists of two major modules, a GIS and an AI toolkit. The crucial integration of the two has been achieved by using *spatial dualism* for representing geographic features. The location based representation is stored in the relational tables of the GIS, whilst the object based representation is held in the frame hierarchy of GoldWorksII, the AI toolkit. Seamless access from GoldWorksII directly into the GIS relational tables has enhanced the integration.

Control for PIRATE is through a Microsoft Windows based graphic user interface (GUI). This is built in Lisp and has direct access to both the GIS and AI components of PIRATE. In addition to displaying maps and interrogating the GIS, the user can design pipeline routes and receive 'intelligent' design assessments directly from the GUI.

The PIRATE GIS uses both raster and vector data structures to optimise locational access speeds and graphic display response. A unique raster structure ensures rapid access and compact data storage with large numbers of geographic features. The GIS has been designed not to limit the volume of geographic data that can be stored.

The PIRATE implementation under GoldWorksII deals with object based feature representation, the GUI, pipeline design and rule based route assessment. Within the GUI, the pipeline design functions have been created with simple operation in mind, and an extensive range of editing and display facilities are provided.

The key part of PIRATE which makes it different from conventional GIS is it's ability to use rules to understand the significance of the geographic information it holds. Where PIRATE further distinguishes itself from other 'intelligent GIS' is that the inference engine has *complete and continuous* control over and direct access to the GIS, the GUI and all information pertaining to the proposed routes. In short, the PIRATE inference engine, and therefore the rules, can completely take control of all parts of the system should this be desired. Rules can access and use any geographic or other information at any time during inferencing.

The rules governing pipeline route assessment are modular, in that they have been designed to be added to or replaced without affecting the rest of the system. The entire rule base could be changed, for example, if a user wished to route electricity cables rather than pipelines.

PIRATE has been designed to support it's use as a *geographic design spreadsheet.* The user may change any fact in the frame hierarchy, or move any pipeline route, and PIRATE will incorporate the changes automatically into subsequent reports that it gives.

The overall architecture of PIRATE has been formulated such that it is independent of the eventual application. The same AI/GIS integration could be used in any area where intelligent geographic information systems need to be explored. The routing functions and rules, however, are naturally more specific to the pipeline design application.

PIRATE has been implemented successfully as a prototype to prove the concepts of intelligent GIS for design. This is reinforced by the findings of the case study, Chapter 8.

The Nantucket Clipper implementation language for the PIRATE GIS was efficient and provided all of the facilities needed for the GIS design.

The Expert Series CAD front end, not to be confused with the Graphic User Interface, proved the concept of a CAD interface. However, the implementation was less than adequate due to limitations in the CAD software and macro language.

The GoldWorksII AI toolkit excelled during the PIRATE implementation. The functionality provided by the system was crucial to the success of the project, and although extremely difficult to get to grips with at the outset, GoldWorksII was an excellent environment to work with. However, because it had just been released the software was 'buggy' to say the least. AI Limited provided a useful though at times unreliable hotline service, and update software often took several weeks to appear. Problems with the dBase interface were especially severe, with GoldWorksII unable to read database tables in excess of 5,000 records. This delayed the project, though the toolkit developers eventually produced a solution.

The Microsoft Windows graphic environment provides an easy to use interface for both PIRATE development and use. Windows version 2 had to be used as GoldWorksII had not been implemented with version 3. Memory management, especially within the DOS partition, was poor with PIRATE loaded. However, it is anticipated that future versions of GoldWorks would cure this.

PIRATE functions well in practice, and this is illustrated in the next chapter, which tests the system using data from a practical pipeline project.

CHAPTER 8

8. CASE STUDY - THE SABLE ISLAND PIPELINE PROJECT

8.1 INTRODUCTION

In 1983 a Canadian company, Sable Gas Systems Limited (SGSL), proposed a new off-shore pipeline to connect the Venture gas field with the coast of Nova Scotia. The Venture field is situated just East of Sable Island, which is itself 170 kilometres East of Nova Scotia. Figure 8.1, at the back of this thesis, is a chart showing the location of the field with respect to Sable Island and the Canadian coastline.

Between Sable Island and the Nova Scotian coast is an area of ocean floor which is extremely complex, rugged and fraught with difficulties for pipeline installation. Figure 8.1 gives the bathymetry of the region, the contours representing a 10 metre depth interval. Initially SGSL attempted the pipeline design, before commissioning J.P. Kenny & Partners (JPK) to conduct a full investigation into the options for the route and the design in general. JPK kindly supplied the author with the full library of documentation and drawings on the design, and it is from this that the case study information has been taken.

Sable Island sits on the edge of the Canadian continental shelf. To the East the sea bed falls away to depths of 5000 metres. To the West, the region in which the pipeline must be situated, depths vary between 20 metres and 300 metres, Figure 8.1. Immediately West of Sable Island a shallow sandy shelf precedes a steep fall into a complex region of deep gorges and basins. This trends North East to South West, blocking a straight line route option. Further West another shallow (90 metre) sandy plain precedes the Inner Shelf, a continuous rocky barrier extending out from the Nova Scotia coastline.

8.2 Manual Pipeline Design Methods

8.2.1 Identifying End Points

JPK engineers initially set about identifying potential landfall sites for the pipeline, and the manner in which the approach to the Venture field should be accomplished (JP Kenny, 1984a,b). Key factors in the choice of landfall included its distance from the Venture field and finding an acceptable route through the Inner Shelf barrier. Of the six sites assessed, Country Harbour was chosen as it allowed entry into an ancient submerged river channel which cut through the Inner Shelf. By routing the pipeline along this channel many of the problems of the Inner Shelf could be avoided.

Clearly, in a route following the course of a river channel there is little scope for alternatives or optimisation. On leaving the channel, however, potential routes are no longer as constrained. SGSL had already identified and assessed a 'Base Case Route' from Country Harbour, and JPK were asked to come up with alternatives. Figure 8.1 shows the Base Case Route as a solid line, and the JPK alternatives as dashed lines.

8.2.2 Producing Constraint Charts

As with any route assessment the primary considerations are the constraints imposed on the path of the pipeline balanced by the need to minimise route length and hence material costs. The word constraints in this context does not necessarily mean absolute constraints but in the majority of cases relates to conditions, such as rough sea bed, which would impose some cost penalty on the construction of the pipeline.

The approach taken by JPK engineers was concerned initially with the definition of constraints in the region, collated as a series of charts. The main constraint categories were considered to be :-

- **Bathymetry**, where possible depth constraints were identified
- **Sediment transport and bedform activity**, where potentially adverse effects of bedform activity were appraised, and potentially hazardous areas for the pipeline were identified

- **Surficial and underlying geology**, where sea bed soil conditions, slopes, boulders and local topography were appraised with respect to their effect on the pipeline
- **Environment**, where the effects of wave and current loading were evaluated primarily with regard to pipeline stability and the requirement for trenching
- **Man made and natural hazards**, where hazards such as wrecks, cables, well-heads, fishing areas, etc, were identified as constraints for a pipeline route

Constraints were considered with respect to their potential effect on the planning, installation or operation of the pipeline. A more exhaustive description of the effects of each constraint type, derived from the knowledge acquisition for PIRATE, is given in Appendix C.

Figure 8.1 is a typical constraint chart as drawn up by JPK engineers. It shows man made hazards in the form of wrecks and communication cables, assumed boulder fields, areas of bedform activity and the paths of vessels that have performed geophysical surveys. This is just one of sixteen constraint charts included in the study documentation (JP Kenny, 1985b).

8.2.3 Assessing Alternative Routes

Using the constraint charts JPK engineers designed alternative routes through the region, each route attempting to minimise contact with the constraints whilst also minimising route length, as shown in Figure 8.1.

Having identified the alternative routes, each was then assessed for 'clashing' with constraints. A series of tables was drawn up for ease of comparison, one table for each of the constraint categories listed in the previous section. Each route was given a short descriptive entry in each table, giving the main problems on the route due to that constraint category. Key aspects were then summarised quantitatively as a table of chainages, with columns giving distances through major constraints, such as boulder fields. Long section profiles were also drawn for each route so that bathymetry and slope could be examined more carefully.

On the basis of this brief descriptive and quantitative comparison two of the nine alternative routes were chosen for further study, involving more detailed design and a comparative cost analysis with the Base Case Route.

8.2.4 Costing

Comparative costs were built up using the following reduced list of key cost parameters:-

Material costs :

- line pipe
- corrosion and concrete coating
- sacrificial anodes
- field joints
- buckle arrestors

Construction cost :

- trenching
- pipe laying

Line pipe, corrosion coating, sacrificial anodes, field joints and buckle arrestors all have contributions directly proportional to the route length. The thickness, and hence cost, of concrete coating is also dependent on environmental loading, as its weight is used to provide stability. This is in turn dependent on route depth and current intensity. Trenching may be needed for a variety of reasons, due to mobile bedforms or areas of high environmental loading, for example.

Once environmental loading, concrete coat thickness and trenching requirements had been calculated the engineers used typical material costs, plant output rates, mobilisation costs and daily operational costs to produce a comparative cost for each route. Additional costs, such as crossing sub-sea cables or pipelines were also added to appropriate route totals.

The comparative costs were used to decide on the best route to continue with for detailed design and route surveys. Detailed design involves a complete structural analysis of the pipeline. For example, the existence of mobile bedforms and their precise configuration can result in overstressing of the pipe, free spans above the sea bed, and even possible exposure of a buried pipeline if the bedforms move (JP Kenny, 1985a). These problems obviously have to be resolved. A full cost analysis was carried out during detailed design. An example costing for the Base Case Route is given in Appendix D. This illustrates how the cost of a pipeline breaks down into individual elements.

8.2.5 Late Changes to Design Parameters

In the year following the completion of the JPK pipeline route design, but before the installation began, a new gas field called Thebaud was discovered to the South of Sable Island. SGSL decided that they needed the pipeline from the Venture field to collect gas from the Thebaud field as well. This represented a complete change in the initial design parameters of the pipeline, as it now effectively had two destinations rather than one. In an attempt to satisfy the requirements whilst minimising overall pipeline length JPK was asked to assess possible routes South of Sable Island. This would make the length of extra pipe to Thebaud shorter, though the length of the main pipeline would increase.

Changing the basic design criteria forced JPK engineers to go through the entire design process once again (JP Kenny, 1986). The resulting documentation was of similar length and complexity, and JPK engineers stated that a similar volume of work was involved for the re-assessment as for the initial assessment. Also, due to all routes South of Sable Island crossing a large area prone to bedform activity, a far more detailed assessment of the potential problems of overstressing, spanning, re-exposure and vortex shedding had to be performed.

8.2.6 Discussion

From the size and detail of the documentation for the Sable Island Pipeline Project it is clear that JPK engineers spent a great deal of time assessing route alternatives in the region. It is also clear that significant expertise was needed to perform the assessment.

The manual design method is labour intensive, taking up the time of engineers for long periods as they prepare constraint charts, propose routes and perform manual analyses. This is one of the major reasons why seven of the nine routes were eliminated after only brief evaluation. In some cases route elimination was clearly justified, but in others the reasons were far less apparent. It can be the case, according to JPK engineers, that on initial investigation two routes appear to have similar cost implications, so only one is fully assessed. Clearly there is a potential for some optimisation here, but using manual techniques the effort of performing detailed analyses in all cases is too time consuming.

The re-design of the pipeline due to the discovery of the Thebaud field clearly demonstrates the shortcomings of the manual design process. The repetition of such a vast quantity of work will be shown by this thesis not to be necessary, if a system such as PIRATE is used for the design.

8.3 PIRATE Tests Using Sable Island Project Data

The Sable Island Pipeline Project has been shown to be a complex feat of design requiring the consideration of many diverse parameters and areas of expertise. The PIRATE system is at this time in a prototype form, and because of this cannot be expected to deal with the entire design. This is simply because the knowledge acquired in the form of rules for pipeline routing has not to date been sufficient, Chapter 5. Rather than attempt a full emulation of the manual design process and results, this case study intends to prove that the PIRATE conceptual design is capable of performing such a task, and that the prototype can already achieve a considerable proportion of this.

In measuring the level to which the prototype has achieved the aims set out in Chapter 5, the following requirements were judged to be needed from PIRATE :

- Ability to handle practical volumes of geographic and other project data
- Provision of rapid, interactive methods to access and manipulate this data.
- Straightforward pipeline route placement and analysis procedures.
- Sufficient output to show that full reporting and cost assessment is merely a matter of extending the PIRATE program using procedures of a type which have already been shown to work. The output must also show that PIRATE is producing *correct* results

- Ability to change design data at any stage, with full implications being reflected in subsequent output *without* user intervention.

With the above points in mind it was decided to use one constraint chart, Figure 8.1, to provide representative geographic base data for the study. Existing boulder fields and megaripple fields were used as example constraint features, and these were augmented with extra polygons to make the tests more severe for PIRATE. The bathymetry was also used, contours being converted to a terrain model within PIRATE. The Base Case Route was employed to provide proof that PIRATE could handle the analysis of a full pipeline route.

The following sections document the progress of the data through the system, and the way PIRATE performs as a pipeline design tool.

8.3.1 Geographic Data Preparation

The geographic information for the case study is entirely represented in Figure 8.1. To get this information into the vector form needed by PIRATE, the lines on the chart first had to be digitised. This was accomplished using the ProDIGIT package from AiC (1990).

The constraint chart in Figure 8.1 was initially prepared by deciding on boundaries for the region to be digitised. The area chosen stretches from the exit of the Inner Shelf river channel in the West, to the Venture Field in the East. Local control coordinates were set to orientate and scale the region for digitising. The chart scale is 1:250,000 and the total area covered by the chosen region is 10,350 square kilometres.

The constraint feature boundaries, contours and proposed pipeline routes were digitised into three separate ASCII files, which were then converted into dBase tables. Every effort was made to ensure the accuracy of the digitising, though where contours merged this was difficult. In these cases the upper and lower contours were digitised, and the terrain model generator was relied upon to perform linear interpolation to fill in the slope between them. Tables for non-spatial feature properties were also built to link with the spatial tables.

Before the vector representation can be converted into the raster format that PIRATE needs, a decision has to be made on the resolution of the raster grid of cells. The accuracy of any results from PIRATE depend on this resolution, as any feature clash distance has a potential error proportional to the cell side length (Section 7.3.5). Two resolutions were considered, with cell side lengths of 100 metres and 200 metres. For the 100 metre resolution 1,035,000 cells, and hence records, would be needed in the main raster database. For the 200 metre resolution 258,750 cells would be required. PIRATE is designed to handle either database size, but it was noted that a 200 metre cell is only 0.8 millimetres square when converted to the chart scale. Any precision greater than this would be ineffective as manual digitising is rarely in excess of millimetre accuracy, and many of the lines on the chart have a width greater than 0.8mm anyway.

The chosen 200 metre resolution was entered, along with global coordinates for the local origin and the extent of the grid, into a parameters database table.

The vector to raster conversion proceeds in two stages. First the feature polygons themselves are rasterised. The PIRATE procedure for this is fully automatic, performing the conversion as soon as it is invoked (Section 7.3.3). The second stage is the creation of the digital terrain model (DTM) from the digitised contour table. This procedure is also fully automatic (Section 7.3.4). The two procedures complete the preparation of the PIRATE GIS location based description.

If a pipeline has already been digitised from existing charts, as it has been in this example with the Base Case Route, the pipeline clash procedure may be run before entry into GoldWorksII. This procedure analyses the pipeline route to discover the extent to which it has traversed constraint features (Section 7.3.5). Once again execution is fully automatic.

GoldWorksII is now entered and the PIRATE lisp files loaded. PIRATE automatically forward chains to read the appropriate dBase tables and set itself up for interactive pipeline routing.

8.3.2 Knowledge Base Rule Preparation

The knowledge base of rules derived from pipeline engineers is essential to the operation of PIRATE. As a result of the knowledge acquisition, detailed in Chapter 5, a large number of rules and general criteria on pipeline routing were discovered. However, as the case study uses just two feature classes in the geographic data, only rules relating to those particular feature types were included in the knowledge base. Primarily, the rules considered the different remedial actions that might be required in order to allow the pipeline to successfully traverse the feature.

Megaripple field features require the pipeline to be trenched, *boulder field* features can demand the use of divers to clear the boulders from the pipeline route and can also require the pipeline to be trenched. The trench depth depends on the diameter of the pipeline, the minimum cover required, and the maximum wave height of the megaripples in the field. Minimum cover requirements depend on the Code of Practice being used and the pipeline diameter. The Code can in turn depend on the country which has jurisdiction over the region though which the pipe is to pass.

Rules were tailored to model the above scenario, with the key aim of proving that the knowledge representation is both powerful and flexible, and that the inference engine works effectively and on cue within the interactive design process. Appendix E gives a full list of the rules in their original GoldWorksII format.

Rules can be entered into GoldWorksII in two ways. The most efficient for non-programmers is to use the rule editor provided within GoldWorksII. This can be menu driven to insert appropriate commands without having to remember the exact forms needed. Other tools, such as the Partial Matcher, allow a rule to be checked to see its effect without actually running the system or changing any facts. The other entry method is to write the rules directly into a Lisp file, which is quicker but the user then needs a thorough knowledge of the rule format in Lisp.

During rule preparation it was noted that to encode all the knowledge gained from the interviews into rules, should it be needed, would take considerable time. Also, even after extensive knowledge elicitation, it was still unclear in some areas as to precisely how the knowledge should be encoded. This suggests that to obtain a fully developed

PIRATE system additional interviews would be needed, and perhaps more sophisticated methods of knowledge representation.

8.3.3 Processing Times and Efficiency

The processing times needed for each stage of PIRATE data preparation provide a useful indication of the system efficiency and practical usability :-

Manual Data Entry	
Digitising Chart and Creating Tables	: 2 days
Creating Databases	
Creating Empty 260,000 Cell Raster Database	: 4 minutes
Vector/Raster Feature Polygon Conversion	: 26 minutes
DTM Creation	: 35 minutes
Initialising PIRATE in GoldWorksII	
Loading GoldWorksII	: 5 minutes
Loading PIRATE into GoldWorksII	: 2 minutes
Initialising PIRATE (Base Case Route Data)	: 16 minutes
System Interaction	
Analysing Base Case Route (44 Segments)	: 45 seconds
Point Interrogation at the GUI	: 2 seconds
Technical Report Generation	: 10 seconds

Times are based on the use of an IBM Model 70 80386 personal computer, with a 20MHz clock speed, 80387 math coprocessor, 14Mb main memory and a 120Mb, 28ms hard disk.

Digitising the constraint chart took a considerable amount of time due to the sheer volume of information present. The chart itself was produced manually from Canadian Hydrographic Charts, presumably by tracing. If a CAD system had been used for constraint chart creation, however, digitising would have been needed anyway. In this case much of the effort in preparing PIRATE data could be offset as being needed for drawing creation. PIRATE data input would then effectively be 'free', a by-product of drawing creation.

Creating the complete raster database ready for PIRATE to use takes just over an hour. This is respectable when one considers that over one quarter million interpolations must be performed for DTM creation alone. The contour data for the interpolation consists of approximately 17,000 line elements. Care during GIS function design has contributed significantly to the final operating statistics. Full analysis of the Base Case Route in 45 seconds, producing not only feature clash data but also full long section coordinate data, is also a more than adequate response given the processing involved.

The time taken to load PIRATE into GoldWorksII is comparable with many of the demonstration systems written for GoldWorksII. However, the PIRATE initialisation procedure is fairly lengthy, and this is mainly due to the system having to read pipeline analysis information in addition to creating the object based GIS representation. However, once initialisation is complete the actual interactive operation of PIRATE is not prone to such delays.

8.4 Using PIRATE For Pipeline Design

Once PIRATE is loaded and initialised within GoldWorksII, pipeline design can begin. The following sections deal with aspects of system operation in the order that the engineer would commonly use them. Naturally, as would be expected of an interactive design environment, the functions illustrated may be used in any order as long as the necessary information is present in the system.

The reader should note that *all* facilities described as part of the PIRATE GUI have been written by the author, using approximately 2,000 lines of Lisp code. None of the facilities for such a graphic interface were available from the GoldWorksII system. The development of the PIRATE GUI took several months, with the express intention of proving that a truly interactive and intelligent interface could be achieved.

8.4.1 Using The GUI For Interrogation

Plate 7.8, page 138, shows a part of the Sable Island case study region as it can be represented at the GUI. This plate shows boulder field and megaripple feature areas, and the SGSL Base Case Route. The engineer has a variety of facilities for creating, manipulating and interrogating the map, all of which are accessed through the menus at the top of the window. The menu hierarchy is given in Figure 8.2, Page 167.

Views are generated by the engineer picking the items he wishes to see from a sub-menu under the 'View' menu option. The menu is updated dynamically whenever the feature types in the region change, so that the user always has a correct list to choose from at the menu. A grid display option is also provided. In Plate 7.8, Page 138, all megaripple fields have been highlighted by choosing the appropriate item under the 'Highlight' menu. Using these facilities the engineer can built constraint charts quickly and easily.

Contours are not currently displayed by the GUI. The number of line segments involved in the display was expected to cause problems with insufficient memory for storage. The engineer is still free to make use of pointwise depth interrogation or request long sections.

A comprehensive series of image manipulation facilities is provided under the 'Zoom' menu option. Zooming in and out is straightforward, and the zoom scale can be changed, A 'Fast Zoom' option allows the user to 'window in' on an area of interest, whilst 'Redraw' and 'Full Window' options allow rapid recovery. Microsoft Windows, the implementation environment used by the developers of GoldWorksII, restricts the graphics to a raster form, as is illustrated by the thickening of lines in Plate 7.9, page 138. The image maintains its functionality, however, and this attribute does not appear to cause any problems during operation.

The designer can interrogate the on-screen map in a number of ways. By choosing 'Attrib at Point' under the 'Report' option, for example, he can find out the existing features at any point. This requires direct access to the PIRATE GIS database from the GUI. 'Attrib at Point' gives the names of the feature instances at a point, but any other information about the features could be accessed by the same mechanism.

Information about any pipelines in the system can be accessed using items under the 'Pipe Enquiry' option. Individual pipeline segments may be identified for enquiry by using the mouse pointer to 'probe' on the hotspots at the segment end points. The hotspots are clearly shown in Plate 7.9, Page 138. Reports are printed in 'popup windows', which can be closed by the user when no longer needed.

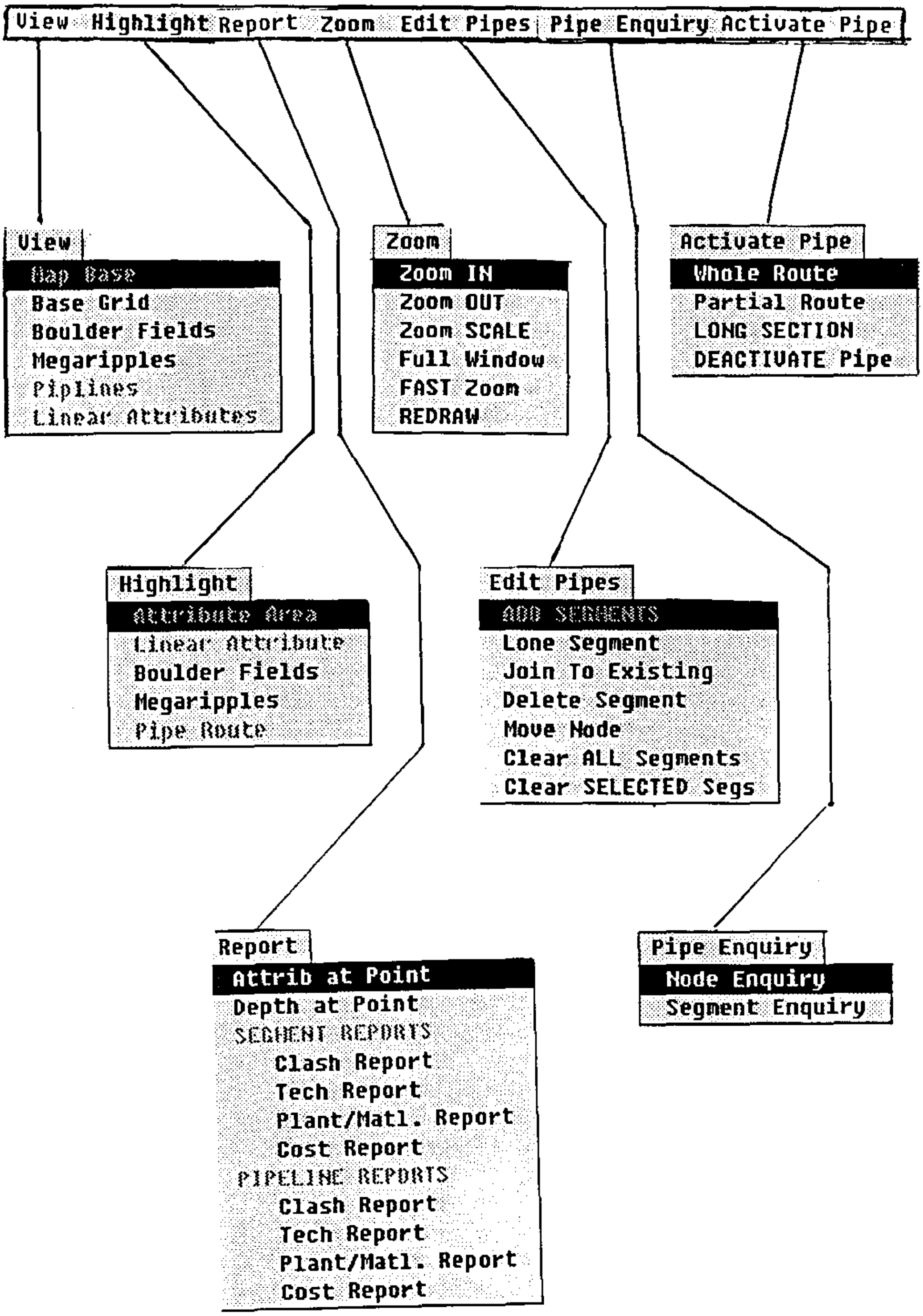


FIGURE 8.2 THE PIRATE GUI MENU HIERARCHY

The GUI has been designed to show how PIRATE can give the engineer freedom of access to his data. It also illustrates how the user can be shielded from the main GoldWorksII interface, which is of little relevance to the engineer and can over-complicate using PIRATE. if necessary the GUI could be extended so that the PIRATE user need never know that he was using an expert system at all.

8.4.2 Pipeline Placement

Having familiarised himself with the data using the GUI interrogation facilities, the user will wish to begin interactive pipeline design. Pipelines are placed on to the map as a connected sequence of straight line segments, by using 'Edit Pipe' menu options. Segments may be placed in isolation, or joined to other segments by probing an existing end point. The order of pipe segment placement is not important. Typically a series of alternative routes may be designed, as illustrated in Plate 8.1, Page 170. Each segment, as it is placed, will gain an instance in the frame hierarchy containing all the information known about it. PIRATE maintains this record throughout the life of the segment.

The PIRATE GUI has comprehensive pipeline editing facilities. For example, Plate 8.1, page 170, and Plate 7.10, page 144, show how a node in the pipeline network can be selected using the mouse and moved to another location. Note that the moved segments change from black lines to blue dotted lines on the Plates.

As noted in Section 7.5.4, any changes made to the graphic images are automatically reflected in the frame hierarchy. In the case of moving a node each segment that has moved will have its coordinate slot values updated, and all other information relating to its previous location will be altered. Segments may also be deleted, in which case they lose their instances in the frame hierarchy.

If changes are made to the frame hierarchy directly, the deletion of a feature or segment instance for example, it will cause the graphic image to be updated automatically. All graphic images are generated directly from information contained within the frame hierarchy. There is no duplication of coordinates, for example, and the GUI refers directly to instance slot values to get the coordinates it needs.

This integral link between instances and images not only maintains integrity at all times, but gives the knowledge base implicit control over the GUI image by its explicit control over the data in the frame hierarchy.

8.4.3 Pipeline Activation and Analysis

Pipeline routes are a connected series of pipeline segments. Each route is chosen from the network of pipe segments by using the 'Activate Pipe' menu and choosing the 'Whole Route' option. The route may then be activated by probing the end points of each member segment in turn. The end of the pipeline is indicated by probing the final end point twice.

When a pipeline is activated it gets an instance of its own which holds information about member segments (Section 7.5.4). The user is then given the option to send the pipeline for assessment. If this is done the pipeline coordinates are passed to the PIRATE GIS for clash analysis and long section profile generation.

To provide a thorough test for PIRATE using the case study data, a hypothetical pipeline route was designed to cross the most complex part of the study region. Plate 8.2, page 170, shows a segment of this route as it crosses an area with up to three overlapping features. The route was activated and sent for analysis by using appropriate GUI commands.

8.4.4 Long Section Profiles

One of the segments in the test route was placed to traverse the complex bathymetric trench that cuts the study region to the West of Sable Island, as shown in Plate 8.3, page 172. The reader is encouraged to compare its position with the contours of the original chart, Figure 8.1.

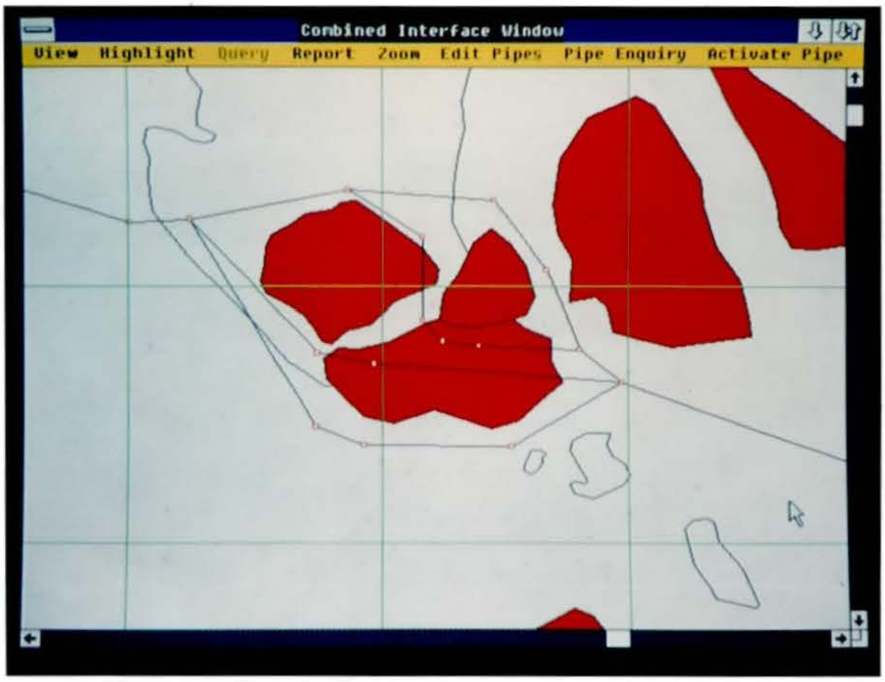

Combined Interface Window
View Highlight Query Report Zoom Edit Pipes Pipe Enquiry Activate Pipe

Plate 8.1 PIPE SEGMENT NETWORK DESIGNED AROUND CONSTRAINTS

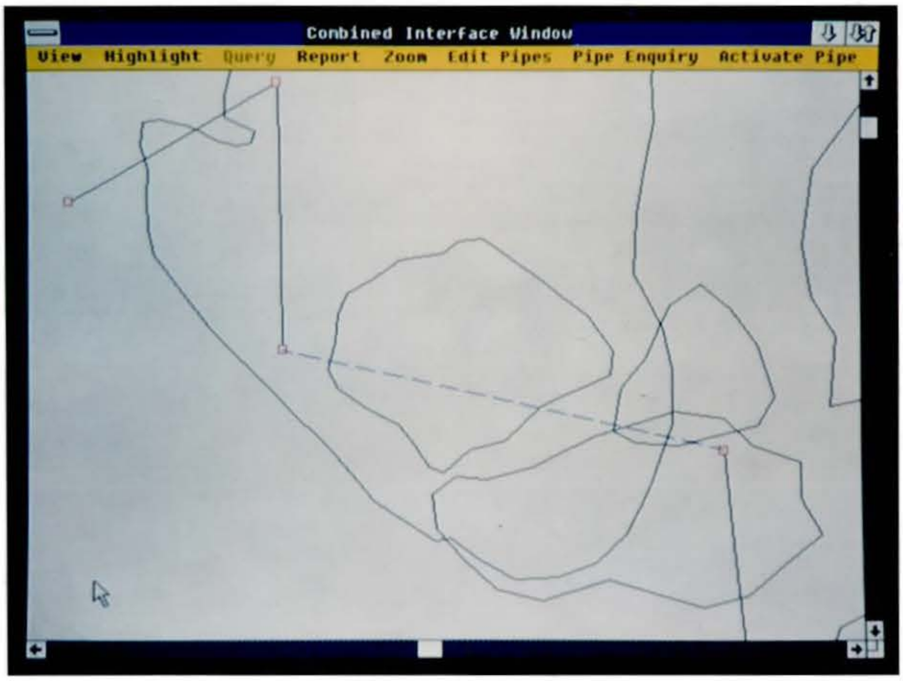

Combined Interface Window
View Highlight Query Report Zoom Edit Pipes Pipe Enquiry Activate Pipe

Plate 8.2 THE TEST CASE SEGMENT FOR FEATURE ANALYSIS

A long section of any pipeline, or any part of a pipeline, may be requested from the GUI by first activating the pipe, then asking for the long section. PIRATE opens a separate 'Long Section Window' to display the image. The long section for the segment in Plate 8.3 is given in Plate 8.4, page 172. At present the long section scales are derived automatically by PIRATE, but the system could easily be modified to give the user more control.

More important than the display itself is the fact that long section information is actually stored within the frame hierarchy, and is therefore available for further analysis by the knowledge base. Although not currently implemented, applications would include slope stability, stress and span analyses, and judgements on the effects of each on the feasibility or cost of the route. Dredging in shallow regions is an example of an operation which can be expensive and difficult to optimise. With long section information PIRATE could be upgraded to advise on dredging requirements and costs.

8.4.5 Clash Analysis and Reporting

PIRATE is able to give the user reports on the features that any pipeline segment traverses, and the exact chainages and distances of each crossing. To obtain this report the user picks the 'Report' menu option and chooses the 'Segment Clash Report' item from it. The end points of the chosen segment must then be probed using the mouse pointer. Figure 8.3 shows the segment used for testing the clash and technical reports. It is the most fitting of the test segments as it passes through the most complex arrangement of features in the study region. It will be referred to as SEGMENT-1. Requesting clash analysis produces a report with chainages summarised in Table 8.1, page 175.

The figures in Table 8.1 are in kilometres. Note that the diagram, Figure 8.2. is not to scale, but clearly correlates with the ordering and overlaps of the features indicated by the table.

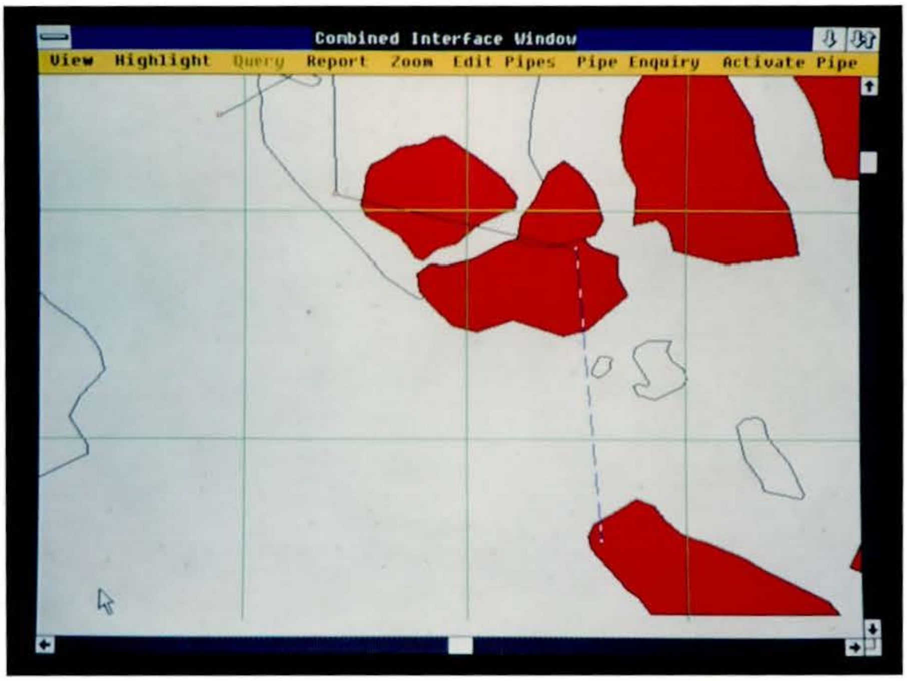


Plate 8.3 CHART SHOWING THE TEST SEGMENT FOR LONG SECTION ANALYSIS

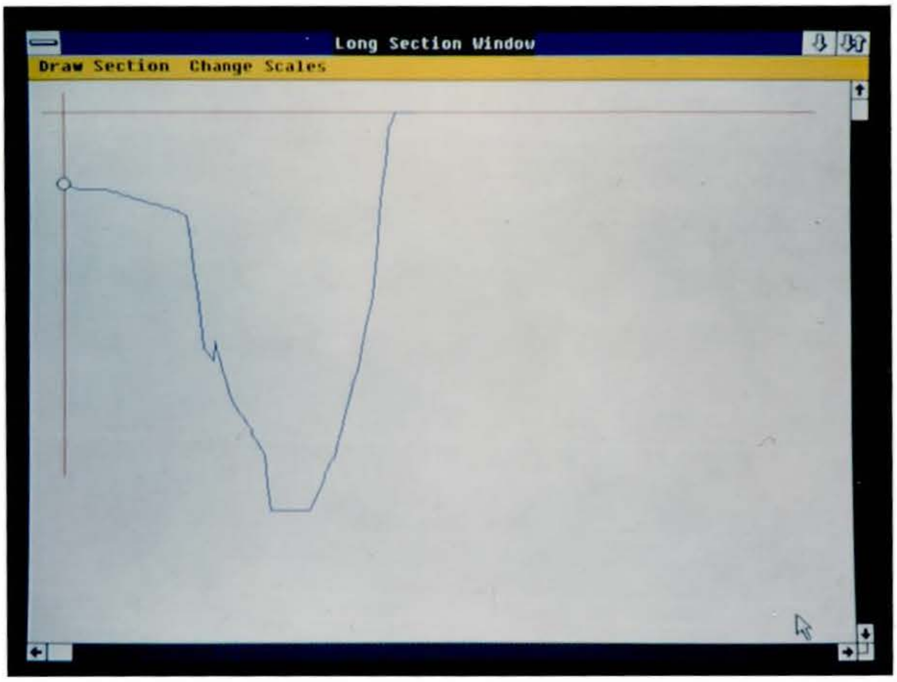


Plate 8.4 LONG SECTION OF BATHYMETRY ALONG THE PIPE SEGMENT SHOWN IN PLATE 8.3

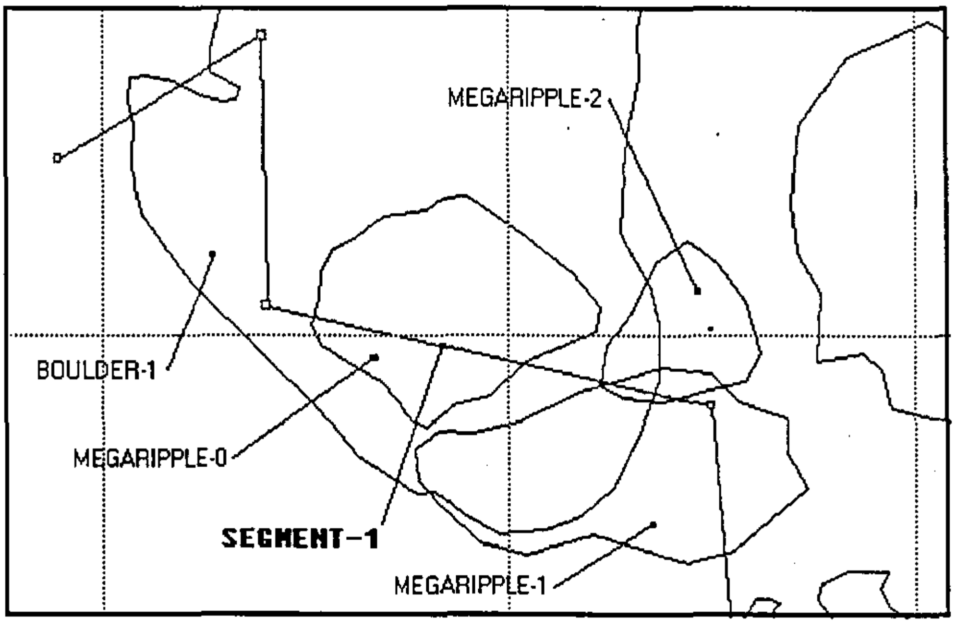


Figure 8.3 A Chart Showing the Location of SEGMENT-1 With Respect to Constraint Features

8.4.6 Technical Analysis and Reporting

Producing clash chainages is essentially a geometric calculation process requiring no expertise. Judging the effects each feature will have on the pipeline, however, does require expertise. The technical report provided by PIRATE uses the expert rules in the knowledge base to find the type and severity of remedial actions needed to safely install the pipe. When more than one feature is present along any part of the pipe the resulting remedial actions may conflict, for example if one feature needs a 0.6 metre deep trench and another needs a 1.2 metre deep trench. PIRATE uses heuristics, or 'rules of thumb' to take the most severe recommendation in these cases. The system also ensures that duplicate recommendations are not made.

The rules used for the case study are given in Appendix E, and are briefly described in Section 8.3.2.

SEGMENT-1 was used to test PIRATE's technical reporting capability. The results naturally depend on the properties of the features crossed and the initial design parameters chosen, so these are given in Tables 8.2, 8.3 and 8.4.

Two types of remedial action, trenching and diver support, can be recommended by the example rule base. The part of the technical report related to trenching is the best to illustrate the knowledge base in use.

Table 8.5, page 177, gives the trenching recommendations for SEGMENT-1. Plate 7.12, page 148, shows how this information is given to the user at the GUI. The following paragraph explains how the results were derived. Chainages along the pipeline are given in brackets, these being taken from Table 8.1. The diagram in Figure 8.3, page 173, will also help the reader understand the explanation.

As one proceeds from West to East along SEGMENT-1, the pipe is first crossing BOULDER-1 alone (from 0.0 km). This requires trenching to the minimum cover depth. The trench depth is thus the minimum cover depth (0.3m) plus the pipe diameter (0.33m), totalling 0.63 metres. Farther along MEGARIPPLE-0 is also entered (at 1.9 km). This requires additional trench depth to allow for a wave height of 0.9m, giving a total depth of 1.53 metres. Both the 0.63 metre and the 1.53 metre trench are conflicting over this region, and the severity heuristic therefore only reports the most severe.

FEATURE	START. (km)	FINISH (km)	DISTANCE. (km)
MEGARIPPLE-2	14.4	18.3	3.9
BOULDER-1	0.0	16.7	16.7
MEGARIPPLE-1	14.9	19.1	4.2
MEGARIPPLE-0	1.9	11.4	9.5

Table 8.1 THE RESULTS OF CLASH ANALYSIS ON SEGMENT-1

FEATURE	WAVE DIRECTION (Degrees)	WAVE HEIGHT (metres)	WAVE LENGTH (metres)
MEGARIPPLE-0	300	0.9	15.0
MEGARIPPLE-1	300	1.3	25.0
MEGARIPPLE-2	270	0.5	15.0

Table 8.2 MEGARIPPLE FIELD PROPERTIES FOR FEATURES OCCURRING ON SEGMENT-1

FEATURE	DENSITY OF BOULDERS (No PER 100m^2)
BOULDER-1	0.9

Table 8.3 BOULDER FIELD PROPERTIES FOR FEATURES OCCURRING ON SEGMENT-1

PARAMETER	VALUE
PIPE DIAMETER	0.33m
NATIONAL JURISDICTION	UNITED KINGDOM
CODE OF PRACTICE	BRITISH
MINIMUM COVER	0.3m

Table 8.4 DESIGN PARAMETERS CHOSEN FOR THE TEST ROUTE

START (km)	FINISH (km)	MAX. DEPTH (metres)	NO. OF PASSES
0.0	1.9	0.63	1
1.9	11.4	1.53	2
11.4	14.4	0.63	1
14.4	14.9	1.13	1
14.9	19.1	1.93	2

TABLE 8.5 RESULTS OF THE TECHNICAL ANALYSIS FOR TRENCHING (PIPE DIAMETER = 0.33 M)

START (km)	FINISH (km)	MAX. DEPTH (metres)	NO. OF PASSES
0.0	1.9	1.3	1
1.9	11.4	2.2	2
11.4	14.4	1.3	1
14.4	14.9	1.8	2
14.9	19.1	2.6	2

Table 8.6 RESULTS OF THE TECHNICAL ANALYSIS FOR TRENCHING (PIPE DIAMETER = 0.9 M)

The pipe then reverts to just BOULDER-1 (at 11.4 km), before entering MEGARIPPLE-2 (at 14.4 km). MEGARIPPLE-1 is also entered soon after (at 14.9 km). At this point three features co-exist, MEGARIPPLE-1 requiring the most severe trench depth, at 1.93 metres.

Although as one continues to the end of the pipe the features change, with first BOULDER-1 (at 16.7 km), then MEGARIPPLE-2 (at 18.3 km) disappearing from the pipe clash report, the trenching recommendation remains at 1.93m. This is because throughout the region MEGARIPPLE-1 continues to dominate the trenching depth.

The "number of passes" column in Table 8.5, page 177, refers to the number of times the trenching vessel needs to follow the line of the pipe to achieve the required trench depth. Trenching vessel information is contained in the frame hierarchy having been pulled in from Lotus spreadsheets. Trenching rules refer to the instance of the chosen trenching vessel to discover the maximum trench depth per pass. The number of passes needed is then calculated.

The technical report example illustrates how PIRATE can produce an overall plan of action for any segment, given multiple, possibly conflicting features and remedial actions. Extending this for entire pipeline routes would involve the amalgamation of segment results. The example also shows how other data, such as construction plant specifications, can be included in the judgements. A more comprehensive knowledge base could use the length of the required trench and the number of passes needed, together with the overall plant output rate and day cost, to produce an overall price for the trenching in each section. By amalgamating these with the costs of other remedial actions, such as diver support, blasting, dredging etc., and material, start up and other costs, PIRATE could generate an overall preliminary cost for the entire pipeline.

8.4.7 The Base Case Route in PIRATE

The SEGMENT-1 example in the previous sections proved that PIRATE works in areas where there are multiple overlapping features. It did not prove that the system could cope with the number of pipeline segments required to adequately depict a real pipeline.

The Base Case Route, a 263 kilometre route originally designed by SGSL, was digitised from the constraint chart, Figure 8.1. This pipeline route is typical of those on the chart. The intention was to use the Base Case Route to test whether the system could cope with the volume of data, resulting instances and analysis for a real pipeline, yet still maintain an acceptable response time. The Base Case Route was not used for the earlier tests because it does not pass through a complex area of interacting features.

The digitised Base Case Route data consisted of 44 straight line segments, adequate to approximate the curves on the route. The data was read by PIRATE GIS functions, and clash analysis performed. The analysis took 45 seconds, producing a clash result table with 50 records and a long section table with 1323 records. GoldWorksII was then entered, with PIRATE initalisation taking a total of 16 minutes to load all the pipeline and results data.

Once within GoldWorksII the GUI response times were not significantly different to those found in other tests. It was noticed, however, that the Lisp environment stopped more often to 'garbage collect' (where it frees memory that is no longer useful). This is assumed to be because of increase in the volume of data in the frame hierarchy. Should this become a significant problem two options could be taken. First, the computer memory could be increased in size. Alternatively the long section coordinate data, currently stored in each pipe segment instance as a huge list, could be allocated a relational table where it could be read when a long section request is made at the GUI. This would reduce the memory requirement substantially.

Conclusions from the Base Case test were, apart from the increased 'garbage collecting', favourable. The system handled a real pipeline route in a competent manner.

8.5 Data Modification and Explanation Facilities

8.5.1 A Geographic Design Spreadsheet

The Sable Island Pipeline Project is convincing proof that changes in design parameters *are* made at a very late stage in design. The new design to cater for the collection of gas from the Thebaud field is an extreme example of the amount of extra work such changes can mean when using manual methods. Less dramatic modifications, changing the pipe diameter for example, can also have a marked effect on the design.

Section 7.5.6 explained how PIRATE uses logical dependency to ensure that the facts in the system are always valid. This is of great benefit when the user decides to change any design parameters, or indeed any fact in the system. The user can use the system to ask "What if .....?" questions by changing information in any part of the system and observing the results.

When a fact in the system is changed all facts that are dependent on the modified fact are deleted. When the user asks for a new report the updated facts are found using rules. Changing the pipe diameter, for example, becomes a simple matter. All that the user needs to do is to display the DESIGN_PARAMETERS instance where the pipeline diameter is stored, then using the mouse and keyboard change the value in the diameter slot. Everything dependent on the pipe diameter is deleted, and requesting a report ensures that backward chaining finds new results using the modified diameter. This is, of course, greatly different to any similar assessment in the case of manual design.

The dependency structure has been verified using the SEGMENT-1 test segment. The original trenching results given in Table 8.5, page 177, were for a 0.33 metre diameter pipeline. This was changed to 0.9 metres to test the effect. PIRATE automatically retracted the technical report and all facts that depended on the pipe diameter, leaving the slots empty.

When a new technical report was requested the system backward chained to produce a new report, the results of which are given in Table 8.6, page 177. Note that the trench depths have not only increased by the difference in diameter, but the Code of Practice rules have sensed the diameter moving above a threshold value to a position where an increased minimum cover (0.1m extra) is also needed. This has therefore been incorporated into the results. The number of passes needed by the trenching vessel has also been recalculated for each entry.

Other examples of "What if....?" questioning that have been tested include changing the Code of Practice used for the pipeline design, and changing the type of trenching vessel used during pipeline construction. Rules from four different Codes of Practice are in the PIRATE rule base. Changing the Code from British to Norwegian, for example, has an immediate effect on all results as the initial design parameters are changed by the system. Similarly, changing the trenching vessel specification alters the maximum depth that can be excavated in a single pass of the trencher. PIRATE automatically alters the results in line with this change.

The dependency structure in PIRATE clearly shows how the system can be used as a *geographic design spreadsheet*, with the engineer able to experiment with design parameters to achieve an optimum solution. If the knowledge base were more extensive the logical dependency mechanism would extend right through to the cost analyses, giving instant updates as initial parameters are varied. Such flexibility is a key attraction of PIRATE as a design environment.

8.5.2 Moving and Re-Assessing Routes

The logical dependency structure of PIRATE extends to the spatial positioning of routes. Section 8.4.2 details how pipeline segments can be added, moved or deleted in the pipe segment network. Any change in pipeline segment position obviously affects the feature crossings that the pipeline experiences. PIRATE ensures that if a pipe segment is moved or deleted, all facts associated with the pipe segment are retracted. Also, if the pipe segment is deleted all PROPOSED_PIPELINE instances that contain it are also deleted. PIRATE automatically takes care of all integrity maintenance. The user is free to experiment with his design as he wishes. PIRATE naturally warns the user should his proposed action cause the deletion of any major parts of the pipeline design.

When routes are moved, added or deleted, they have to be re-assessed by the PIRATE GIS clash functions before work can proceed. Having changed the pipe network the user must re-activate his chosen route(s), which are sent for analysis from the GUI. The user may then request reports as usual, with PIRATE ensuring that the new assessment results are used.

Tests employing re-routing and pipeline deletion proved that the system was functioning satisfactorily in this respect.

8.5.3 Explanation Facilities

As the complexity of the knowledge base increases the logic and rules PIRATE uses to achieve results may not always be clear. One of the quoted advantages of expert systems is that they can explain their reasoning. PIRATE is no exception. The user can ask the system to explain any fact that it holds, PIRATE responding with the line of reasoning that lead to this fact.

Plate 7.13, page 148, gives an example of PIRATE explanation using the in-built GoldWorksII explanation facility. PIRATE was asked to explain why it had recommended trenching as a remedial action for SEGMENT-1 traversing MEGARIPPLE-1 in the test case. The response details the rules and justifying facts which lead to the trenching recommendation. The explanation is not very readable, a fault of the GoldWorksII toolkit, but by including explanatory comments in each rule definition the text becomes easier to understand. Plate 7.13 does not give the explanations clearly as the window scroll bars need to be used to scroll the rest of the text into view. Full comments can be seen incorporated into the rules listed in Appendix E.

8.6 Extending PIRATE Capabilities

As a prototype PIRATE has a large enough knowledge base and enough functionality to prove that the concepts proposed by this thesis are valid. Extending PIRATE beyond this must also be discussed. PIRATE has been designed from first principles to cater for problems involving large numbers of facts, rules and different types of geographic feature. The case study has shown the system working with two types of feature, a relatively small knowledge base and two major types of report. A practical application of PIRATE would require more than this.

8.6.1 Adding More Rules

Rules are divided into two types in the knowledge base, forward and backward chaining. The forward chaining rules perform the control functions of the system, such as reading databases and creating instances on initialisation. Backward chaining rules are used for judgements about pipeline routes. Forward chaining rules are an intrinsic part of PIRATE and should not be altered. Backward chaining rules can be added to and changed at will by the user as they contain purely knowledge about pipeline design. Future users can continue to develop the knowledge base by using backward chaining rules.

It has been noted by previous users of GoldWorks, that the system response time slows considerably as the number of rules in the knowledge base builds up (Allwood, 1989b). Currently there are relatively few rules in the PIRATE knowledge base, but substantial rule additions are expected to slow the production of technical and other reports. However, the GUI has been kept independent of the rule mechanism, and so the normal interactive dialogue with the user should not suffer because of an increasing knowledge base size.

One fault in the GoldWorksII inference engine has been in its failure to ask users for values which it cannot find or infer. This can be an important aspect of the behaviour of an 'intelligent' system, as it should be able to recognise when it needs more information and respond to this by asking the user for it. If no Code of Practice was chosen in PIRATE, for example, the system should ask for it to be specified. GoldWorksII was supposed to provide this facility using a 'query form facet'. However, the author found that it would not work, and this has since been confirmed as a GoldWorksII system bug which has yet to be fixed (AI Ltd, 1989). Future developers of PIRATE will have to work around this problem until a software solution is found.

8.6.2 Adding More Feature Types

A variety of feature classes are provided in the frame hierarchy, Plate 7.1, page 130. To use them each must have a property table in the PIRATE GIS. Fields represent individual feature properties, records represent individual feature occurrences, (Section 7.3.1). Each table must also have a feature code field.

In GoldWorksII each new feature class must be given a frame and a base instance to indicate that it is active, (Section 7.5.1). The frame must have the GIS feature property table filename in its DATABASE_NAME slot. PIRATE uses forward chaining rules to match on every base instance during initialisation, opening the associated GIS property databases and creating instances from the records under the appropriate frames.

It is expected in future versions of PIRATE that software procedures would be written in Lisp to automatically perform all the steps in new feature class creation. The user would then be free to develop a library of feature classes for use as and when they become appropriate.

8.6.3 Adding More Report Types

Reports are requested from menus at the GUI, and operate via an instance of the REPORT-CONTROLLER frame (Section 7.5.5). Slots exist in this for a number of different reports, with daemons overlooking each to sense if the a value changes to REQUIRED.

Daemons are currently written for pipe segment clash and technical reports. To create more reports new daemons need to be written following the format of the existing ones. The report daemons are Lisp functions which create output windows, generate backward chaining queries, and control the report output format.

To write the new daemons a working knowledge of Lisp and the PIRATE structure are needed. In future versions the author would recommend the creation of a 'report toolkit' in Lisp, giving the user a library of high level function which can be 'bolted together' to create reports on the subjects he needs, in the style he desires. Using the windows functions employed for the GUI, the toolkit could be made suitable for a non-Lisp programmer to use.

8.7 SUMMARY

To assess the practicalities of intelligent GIS for design, PIRATE was tested using data from a real pipeline project. The Sable Island Pipeline Project was used as the basis for the tests. Full documentation and sixteen constraint charts were provided by J.P.

Kenny & Partners (JPK). The manual design approach taken by JPK for this project was first studied in detail.

The PIRATE tests were based on the use of one of the sixteen constraint charts. All occurrences of two geographic feature types were digitised, together with the bathymetric contours for the region. It is acknowledged that this is a relatively small subset of all the information available for pipeline design. However, a full emulation of the manual design results was not an aim of the tests. PIRATE is in a prototype form and does not have sufficient rules in it's knowledge base to cope with a greater diversity of feature types. In terms of testing the structure and concept of PIRATE, the data volumes used were more than adequate.

PIRATE facilities were examined as the digitised data was input and processed into the GIS raster form. Then PIRATE within GoldWorksII was initialised. General GUI functions of map display, overlay, GIS interrogation, pipeline placement and editing all functioned correctly. Pipeline assessment by the GIS was rapid considering the amount of database access involved, and the production of long section profiles from the PIRATE DTM was also shown to function satisfactorily.

The results of pipeline analysis were studied in some depth to ensure that PIRATE was producing correct results. This examination was achieved by assessing a single pipeline segment traversing the most complex area of overlapping features in the case study region. The numeric results were shown to be correct, and the technical assessments made by the PIRATE knowledge base were also shown to be valid. The validity of the case study results may be legitimately extrapolated to cover much more complex pipelines and feature interactions. The reason for this is the nature of the Lisp implementation and the recursive functions used in the assessment. In proving that a recursive function works in a small but representative problem the proof can be extended to larger problems of the same type.

To test the capacity of PIRATE in handling real pipelines, a real 263 kilometre route from the Sable Island Project was digitised and assessed. The GIS feature clash assessment took 45 seconds, and thereafter there appeared to be little degradation in response time at the GUI. However, it was noted that there was a significant increase in the frequency of 'garbage collection' by the Lisp environment. This usually indicates that the computer memory is becoming more replete. Clearly, if the amount of data held within GoldWorksII became too great the system would fail. However, this is a function of the capacity of the hardware rather than the design of the software. It was noted, however, that if the long section profiles were stored in relational database tables rather than in the frame hierarchy, a significant proportion of the memory loading would be avoided. It is also expected that, due to the operating characteristics of GoldWorksII, as the number of rules increases the time taken to assess pipelines using the knowledge base will also increase.

PIRATE was further tested in it's role as a geographic design spreadsheet. The pipe diameter was changed for a previously assessed pipeline. New reports were requested which, when compared with the old, showed that the correct changes had indeed been made to the results of the analyses. The explanation facilities, where PIRATE informs the user of the reasoning behind any decision it makes, was also tested and shown to be working correctly. However, the vocabulary and prose of the report produced is currently rather poor.

The PIRATE system was specifically designed to allow for further expansion when necessary. The final part of the Chapter discussed briefly how such an expansion could be achieved.

CHAPTER 9

9. CONCLUSIONS AND RECOMMENDATIONS

9.1 CONCLUSIONS

Engineering design relies heavily on the use of geographic and spatial information. However, GIS have almost universally failed to make an impact on the discipline, or to become a worthwhile design tool. This is because conventional GIS cannot understand the information they hold in the context of the design problem, and cannot make an intelligent contribution to the solution being sought.

The Pipeline Route Analysis and Testing Environment (PIRATE) is a system developed during course of this research which has proven that it is possible to create an *intelligent* GIS for design. PIRATE has been applied to the specific problem of off-shore pipeline route design. Unlike conventional GIS the system *can* understand the information it holds, and *can* make a positive, intelligent contribution to the design. Moreover, by creating a practical, working system during the course of the research, the concept has not only been proven possible, but has also been proven *practically viable* using current software and hardware tools.

The PIRATE architecture shows how the widely dissimilar technologies of artificial intelligence (AI) and GIS can be brought together into a harmonious structure, where the advantages of both are optimised. The key concept lies in integrating the two components so closely that they begin to merge. The crux of this integration lies in the use of *spatial dualism*, where the location based (spatial) description of geographic features is stored in the GIS, and the object based (non-spatial) description is stored in the frame hierarchy of an AI toolkit. As a result of this integration it is now possible to write a knowledge base of rules which has full access to, and complete control over, all geographic information in the GIS database. This is believed to be the first time such an integrated structure has been used with a formal AI toolkit.

During the design and implementation of PIRATE, practicing pipeline engineers were consulted extensively to ensure that the system would be applicable and effective. The engineers stated that they did not want a computer system that was a 'black box', but preferred to retain control of the routing itself. They also required a simple user interface and needed to avoid direct contact with the complexities of an AI system. The PIRATE graphic user interface has been successful in masking the AI component of the system during pipeline route design. It also has the 'look and feel' of a conventional CAD system, which is more familiar to the engineer than an AI toolkit interface. The AI system operates in the background, assessing routes and providing reports. This minimises the exposure of the engineer to system complexities.

PIRATE was tested using data from a real pipeline design project. Pipeline routes were proposed and the system was asked to provide design assessments for them. Whilst not a full emulation of the manual design, the tests proved that PIRATE could handle a substantial amount of geographic information, allow the interactive design of pipeline routes, and automatically produce correct assessments of those routes. Rules in the PIRATE knowledge base were a sub-set of those gained from interviews with practicing pipeline engineers, and referred to the remedial actions that would be needed as a result of the pipeline crossing certain geographic features.

Aside from making rapid route assessment possible, the most formidable impact PIRATE makes on the working practices of engineers is, perhaps, in it's ability to act as a *geographic design spreadsheet.* The engineer can change any design parameter at any time, the system will immediately incorporate those changes into the design to show the engineer the effect of his modifications on the final result. In this way engineers can rapidly experiment with their designs in the same way as accountants experiment with figures using conventional spreadsheets.

On a more general note, fully automated design using unconstrained geographic information was shown not to be possible until a formal language for the representation of spatial objects and relationships was defined. In the event fully automated spatial reasoning was not required for pipeline design, but in other applications this potential stumbling block should be bourne in mind.

Intelligent GIS for design, such as PIRATE, have only become possible with the advent of sophisticated AI toolkits, which have a wide range of AI paradigms, seamless interaction with relational databases and tools for building graphic interfaces. The increasing speed and functionality of AI toolkits as the technology develops should have a positive effect on future intelligent GIS.

There is a common belief prevailing that AI has been the "great white elephant" of computing technology in recent years. The rather modest successes of the Alvey initiative and of high failure rate of commercial applications of expert systems have fuelled this scepticism. However, this project has shown how AI, acting as background support to more conventional technology, can make a substantial and effective contribution in practical working environment.

In the rapidly advancing field of GIS, commercial developers are solving many of the current difficulties in the capture, storage and analysis of geographic information. Once these problems are successfully addressed, the future lies in making the best use of geographic information, in making it work as effectively as possible. This research has shown one way in which such effectiveness might be achieved.

9.2 RECOMMENDATIONS FOR FURTHER WORK

The PIRATE architecture, with it's close integration of AI and GIS, has implications beyond applications in design. The concepts illustrated could be used in any area where automated reasoning with geographic information needs to be explored. Indeed PIRATE itself, stripped of the rules and functions relating to pipeline route design, could be a vehicle for such an exploration. A less radical change, altering the design rules alone, would allow PIRATE to assist in the design of other types of route, electricity power cables for instance. In either case PIRATE has the potential to provide the basis for a *geographic expert system shell.* Using such a system the user could create his own knowledge base of rules relating to geographic features. The AI toolkit, which is an intrinsic part of the system, provides a wide range of AI paradigms which a user could employ in his knowledge base, in addition to the close AI/GIS integration. PIRATE has not been used in this way to date, but the basis for it is clear as a result of this work.

An area which is causing considerable difficulty is the formal definition of terms to describe spatial relationships between irregular geographic features. Terms such as 'direction' and 'distance' lose formal meaning when used with irregularly shaped objects, yet are critical for geographic data to be interpreted *semantically*. These terms would form the basis of a formal *language* for spatial representation. Such a language is the first step to fully automated spatial cognition, where an AI system could take a holistic view of geographic data and reason with it in a manner more akin to our own mental processes. In Section 4.3 the existing theories in the area were discussed. PIRATE itself provides an environment which could be used for further experimentation. This issue is crucial to future development, and it is hoped that someone will address it in the near future.

One final issue which merits continued attention is that of knowledge acquisition and the subsequent conversion of the acquired knowledge into a software representation. This research highlighted a some of the practical difficulties in the area, and solved a few of them. However, more formal research is needed as many of the projects that have contributed to the discipline so far have taken a 'trial and error' approach, rather than one based on a sound theoretical hypothesis which was subsequently proven true or false. For example, the author knows of no published material which deals with the problems of knowledge acquisition in domains with a high spatial or geographic data content. This will become an increasingly acute problem as more AI systems are applied in disciplines such as GIS and engineering design.

GLOSSARY

GLOSSARY

Access Oriented Programming - a programming technique which uses daemons attached to frames to create a dynamic program which responds when the data in the frame hierarchy changes.

Agenda - in forward chaining only - a list of rules that could possibly fire given the current state of the facts base. The list is usually ordered according to some priority, and when complete the top rule is fired. If this causes changes to the facts in the facts base the rest of the agenda will become invalid and must be re-compiled

AI Toolkit - A programming and a user environment where a variety of different AI techniques are available. Usually based on a core AI language like Lisp, and having a variety of interfaces and programming and debugging aids.

Assert - verb: assert - to state that a fact is true, to place a fact in the facts base.- noun: assertion - a fact that has been placed in the facts base

Backward Chaining - the opposite to forward chaining - where an individual fact, or goal, needs to be proven. The THEN parts of rules are checked to see if they could possibly prove the goal, if so the IF parts are checked and any items issued as sub-goals if not found in the facts base. the system propagates until either a goal is proven, or is unproven. This method uses minimal processing to find individual facts as they are needed. cf forward chaining.

Bi-directional Reasoning - a method by which both forward and backward chaining are used, with the intention of gaining from the advantages of each

Daemon - a procedural function which is attached to the slot of a frame, and is executed when the value in the slot is modified or accessed.

Expert - in knowledge engineering - also *domain expert* - some person who has expertise in the subject that an expert system is intended to help with.

Expert System - a knowledge based system which has been built specifically to model the expertise of a real expert in a 'real world' problem - a practical application of a knowledge based system

Expert System Shell - An expert system with nothing in the knowledge base. Often sold as a software package for the user to fill the knowledge base with his own rules - a quick way to write an expert system.

Facts Base - an occasionally used term - refers to the categoric facts that are known to an AI system, as opposed to the rule base which is primarily using the facts in the facts base to infer new information.

Forward Chaining - where rules in a knowledge base are used in a way that checks the IF parts of the rules. If any are satisfied it fires the rule to infer new facts. The method ensures that *all possible* facts are discovered given a set of known facts and rules

Frame - A collection of attributes or properties which describe a type or class of objects. Frames can build up into a hierarchy, like a family tree, with child frames inheriting the attributes of parents. Frames do *not* represent individual objects - see instance

GoldWorks - A commercial AI toolkit built around a Lisp language core. The second version, GoldWorksII, is built within the Microsoft Windows environment and provides a full graphic user interface and facilities for the user to create his own GUI

Graphic User Interface (GUI) - An interface to the user that has a graphic capability, using a 'mouse' and menu operation. In some cases the GUI will also display detailed graphic data, such as maps and drawings.

Inference - verb : inferencing - to use a rule to infer, or state the existence of, a fact because of the existence of other facts. - noun: inference - a fact resulting from inferencing.

Inference Engine - The program which uses rules to infer facts. It is classed according to the inference mechanism it uses, such as forward or backward chaining.

Instance - The representation of an individual object which is a child of a frame. Instances hold slot values describing the object.

Knowledge - information which is generally used to yield new facts given existing facts.

Knowledge Base - a collection of pieces of knowledge, usually in software form as part of a knowledge based or expert system.

Knowledge Based System (KBS) - an AI system where the knowledge element of the program has been explicitly separated from the rest of the system, such that it can be updated without making changes to the rest of the program. KBS are also said to have an inference engine, though this is not necessarily discrete

Lisp - A language for LISt Processing. The core of the GoldWorks AI Toolkit and a popular AI language.

Lisp List - a sequence, or array, of elements which can be numbers or symbols, and which are enclosed in brackets. The length of the list is not previously defined and cannot be defined in advance. For example, (CAT DOG TROUSERS 3) is a list of elements - The basis of the Lisp language, in which all program code consists of lists, and lisp functions operate on data contained in lists. Lisp can thus operate on its own program code because of the identical structure.

Message - in object oriented programming - a call which is received by an object telling a method to execute, or fire.

Method - in object oriented programming - a method is a piece of procedural program, a function, which is attached to an object and executes when it receives a message to do so.

Object - A term to describe an individual real world entity, or an abstract idea which can be described by its attributes. - in object oriented programming - a collection of property values describing a real world entity or abstract entity, together with methods which describe its behaviour. Objects can be arranged into an inheritance hierarchy like frames.

Object Oriented Programming - a programming technique by which collections of objects and methods are defined, control being effected by the passing of messages between objects - many graphic user interfaces are written using object oriented techniques

Procedure - a sequence of steps in a program which always occur one after another.

Recursion - a programming method where a function calls itself within itself. Used for dealing with data of unknown length and trees of data with unknown depth. A recursive function must contain a cut off condition, otherwise it will continue nesting into itself forever. An analogy is the visual effect of looking into a mirror whilst holding another mirror, aligned so that you see yourself in the mirror you are holding. The resulting 'tunnel' of mirror images is the function of the mirror recursing, and is theoretically infinite.

Retract - verb: retract - to negate an assertion, to state that a fact is no longer true, to remove a fact from the facts base. - noun: retraction - the completed act of retracting, the fact that has been retracted.

Rule Set - a collection of rules which are grouped together. - GoldWorks specific - a set of rules which can be excluded from chaining until needed, and can be used to implement bidirectional reasoning.

Slot - A part of a frame. Slots are spaces for properties which describe the object class represented by the frame.

Slot Value - The value held within a slot, the actual property value of an object in an instance or frame

Sponsor - GoldWorks specific - A way of grouping forward chaining rules. Each receives its own agenda. Sponsors exist in a parent/child hierarchy, though there is no inheritance, and can be turned on or off, which includes or excludes the rules within it in the chaining process.

Symbol - A word which within a computer program refers to a real world object or abstract idea, for example DOG or BRIGHTNESS. The user, when he sees the symbol, equates it with the real world object that it represents.

Window - in graphic user interfaces - an area of the screen dedicated to the display of a particular item or process. It can usually be moved and re-shaped by the user, and more than one window can exist on a computer screen at any one time.

REFERENCES

REFERENCES

AI LTD, (1989): "GoldWorksII - Summary of Outstanding Bugs", AI Limited, Greycaines Road, Watford, England, October.

AIAI (1987a): "Introduction to Inference ART", Short Course Notes, Artificial Intelligence Applications Institute, 80 South Bridge, Edinburgh, 26-27 April, 1987.

AIAI (1987b): "Introduction to KEE", Short Course Notes, Artificial Intelligence Applications Institute, 80 South Bridge, Edinburgh, 3-4 May, 1987.

AIAI (1987c): "AI Programming Paradigms", Short Course Notes, Artificial Intelligence Applications Institute, 80 South Bridge, Edinburgh, March, 1987.

AiC (1990) "ProDIGIT User Manual", Applications in CADD Ltd., 21 Britannia Street, Shepshed, Leicestershire.

ALBERT, T.M. (1988): "Knowledge-Based Geographic Information Systems (KBGIS): New Analytic and Data Management Tools", Mathematical Geology, Vol 20, No 8.

ALLWOOD, R.J. (1989a): "Techniques And Applications Of Expert Systems in the Construction Industry", Ellis Horwood Ltd, (Pub.), ISBN 0 7458 0538 8.

ALLWOOD, R.J. (1989b): Personal Communication on GoldWorks Applications.

ANJEWIERDEN, A. (1987): "The KADS System", in Memorandum 92 of the VF-project "Knowledge Acquisition in Formal Domains", University of Amsterdam, Department of Social Science Informatics, Herengracht 196, 1016 BS Amsterdam, Netherlands.

ASHTON TATE, (1986): "Learning and Using dBase III Plus", Copyright Ashton Tate.

AUTODESK, (1988): "AutoCAD Reference Manual", Copyright AutoDesk Ltd.

BAGOT, K.H. (1985): "Digital Processing of Remote Sensing Data", in "Remote Sensing in Civil Engineering", Kennie, T.J.M. & Matthews, M.C. (Eds.), Halsted Press (Pub.), ISBN 0-903384-48-5.

BARNSLEY, M.F. & SLOAN, A.D. (1988): "A Better Way To Compress Images", BYTE, Vol 13, No1, January.

BILJON, W.van. (1987): "A Geographical Database System", AUTOCARTO 8, Proceedings of the Eighth International Symposium on Computer Assisted Cartography, Baltimore, 29th March - 3rd April.

BOWEN, J.A., CORNICK, T.C. & BULL, S.P. (1986): "BERT - An Expert System for Brickwork Design", Proceedings of Expert Systems 86, British Computer Society, 15th to 18th December.

BREUKER, J. & WIELINGA, B. (1987): "Knowledge Acquisition as Modelling Expertise: The KADS Methodology", in Memorandum 92 of the VF-project "Knowledge Acquisition in Formal Domains", University of Amsterdam, Department of Social Science Informatics, Herengracht 196, 1016 BS Amsterdam, Netherlands.

BREUKER, J. & WIELINGA, B. (1988): "Models of Expertise in Knowledge Acquisition", Memorandum 103 of the VF-project "Acquisition of Expertise", University of Amsterdam, Department of Social Science Informatics, Herengracht 196, 1016 BS Amsterdam, Netherlands.

BUCHANAN, B.G. & SHORTCLIFFE, E.H. (1984): "Rule Based Expert Systems: the MYCIN Experiments of the Stanford Heuristic Programming Project", Addison Wesley (Pub.), Reading, Mass.

BULLARD, R.K & DIXON-GOUGH, R.W. (1985): "Britain From Space - An Atlas of Landsat Images", Taylor & Francis.

BUNDOCK, M.S. (1987): "An Integrated DBMS Approach to Geographical Information Systems", AUTOCARTO 8, Proceedings of the Eighth International Symposium on Computer Assisted Cartography, Baltimore, 29th March - 3rd April.

BURROUGH, P.A. (1989); "Principles of Geographic Information Systems for Land Resources Assessment", Oxford Science Publications.

CALLEN, M., JAMES, I., MASON, D.C. & QUARMBY, M. (1986): "A Test Bed for Experiments on Hierarchical Data Models in Integrated Geographic Information Systems", in Spatial Data Processing using Tesseral Methods, Diaz & Bell (Eds.), NERC Unit for Thematic Information Systems, Reading. pp 193 - 212.

CAMPBELL, W.J. & GOETTSCHE, C. (1989): "Development of an Intelligent Interface For Adding Spatial Objects to a Knowledge-Based Geographic Information System", Proceedings of the 1989 Goddard Conference on Space Applications of Artificial Intelligence, April, Rash, J. (Ed.).

CHANDRA, N. & GORAN, W. (1986): "Steps Toward a Knowledge Based Geographical Data Analysis System", Geographic Information Systems in Government, Opitz, B.(Ed.), Hampton, Virginia, pp749-763.

CHARNIAK, E. & McDERMOTT, D. (1985): "Introduction to Artificial Intelligence", Addison Wesley Publishing Co.

CHEN, Z.T. (1984): "Quad Tree Spatial Spectra - Its Generation and Applications", Proceedings of the International Symposium on Spatial Data Handling, Zurich. Vol 1, pp208-237.

CHUNG, P.W.H. & KUMAR, B. (1987): "Knowledge Elicitation Methods: A Case Study in Structural Design", in Proceedings of CIVILCOMP, 1987.

CLOCKSIN, W. & MELLISH, C. (1987): "Programming in PROLOG (Third Edition)", Springer-verlag (Pub.).

CODD, E.F. (1970): "A Relational Model of Data for a Large Shared Data Bank", Communications of the ACM, Vol 13, No. 6, pp. 377-387.

CODD, E.F. (1971): "Further Normalisation of the Data Base Relational Model", in Proceedings of the Courant Computer Science Symposium, 6: Database Systems, New York.

COLBY, P.J. (1986): "PATCH: An Expert System for Advising on the Choice of Herbicides", Restricted Report, Project L 8002, LRS T 859, British Gas plc., London Research Station.

COLEMAN, J. (1989): "Pipeline Route Planning Using a GIS", Internal Report (unpublished), Department of Surveying, University of Newcastle, Newcastle Upon Tyne, NE1 7RU, England, September.

COUNCIL FOR SCIENCE AND SOCIETY, (1989): "Benefits and Risks of Knowledge Based Systems - Report of a Working Party", Oxford University Press (Pub.).

COYNE, R.D. & GERO, J.S. (1985a): "Design Knowledge and Sequential Plans", Environment and Planning B, Vol 12, pp401-418.

COYNE, R.D. & GERO, J.S. (1985b): "Design Knowledge and Context", Environment and Planning B, Vol 12, pp419-442.

COYNE, R.D. & GERO, J.S. (1986): "Semantics and the Organisation of Knowledge in Design", Design Computing, Vol. 1, pp68-89, John Wiley & Sons (Pub.).

DAVIES, E. (1986): "Representing and Acquiring Geographic Knowledge", Pitman (Pub.), London

DAYAL,U. & BUCHMANN,F. (1987): "Overview of PROBE: An Object Oriented Extensible Database System", AUTOCARTO 8, Proceedings of the Eighth International Symposium on Computer Assisted Cartography, Baltimore, 29th March - 3rd April.

DeCOLA, L. (1989): "Fractal Analysis of a Classified Landsat Scene", Photogrammetric Engineering and Remote Sensing, Vol 55, No 5, pp601-610, May.

DENHAM, C.M. HOLROYD, F.C. & JOHNSON, J.H. (1986): "Tessellation and Pixel Addressing Systems for Image Processing", in Spatial Data Processing using Tesseral Methods, Diaz & Bell (Eds.), NERC Unit for Thematic Information Systems, Reading. pp 87 - 98.

DEPARTMENT OF INDUSTRY, (1982): "A Programme for Advanced Information Technology : The Report of the Alvey Committee", HMSO.

DEPARTMENT OF THE ENVIRONMENT (DoE), (1987): "Handling Geographic Information", Report of the Committee of Enquiry chaired by Lord Chorley, Her Majesty's Stationary Office.

DEPARTMENT OF THE ENVIRONMENT (DoE), (1988): "Handling Geographic Information", The [UK] Government's Response to the Report of the Committee of Enquiry chaired by Lord Chorley, Her Majesty's Stationary Office, February.

Det NORSKE VERITAS, (1974): "Deepwater Pipelines", Norway's Official Reports, NOU 1974:40, Ministry of Industry, translated from Nowegian by Finaisanalyse A/S for the Gas Council (Exploration) Limited.

DIAZ, B.M. & BELL, S.B.M. (1986): "Spatial Data Processing using Tesseral Methods", NERC Unit for Thematic Information Systems, Reading, England.

DICKENS, J.G. & FINNIEAR, L.J. (1987): "Loughborough Benchmark Challenge - A Qualitative and Quantitative Assessment of Commercial Terrain Modelling Software", CADCAM 87, National Exhibition Centre, Birmingham, March.

DOBBS, V.S., DAVIS, H.W., & LIZZA, C. (1988): "An Application of Heuristic Search Techniques to the Problem of Flight Path Generation in a Military Hostile Environment", Proceedings of the 1st International Conference on Industrial and Engineering Applications of AI and Expert Systems, 1st-3rd June.

DURRANT, A.M (1988): "A Computer Aided Simulation Of Hydraulic Tailings Disposal", PhD. Thesis, University of Technology, Loughborough, Leicestershire, England.

EGENHOFER, M.J. & FRANK, A.U. (1990): "LOBSTER: Combining AI and Database Techniques for GIS", Photogrammetric Engineering and Remote Sensing, Vol 56, No 6, pp919-926, June.

ESTES, J.E., FRIEDL, M.A. & STAR, J.L. (1988): "Advanced Feature Extraction in Remote Sensing using Artificial Intelligence and Geographic Information Systems", Recent Advances in Sensors, Radiometry and Data Processing for Remote Sensing (Proceedings of the Meeting of the Society of Photo-Optical Instrumentation Engineers).

EWING, K.S. (1989): "Understanding Microsoft Windows", H.W. Sams (Pub.).

FEIGENBAUM, E.A. & McCORDUCK, P. (1984): "The Fifth Generation", Michael Joseph Ltd., London (Pub.) ISBN 0 7181 2401 4.

FINNIEAR, L.J. (1986a): "Computer Aided 3D Ground Modelling and Data Capture Systems", BSc. Project Report, Department of Civil Engineering, University of Technology, Loughborough, Leicestershire. April.

FINNIEAR, L.J. (1986b): "Scholarship Training Report", Internal Report, British Gas plc., Pipelines Division, Hinckley Operational Centre, August.

FINNIEAR, L.J. (1986c): "The Intergraph ETI, ICS and DTM Packages", Internal Report, British Gas plc., Pipelines Division, Hinckley Operational Centre, August.

FINNIEAR, L.J. (1987a): "An Analysis of Current Survey Data Capture and Transfer Techniques within British Gas, with Proposals For Improvement", Internal Report, British Gas plc., Construction Design Department, 7 High Holborn, London, 12th August.

FINNIEAR, L.J. (1987b): "The Loughborough Benchmark Challenge Phase 1 - A Summary of Major System Attributes", Proceedings of the Short Course in Terrain Modelling in Surveying and Civil Engineering, Glasgow University, 1st to 3rd September.

FINNIEAR, L.J. (1987c): "The Loughborough Benchmark Challenge Phase II - The Quantitative Test", Proceedings of the Short Course in Terrain Modelling in Surveying and Civil Engineering, Glasgow University, 1st to 3rd September.

FINNIEAR, L.J. (1987d): "Research Report", Department of Civil Engineering, University of Technology, Loughborough, Leicestershire, September.

FINNIEAR, L.J. (1988a): "Research Report - 2nd Year", Department of Civil Engineering, University of Technology, Loughborough, Leicestershire, September.

FINNIEAR, L.J. (1989): "Survey Working Practices for the 3D Environment", Internal Report (Restricted), Construction Design Department, British Gas plc., 7 High Holborn, London, 28th September.

FINNIEAR, L.J. (1990): "Routing Off-Shore Pipelines", in "AI for Engineers", Artificial Intelligence Applications Institute, Paul Chung (Ed.).

FINNIEAR, L.J. (1991): "An Intelligent Geographic Information System for Design", to be presented at Mapping Awareness '91, Olympia, London, 6th - 8th February.

FINNIEAR, L.J. DICKENS, J.G. & STRODACHS, J. (1988): "Artificial Intelligence in Terrain Modelling", Proceedings of the Third International Conference in Computing in Civil Engineering, University of British Columbia, Vancouver, Canada, 10th - 12th August.

FISHER, P.F. & MACKANESS, W.A. (1987): "Are Cartographic Expert Systems Possible ? ", AUTOCARTO 8, Proceedings of the Eighth International Symposium on Computer Assisted Cartography, Baltimore, 29th March - 3rd April.

FISHER, R. (1987): "Design of the IMAGINE II Scene Analysis Program", DAI Working Paper No 211, Department of Artificial Intelligence, University of Edinburgh, December.

FORBES, R. (1984): "Very High Level Relational Languages and GIS: Toward the Fifth Generation", Theodolite to Satellite, Technical Papers of the 51st Annual American Society of Photogrammetry Meeting, Washington DC., Vol. 1, pp414-420.

FRANK, A.U. (1982): "MAPQUERY: Data Base Query Language for Retrieval of Geometric Data and their Graphical Representation", Computer Graphics, Vol. 16, pp199-207.

FRANK, A.U. (1984): "Extending a Network Database with PROLOG", First International Workshop on Expert Database Systems, October, Kiawah Island, South Carolina.

FRANK, A.U. (1988): "Requirements for a Database Management System for a GIS", Photogrammetric Engineering and Remote Sensing Vol 54, No11, pp1557-1564, November.

FRANK, A.U., PALMER, B. & ROBINSON, V.B. (1986): "Formal Methods for Accurate Definition of some Fundamental Terms in Physical Geography", Second International Symposium on Spatial Data Handling, Seattle, Washington, pp583-599.

FRANKOT, R.T. & CHELLAPPA, R. (1990): "Estimation of Surface Topography from SAR Imagery Using Shape from Shading Techniques", Artificial Intelligence 43, pp271-310, Elsevier Science Publishers B.V.

FROST, R.A. (1986): "Introduction to Knowledge Base Systems", Collins Pub. ISBN 0-00-383114-0.

GANE, C. & SARSON, T. (1979): "Structured Systems Analysis: Tools and Techniques", Prentice Hall (Pub.)

GARGANTINI, I. (1982): "Linear Octrees for the Fast Processing of Three-Dimensional Objects", Computer Graphics & Image Processing, 20, pp264-269.

GOLDHILL (1987): "GoldWorks", The GoldWorks Version 1 Operating Manuals, 9 Volumes, GoldHill Computers Inc.

GOLDHILL (1989): "GoldWorksII", The GoldWorks Version 2 Operating Manuals, 16 Volumes, GoldHill Computers Inc.

GRADWELL, D.J.L. (1990): "AGI Standards Committee SQL Working Party : Analysis of Requirements for Database Software for Constructing GIS", Revision 4, Workshop on Standards for Handling Geographic Information, Nottingham, 24th September.

GREEN, S. (1986): "SPACES - A System for the Representation of Commonsense Knowledge About Space for Design", Proceedings of Expert Systems 86, British Computer Society, 15th to 18th December.

GREENWELL, M. (1988): "Knowledge Engineering for Expert Systems", Ellis Horwood (Pub.), ISBN 0-7458-0513-2.

GUDMANDSEN, P.E. (1983): "Application of Microwave Remote Sensing to Studies of Sea Ice", Philosophical Transactions of the Royal Society, London, A309, 433-445.

GUPTILL, S.C. (1987): "Desirable Characteristics of a Spatial Database Management System", AUTOCARTO 8, Proceedings of the Eighth International Symposium on Computer Assisted Cartography, Baltimore, 29th March - 3rd April.

HADIPRIONO, F.C., LYON, J.G. LI, T.W.H. & ARGAILAS, D.P. (1990): "The Development of a Knowledge-Based Expert System for Analysis of Drainage Patterns", Photogrammetric Engineering and Remote Sensing, Vol 56, No 6, pp905-909, June.

HART, A. (1986): "Knowledge Acquisition for Expert Systems", Kogan Page (Pub.), ISBN 1-85091-091-X

HAYES-ROTH, F. WATERMAN, D.A. & LENAT, D.B. (1983): "Building Expert Systems", Addison Wesley (pub.).

HEAD, G.O. (1987): "AutoLISP in Plain English", Ventana Press (Pub.), ISBN 0-940087-07-3.

HERRING, J.R. (1987): "TIGRIS: Topologically Integrated Geographic Information System", AUTOCARTO 8, Proceedings of the Eighth International Symposium on Computer Assisted Cartography, Baltimore, 29th March - 3rd April.

HERRING, J.R. (1990): "TIGRIS: A Data Model for an Object Oriented Geographic Information System", Proceedings of the Conference on GIS Models and Functionality, Midlands Regional Research Laboratory, University of Leicester, 21-22 March, 1990.

HESSION, W.C. & SHANHOLTZ, V.O. (1988): "A Geographic Information System for Targeting Nonpoint-source Agricultural Pollution", Journal of Soil and Water Conservation, Vol 43, No3, pp264-266.

HOGG, J. (1988); "Representing Spatial Data by Linear Quadtrees", Computing, 10th March.

HOLROYD, F.C. (1986): "Joining Tiles in Hierarchies: A Survey", in Spatial Data Processing using Tesseral Methods, Diaz & Bell (Eds.), NERC Unit for Thematic Information Systems, Reading, pp17-36.

HOSKEN, E. (1989): "Geographical Information Systems for Civil Engineering", Civil Engineering Research Newsletter, Science and Engineering Research Council & Department of the Environment, No9, September.

HOUSE OF LORDS (1983): "Report on Remote Sensing and Digital Mapping", Select Committee on Science and Technology, Report 98/L, HMSO, London.

HYDROCARBONS (Great Britain) LTD. (1986): "The Morecambe Phase II Feasibility Study", Volumes 1 to 3, Internal Reports.

INTELLICORP (1987a): "Technical Description of KEE Software", IntelliCorp Inc., 1975 El Camino Real West, Mountain View, California.

INTELLICORP (1987b): "KEEConnection - A Bridge Between Databases and Knowledge Bases", IntelliCorp Inc., 1975 El Camino Real West, Mountain View, California.

ISO9075, (1989): "ISO/IEC 9075:1989 - Information Processing Systems: Database Language SQL With Integrity Enhancement".

JACKENDOFF, R. (1983): "Semantics and Cognition", Cambridge, Mass, MIT Press (Pub.)

JACKSON,M.J. & MASON,D.C.(1986): "The Development of Integrated Geo-information Systems", International Journal of Remote Sensing, Vol7, No6, pp723-740.

JOHNSON, L. & KERAVNOU, E.T. (1988): "Expert System Architectures", Kogan Page Pub. ISBN 1-85091-475-3.

JP KENNY, (1984a): "East Bar Route Assessment", Internal Report, J.P. Kenny & Partners, Report No. 14 7 01 008 K 01, Project 2206.17, May.

JP KENNY, (1984b): "Engineering Review and Evaluation of Pipeline Landfalls", Internal Report, J.P. Kenny & Partners, Work Task 18, Report No. 14 3 01 017 K 02, Project 2206.18, October.

JP KENNY, (1985a): "Interactions of Bedforms and Pipelines", Internal Report, J.P. Kenny & Partners, Note No. 14 7 01 014 K 02, Project 2206.31, January.

JP KENNY, (1985b): "Offshore Pipeline Landfall and Route Assessment Study", Internal Report, J.P. Kenny & Partners, Work Task 32, Report No. 14 3 01 018 K 02, Project 2206.32, May.

JP KENNY, (1986): "Pipeline Route Assessment South of Sable Island", Internal Report, J.P. Kenny & Partners, Work Task 44, Report No. 14 3 01 020 K 03, Project 2206.44, May.

JP KENNY, (1989a): "Pickerill Field Development - Pipeline Route Survey Alignments", Internal Document, J.P. Kenny & Partners, Project No 3246, Dwg. No. 3246-CLX-DW-L-001, July.

JP KENNY, (1989b): "Pickerill Field Development - Preliminary Routing Study", Internal Report, J.P. Kenny & Partners, Project No. 3246, Report No. 3246-CLX-TN-L-001, November.

KAPETSKY, J.M. HILL, J.M. & DORSEY WORTHY, L. (1988): "A Geographical Information System for Catfish Farming Development", Aquaculture, Vol 68, No 4, pp311-320.

KARIMI, H.A., KRAKIWSKY, E.J., HARRIS, C., CRAIG, G. & GOSS, R. (1987): "A Relational Database Model for an AVL System and an Expert System for Optimal Route Selection", AUTOCARTO 8, Proceedings of the Eighth International Symposium on Computer Assisted Cartography, Baltimore, 29th March - 3rd April.

KAVORAS, M. (1985): "Design of a Geometry System to Handle 3-D Mining Information", Proceedings of the Sixth International Congress on Mining Surveying, Harrogate, England, September.

KEATING,T. PHILLIPS,W. & INGRAM,K. (1987): "An Integrated Topologic Database Design for Geographic Information Systems", Photogrammetric Engineering and Remote Sensing Vol53, No10, pp1399-1402.

KENNIE, T.J.M. & MATTHEWS, M.C. (1985): "Remote Sensing in Civil Engineering", Surrey University Press.

KUBO, S. (1986): "The Basic Scheme of TRINITY : A GIS With Intelligence", Proceedings of the Second International Symposium on Spatial Data Handling, Seattle, Washington.

LANE, C. (1990): "Spatial Units", presented to the Association for Geographic Information Workshop on Standards in GIS, Department of Geography, University of Nottingham, 24th September.

LASERSCAN, (1990): "HORIZON - GIS for the Environment", Mapping Awareness, Vol 4, No 7, September.

LESSARD, P. L'EPLATTENIER, R. et al (1988): "The Use of Geographical Information Systems in Estimating East Coast Fever Risk to African Livestock", Acta Veterinaria Scandinavica, Vol8, No84.

LONGLEY, D. & SHAIN, M. (1989): "Macmillan Dictionary of Information Technology", 3rd Edition, Macmillan Press (Pub.), ISBN 0-333-44971-1

LOOMER, S.A. (1986): "Computer Assisted Terrain Analysis on a Microcomputer", AUTOCARTO 8.

MAGUIRE, D.J. & RAPER, J. (1990): "Design Models and Functionality in GIS", Proceedings of the Conference on GIS Education and Training, University of Leicester, 20-21 March.

MANDLEBROT, B.B. (1983): "The Fractal Geometry of Nature", 3rd Edition, W.H. Freeman & Company.

MAPPING AWARENESS, (1989): "Trade Directory: GIS Systems and Supporting Software", Mapping Awareness, Vol 3, No 4, pp 43 - 47, September/October.

MARK, D.M. & GOODCHILD, M.F. (1986): "On the Ordering of Two-Dimensional Space: Introduction and Relation to Tesseral Principles", in Spatial Data Processing using Tesseral Methods, Diaz & Bell (Eds.), NERC Unit for Thematic Information Systems, Reading, pp179-192.

MARK,D.M. (1986): "The Use of Quadtrees in Geographic Information Systems and Spatial Data Handling", Proceedings of Auto Carto, London, Vol 1, pp. 517 - 526.

MASON, P.A. & AMOS, E.M. (1985): "Environmental Applications of Thermal Infrared Imagery", in Remote Sensing in Civil Engineering, Kennie, T.J.M. & Matthews, M.C. (Eds.), Surrey University Press.

MASON, R.W. (1989): "Data Conversion - Raster and Vector Formats", Proceedings of the 1st National Conference of the Association for Geographic Information, 11-12th October.

MAYO, R.H. DURRANT, A.M. & FINNIEAR, L.J. (1986): "The Application of the New Medusa Ground Modeller - Preliminary Findings of the Beta Test", User Group Meeting, Cambridge Interactive Systems, Harston Mill, Cambridge, March.

MEDYCKYJ-SCOTT, D. (1989): "User and Organisational Acceptance of Geographical Information Systems: The Route to Failure or Success", AGI 89, Conference of the Association of Geographic Information, Birmingham, 11 - 12 October.

MENON, S. & SMITH, T.R. (1989): "A Declarative Spatial Query Processor for Geographic Information Systems", Photogrammetric Engineering and Remote Sensing, Vol 55, No 11, pp1593-1600, November.

MINSKY, M. (1975): "A Framework for Representing Knowledge", in "The Psychology of Computer Vision", Winston, P.H. (Ed.), McGraw Hill (Pub.)

MOOREHOUSE (1990): "The ARC/INFO Geographic Information System", Proceedings of the Conference on GIS Models and Functionality, Midlands Regional Research Laboratory, University of Leicester, 21-22 March, 1990.

MORSE, B.W. (1987): "Expert System Interface to a Geographic Information System", AUTOCARTO 8, Proceedings of the Eighth International Symposium on Computer Assisted Cartography, Baltimore, 29th March - 3rd April.

MORTON, G.M. (1966): "A Computer Oriented Geodetic Data Base, and a New Technique in File Sequencing", Internal Report, IBM Canada.

NANTUCKET (1988): "Clipper Programmers Manual", Supplied with the Nantucket Clipper compiler for the dBase programming language.

NAYLOR , C. (1987a): "Sluggish Sales Dampen Expert Expectations", PC Week, 14th January, 1987.

NAYLOR , C. (1987b): "Cut Out The Professional Advisor And Tap Into A Private Pool Of Knowledge"", PC Week, 21st January, 1987.

NAYLOR , C. (1987c): "Faster, Easier To Learn, But Less Satisfying Than Xi-Plus"", PC Week, 28th January, 1987.

NEWELL, R.G. & THERIAULT, D.G. (1990): "Is GIS Just a Combination of CAD and DBMS ?", Mapping Awareness, Vol 4, No 3, April.

OLLE, T.W. (1978): "A CODASYL Approach to Database Management", John Wiley & Sons (Pub.).

OSHIMA, T., YASUDA, Y., & EMOORI, Y. (1986): Computer Aided Route Selection on the Base of GIS by a Micro-Computer", Second International Symposium on Spatial Data Handling, Seattle, Washington.

OXMAN, R. & GERO, J.S. (1987): "Using Expert Systems for Design Diagnosis and Design Synthesis", Expert Systems, Vol. 4, No. 1, February.

PALMER, B. (1984): "Symbolic Feature Analysis and Expert Systems", Proceedings of the International Conference on Spatial Data Handling, Zurich, Switzerland.

PAPERBACK SOFTWARE, (1987): "VP-Expert - Rule Based Expert System Development Tool", Software Users Guide supplied by Paperback Software Inc.

PARTRIDGE, C. (1990): "Land Registry's Digital Deeds", CADCAM International, April.

PASCOE, G.A. (1986): "Elements of Object Oriented Programming", BYTE, Volume 11, Number 8, pp139-144, August.

PAVELIN, C. (1988): "Logic in Knowledge Representation"; in "Approaches to Knowledge Representation, An Introduction", Ringland, G.A. & Duce, D.A. (Eds.), 1988, Research Studies Press Ltd. (Pub.), ISBN 0 86380 064 5.

PEACEGOOD, G. (1985): "Expert Systems in Remote Sensing : A Preliminary Application", Proceedings of the Poster Sessions of the International Conference of the Remote Sensing Society, London, September.

PETRIE, G. (1985): "Remote Sensing and Topographic Mapping", in Remote Sensing in Civil Engineering, Kennie, T.J.M. & Matthews, M.C. (Eds.), Surrey University Press.

PETRIE, G. (1987): "Terrain Data Acquisition and Modelling From Topographic Maps", Short Course on Terrain Modelling in Civil Engineering, University of Surrey, 7th-9th April.

PEUQUET, D.J. (1983): "A Hybrid Structure for the Storage and Manipulation of Very Large Spatial Data Sets", Computer Vision, Graphics and Image Processing, 24(1), pp14-27.

PEUQUET, D.J. (1984): "Data Structures for a Knowledge Based Geographic Information System", Proceedings of the International Symposium on Spatial Data Handling, Zurich, Switzerland, 20th-24th August.

PEUQUET, D.J. (1987): "Advanced Techniques for the Storage and Use of Very Large Heterogeneous Spatial Databases - The Representation of Geographic Information", Interim Report II, Pennsylvania State University, submitted to the NASA Goddard Space Flight Center, 15th December.

PEUQUET, D.J. (1988): "Toward the Definition and Use of Complex Spatial Relationships", Proceedings of the Third International Symposium on Spatial Data Handling, Sydney, Australia,pp 211-233.

PEUQUET, D.J. (1989): Personal Communication.

POLLITT, M. (1989): "Cost Cutting in Cork", CADCAM International, December.

POPPLESTONE, R.J. (1979): "Specifying Manipulation in Terms of Spatial Relationships", Proceedings of the International Seminar on Programming and Languages for Industrial Robots, IRIA Rocquencourt, France, 27th-29th June.

PRICE. N. & WEBB, H. (1987): "Creation of Terrain Models using Analytical Photogrammetry and their use in Civil Engineering", Short Course on Terrain Modelling in Civil Engineering, University of Surrey, 7th-9th April.

PURVIS, J. (1989): "GIS - Moving Outside the Square", AGI 89, Conference of the Association of Geographic Information, Birmingham, 11 - 12 October.

REICHGELT, H. & VAN HARMELEN, F (1985): "Relevant Criteria for Choosing an Inference Engine", Proceedings of the Fifth Technical Conference of the British Computer Society Specialist Group on Expert Systems.

RIPPLE, W.J. & ULSHOEFER, V.S. (1987): "Expert Systems and Spatial Data Models for Efficient Geographic Data Handling", Photogrammetric Engineering and Remote Sensing, Vol 53, No 10, pp1431-1433, October.

ROBINSON BARKER, G. (1988): "Remote Sensing: The Unheralded Component if Geographic Information Systems", Photogrammetric Engineering and Remote Sensing Vol 54, No2, pp195-199, February.

ROBINSON, G. & ZALTASH, A. (1989): "Application of Expert Systems to Topographic Map Generalisation", Proceedings of AGI 89, Birmingham, Association for Geographic Information, 11th-12th October.

ROBINSON, V.B. & FRANK, A.U. (1987a): "Expert Systems Applied to Problems in Geographic Information Systems: Introduction, Review and Prospects", AUTOCARTO 8, Proceedings of the Eighth International Symposium on Computer Assisted Cartography, Baltimore, 29th March - 3rd April.

ROBINSON, V.B. & FRANK, A.U. (1987b): "Expert Systems for Geographic Information Systems", Photogrammetric Engineering and Remote Sensing, Vol 53, No 10, pp1435-1441, October.

ROBINSON, V.B., FRANK, A.U. & BLAZE, M.A. (1986a): "Expert Systems and Geographic Information Systems: Review and Prospects", Journal of Surveying Engineering, Vol. 111-112, pp119-130.

ROBINSON, V.B., FRANK, A.U. & BLAZE, M.A. (1986b): "Introduction to Expert Systems for Land Information Systems", Journal of Surveying Engineering, Vol. 111-112, pp109-118.

ROLLAND, F.D. (1990): "Relational Database Management with Oracle", Addison Wesley (Pub.).

SAMET, H, SHAFFER, C.A. NELSON, R.C. HUANG, Y-G. FUJIMURA, K. & ROSENFELD,A. (1987) : "Recent Developments in Linear Quadtree-Based Geographic Information Systems", Image and Vision Computing, Vol 5, No 3, August.

SAMET, H. ROSENFELD, A. SHAFFER, C.A. WEBBER, R.E. (1984a): "A Geographic Information System Using Quadtrees", Pattern Recognition, Vol 17, No 6, pp647-656.

SAMET, H. ROSENFELD, A. SHAFFER, C.A. WEBBER, R.E. (1984b): "Use of Hierarchical Data Structures In Geographical Information Systems", Proceedings of the International Symposium on Spatial Data Handling, Zurich.

SCHRIEBER, G. BREUKER, J. BREDEWEG, B. & WIELINGA, B. (1988): "Modelling in KBS Development", Proceedings of the Second European Knowledge Acquisition Workshop EKAW '88, Bonn, June.

SERNADAS,A. SERNADAS,P. & HANS-DIETER,P.(1987): "Object Oriented Specification of Databases: An Algebraic Approach", Proc. 13th Very Large Data Bases Conference, Brighton.

SHELDON, T. (1990): "Windows 3 Made Easy", Osborne McGraw Hill (Pub.).

SINAH, A.K. & WAUGH, T.C. (1988): "Aspects of the Implementation of the GEOVIEW Design", International Journal of Geographical Information Systems, Vol. 2, No. 2.

SKIDMORE A.K. (1989): "An Expert System Classifies Eucalypt Forest Types Using Thematic Mapper Data and a Digital Terrain Model", Photogrammetric Engineering and Remote Sensing, Vol 55, No 10, pp1449-1464, October.

SLATTER, P.E. (1987): "Building Expert Systems - Cognitive Emulation", Ellis Horwood Pub. ISBN 0-7458-0065-3.

SMITH, T., PEUQUET, D.J., MENON, S. & AGARWAL, P. (1987): "KBGIS-II : A Knowledge-Based Geographic Information System", International Journal o Geographical Information Systems, Vol. 1, No. 2, pp149-172.

SMITH, T.R. (1988): "A Knowledge Based Geographic Information System", in the Final Report, Year 5, Remote Sensing Information Sciences Research Group, University of California, Santa Barbara, Estes, J.E. et al (Eds.). 1st June.

SMITH,T.R. & PAZNER, M. (1984): "Knowledge-Based Control of Search and Learning in a Large-Scale GIS", Proceedings of the International Symposium on Spatial Data Handling, Zurich, Switzerland, 20th-24th August.

SOBEL, M.G. (1988): "A Practical Guide to the UNIX System", Benjamen Cummings (Pub.).

SPOT, (1988): "SPOT - Images A La Carte", Product Literature, SPOT Image, 16 BIS, Av. Edouard Belin, 31055 Toulouse, CEDEX, France.

STOMS, D.M., STAR, J.L. & ESTES, J.E. (1988): "Knowledge Based Image Data Management: An Expert Front End for the BROWSE Facility", Technical Papers of the ACSM-ASPRS Annual Convention, Vol 4, Image Processing/Remote Sensing, St Louis, March 13-18.

STROUSTRUP, B. (1986): "The C++ Programming Language", Addison Wesley (Pub.).

TODD, P. (1989): "Cost-benefit - The Solution", AGI 89, Conference of the Association of Geographic Information, Birmingham, 11 - 12 October.

Van CLEYNENBREUGEL, J., FIERENS, F., SUETENS, P. & OOSTERLINCK, A. (1990): "Delineating Road Structures on Satellite Imagery by a GIS-Guided Technique", Photogrammetric Engineering and Remote Sensing, Vol 56, No 6, pp893-898, June.

Van der LANS, R.F. (1989): "The SQL Standard, A Complete Reference", Prentice Hall International (Pub.).

WATERMAN, D.A. (1986): "A Guide to Expert Systems"; Addison Wesley Publishing Co.

WAUGH, T.C. (1986): "A Response to Recent Papers and Articles on the Use of Quadtrees For Geographic Information Systems", Proceedings of the Second International Symposium on Spatial Data Handling, Seattle, Washington, pp33-37.

WAUGH,T.C. & HEALEY,R.G. (1986): "The GEOVIEW Design - A Relational Data Base Approach to Geographical Data Handling", Proceedings of the Second International Symposium on Spatial Data Handling, Seattle, Washington, pp 193-212.

WILLIAMSON, M. (1987): "If It Looks Like A Duck, Walks Like A Duck, Then........."; PC-Week, 6th May, 1987.

WINSTON, P.H. & HORN, B.K.P (1984); "Lisp", 2nd Edition, Addison Wesley Publishing Co. ISBN 0-201-08372-8.

ZHOU, Q. (1989): "A Method of integrating Remote Sensing and Geographic Information Systems",Photogrammetric Engineering and Remote Sensing Vol 55, No5, pp591-596, May.

ZORTECH (1988): "Zortech C++ Compiler", Zortech Limited.

APPENDICES

A. TYPICAL TRANSCRIPT OF AN ELICITATION INTERVIEW

The following transcript is from one of the unstructured interviews with Chas Willis, a pipeline design engineer with JP Kenny & Partners. The transcript illustrates many of the frustrations of the early interviews, including severe digression by the engineer, inappropriate case studies, spatial referencing with only audio recording, and a general lack of structure to the whole thing. The author was thoroughly bewildered during this interview, and struggled to maintain intelligent comments whilst trying to grasp the concepts involved. The transcript has been left in its un-edited form, complete with abbreviations and mis-spellings. The extract given here is approximately one tenth of the total transcription, but it gives a flavour of the work involved.

INTERVIEWS WITH CHAS WILLIS

JP KENNY &PTNRS. TAPE 2 SIDE 1,

MIssed 5 mins on sand waves, before putting tape on

CHAS : Mech. on sand wave movements not well understood, but have been meas. and 30m pa meas. in some places. But edge of field doesn't move like this, the boundaries do not move a great deal, so if you can avoid a field completely & WHOLE JOB can avoid sand wave fields than you would seriously consider putting an extra pipelength in.

LEE: but if you cannot avoid all, then you wouldn't worry?

CHAS : that's right, you could look at the detailed survey and weave between the sand waves, but by the time they come to install it the sand waves have all changed anyway. Survey data year old. New survey done just before pipe inst., just before pre sweeping, but too late then. So sand wave field, go through. Rock ridges, make more effort to avoid.

LEE : So you make more effort to avoid features that mean you req. extra eqpt. on site that you would not normally have needed?

CHAS : Yes, specialist vessels cost lots to mobilise, say day cost is £10 000, but say £70 000 mobilization add on.

LEE : So if you need it for a bit, you might as well use it for longer + go for a straighter route perhaps?

CHAS : Yes,

LEE : Rock ridges, are there any other problems, barriers,

CHAS : not a lot of rock in S. North sea. LOts of chalk inshore with sand on top, not impossible to trench, but may need diff. machine. & poss diff. contractor, price blackmail. Monopoly,

All pipelines to Bacton through chalk, cant really avoid it.

Not a real problem.

Another project, sole pit pipeline runs from Bacton right up North to coal pit and sole pit. The water here down to about 70m. Deep compared with surrounding N sea. Pipes avoided deep bit but in otherwise sim. to last. Followed existing pipe 100m away, up. Could then have diverged, but had better survey info close to existing pipe. so kept going, then to platform. Straight across up to 12.5m deep, limit for barge, so followed exist pipeline a bit further around the bend, then in 18m water. Could have gone right around the sand bank, but that's a fair detour, and the sand bank at this point not shallow enough to be a problem so we cut through here.

LEE : Ok, but you have to cross a telephone cable, + 2 gas lines.

CHAS : phone cable dead so OK, except that chew up trenchers, so must cut cable using divers and pull away. Pipelines, cross older one by coming up out of the sea bed and spanning. the newer one (only in the season before) negotiations. in advance, so first pipe put in a lot deeper at that point. Our pipe crossing then a lot cheaper. On std. crossing the spanning structure is normally either bitumen mattresses or flexible concrete mats(blocks of conc. linked together with ropes.). You can have other things, concrete plinths, steel structures etc, but not talking very high here. If existing pipeline on the sea bed, not below, height of structure bout .25m high so proper structure not reqd. You wouldn't have anything resting on top of the existing pipe. Thin mattress perhaps in case but not used for support. Normally must cover the whole lot with rock dump, as susceptible to vortex shedding and hooking trawl boards. One crossing maybe use extra mats, to avoid mobilising rock dump vessel, but say 3 + need scour protection for the platform then rock dump cheaper. All Thames pipeline rock dumped.

The routing for the bet two platforms, sole 1 and.... . The bed deepish, and slope steep, like holes in the sea bed, slope can be a problem, 5 d. OK, 10 degrees say you need more tension to lay. Coast of Norway have very steep shore approaches, rocky, pre blasting, embankments to provide a smooth profile, doesn't affect the routing, because no matter where you go you still have to do the same. You can optimise to get a better profile, but you have to go across somewhere. Detail routing only. Even used a concrete tunnel once in 20m sections, spanning from rock bit to rock bit, + pull pipe up tunnel. Support + protection. Many solns. Blasting still gives bumpy bits, so divers have to rock dump, pour concrete etc.

Main problems then : 1. existing pipelines

2. sand banks

3. sand waves

Anchors part of the positioning.

LEE : how do you work out the anchor lengths ?

CHAS : Get hold of one of the sketches that the contractors produced on the last job, and say "well it looked like this before so it will look like this again !"ANchor drawings were actually produced by the contractors, its their vessel so they know where they want to put their anchors.

Back to welland job

Consultants sometimes don't think. They put umbilicals in between the existing lines and a future pipeline in between 2 existing pipelines. Had only 25m spacing bet lines. trying to thread 2 new pipelines into a 75m gap bet 2 existing. STUPID!. Got all sand waves here re establishing, so need to sweep again, but will damage existing pipelines. If you HAD to, it could be got around, but no need if another route could be found. Here you can.

Welland : split pipelines and Umbs. as separate contracts, also pre sweeping of Umbs is less as they bend more, so cheaper. Also aesthetics (?!).

Because inst. costs are so high, you could sacrifice a lot of matl. costs (route length) to save a couple of days on the installation. Many vessels could be out there, and could be delayed by other operations, so must keep them busy. Separating small and large dia. lines could allow 2 diff. vessels to work simultaneously etc. 20 miles of 8" pipe on a reel, lay direction can be quite important as it takes a long time to get a barge from one end to the other. Speed 17km takes 2 days at tow speed (not lay speed !). 2 days at 100 000 per day !.

But, if you have a start up and a lay down at one platform you have to have 2 sets of procs. So harder.

This job (welland) simplified as all lines laid before any of the tie ins are done. Otherwise anchors a problem, and may need to lay away rather than towards.

LEE : How long to design a typical small pipeline ?

CHAS : Not allowed to build a platform in a deep water shipping lane. Prelim design by John Brown in 2 3 months 6 months. Detailed design by us, could be just checking and expanding on the calcs, or proposing a completely diff. design and starting from scratch. Welland to be installed next year. 3-4 months for detailed design. Things like buying pipe takes a long time, so need to leave time for this. work out exactly how much pipe we need (verticals, wiggles, late obstructions not acc. for, weld qualifications, buckle repair). Need to order extra pipe for these eventualities.

LEE : Wrecks?

CHAS : cant lay over top, but can go close, and also lay close to avoid anchor handling snags.

END OF TAPE 2 SIDE ONE.....

INTERVIEWS WITH CHAS WILLIS

JPKENNY, 1ST AUGUST 1988, TAPE 2 SIDE 2

LEE : When do you get your info ?

CHAS : there are 3 stages to the design , conceptual, prelim. and detail, sometimes they overlap, or one may get missed out completely, Welland is detailed. We do the fiddly bits.

Conceptual is were you say "we've got a field here and here and we want to join them . the prelim. might come up with diff. routes. You prob. wouldn't have enough envir. data for stability so make a wild stab. Cant incorporate detailed stuff into the prelim choice of the route. Ob. in shallow water there is more of a stability problem, so needs more concrete.

Getting env. data is often a prob. for prelim design. More data at an existing platform, or even at a proposed platform, but along a route may be years out of date and sketchy. TOpographic is perhaps reasonably reliable but not accurate enough. SOle Pit job : comparison of alternatives. gave detailed survey data of the route following existing pipeline + copies of prev. surveys for diff. pipelines which had followed this alternative route. But not detailed enough to directly compared. Little wiggles in surface stated as meggaripples, no size indication. But in first survey showed more detail. So could work out pre sweeping etc immed. Other route needed new survey. (additional cost just to compare the two routes, no ind. that any was better, so didn't bother with other survey). Stayed with well surveyed route.

Admiralty charts not good enough.

Survey provided by client for route design. Look like pipeline alignment sheets, should provide contours , bathymetric detail, existing pipelines , and a prelim route profile, chosen in prelim route design.

LEE : Do you do prelim designs ?

CHAS : Yes.

LEE : What info did you have for that ?

CHAS : Only prev surveys done for prev jobs. plus admiralty charts which are not detailed enough to make any detailed decisions. Ok, you can see the sand banks but not much more than that. extrapolate from old pipeline surveys in area. Cant survey whole lot so must decide on a restrictive corridor before surveying.

LEE : How wide corridor ?

CHAS : As wide as you like but it just costs more money ! Usually, 100m ish wide and to do this maybe 5 runs of the route by survey ship reqd.

LEE : But how do you decide on the corridor route ?

CHAS : In this area (Welland) do what the other pipelines did ! (as 1st stab). In wide open spaces you prob go for the straight line approach ! until you hit a problem.

Other restrictions, up north in the sole pit devpt a block was owned by Ranger Oil, and they said "you cant put a pipeline here cos we want to drill here !!". We had to divert during detailed design and re survey.

LEE : Do you get your sand bank info from the Admiralty Charts (ACs) ?

CHAS : Unless someone else has surveyed, the AC is all we have. BUt see AC caution "depths derived from lead line surveys between 1886 to 1931, the charted depths cannot be relied upon" (!!!) So you know roughly, but cannot take the actual readings literally.

So after rough route do a single line survey, if flat all along, suppose that small devs. in route will be Ok. You will eventually req a detailed survey, but don't need for prelim design. But if general terrain characteristics show many features of significance then you couldn't realign by 5km without a re survey. Normally the detail design doesn't change the route. SO stay within the detail survey corridor and can wiggle as much s you want. straying outside gives problems because you only have very sketchy information outside the corridor.

We had the problem with the Thames flowline that, we started running parallel to an existing pipeline, 100m apart, nice, no anchor problems; then dev towards the well. That was when we were doing the startup from the platform leg. Because they eventually turned around and said we had to start from a normal anchor in the opposite direction. We couldn't get parallel this way without being farther away (250m), + outside the detailed corridor. So more survey work had to be carried out.

This whole area is sand waves. size 3m, not severe but enough to analyse prior to deciding whether to pre sweep. These (x sections?) have a 20:1 vt. to hz. distortion, enough so that things that look alright are not, and vice versa. But not very detailed. At 40:1 you see the shape of the sea bed a little bit more, but even more confusing. A pipe span that would be acceptable would look like that. It looks silly. the curvature of the sea bed looks different depending on the slope. What looks like a big lump at the top of a sand wave, if it was half way down the side it would disappear in the plotting.

LEE : I notice you have some info. about soils here. is that normal

CHAS : All provided by the client as a result of the surveys for conceptual routing, or taken from surveys for prev. jobs. Much is guesswork. They show horizons etc but sometimes they draw conclusions which really are not justified, even though it is clearly marked on the plan! Other times they give you data which is so vague and woolly, that you can't use it.

On Sole Pit job we knew that the chalk was outcropping in placesin the first few kilometres, until at 20km it was about 4m down and so no longer of consequence. But it was only after nearly finishing the job that we found that within the 1st km the chalk didn't actually outcrop, but it was only 200mm below, so trenching was tricky, but closer still it was deeper again. So we changed the length of the shore approach to minimise the crossing of the chalk layer. The shore approach was (pre)dug in the thick sand, and the pipeline was laid on top of the chalk in the thin sand, and then trenched to its abs min reqmt afterwards. But this info was only found when they did an extra detailed survey AFTER we had done the detailed design.

LEE : * so is it all basically guesswork until you get the reqd info to back you up ?

CHAS : Guesswork is perhaps not a good word to use.... but its not far wrong. The topo and env data is usually lagging behind interms of time. I suppose because it takes time to get the surveys done etc. but also the investment. It cheaper to have a guy at a desk doing calcs. than it is to have a boat out so you don't do a survey until you need it...

LEE: and have gone as far as you can with the other info.

CHAS :A lot of interpolation + extrapolation is better than "guesswork".

LEE : Hmmmmmm

CHAS : * Another of the factors we were talking about: platform approaches. Welland pipeline heading

LEE: We are looking at a plan of the Welland field facilities

CHAS : (explaining how old design crossed unnecessarily other pipelines Not relevant) min 1000m rad for a large pipeline. We've been using 1500m. Its not until you know how heavy the line is, by doing the detailed design, that you know the min radii for the pipe. Depends on the lateral friction. If you lay it on too tight a curve it will simply straighten itself out before it gets trenched, an elastic effect, friction depends on the weight.

In the routing you stick to nice round numbers (eg 1500m)for radius as the contractors don't like doing it and they wouldn't guarantee the accuracy of the curvature.

Telephone repeater station. BT Couldn't guarantee the position of it within 200m. Also not allowed to be within 500m. When crossing an existing pipeline its very hard to do in a curve. The curvature relies on bottom friction, so when the pipeline is lifted onto supports, you haven't got the same friction and the pipeline will straighten. V. Large curves might be OK.

Tie in spools must be kept short. 100m is very excessive.

....

LEE : Back on the air !!! We are looking at detailed design.

CHAS :Must consider future plans for platforms etc. in the routing of the pipeline.

LEE : at the time you design the line you have this information ?

CHAS : We were effectively given this drawing showing that platform, that platform and that platform. Saying you have to avoid these. They may tell you later that they have moved the platform and you have to move the line.

LEE : But you wouldn't be guessing that a platform might be there, say

CHAS : No, they would be definite or none at all. Here we have an alternative jack up site because one of the companies they may use has this enormous great rig. (The jack up rig cantilevers over the platform and drills through the platform.)

[Looking at another job with Pls coming into a platform. I am not sure which platform...]

LEE : So I think the problems are concentrated around these rigs are they not ?

CHAS : they've ben shown (the pipelines) as coming in only 5m apart. and at the lay down pts that is about as close as you can get. a target zone (shallow water 35m) would be 4m wide. You could say tat the pipeline will definitely be within this zone and an axial tolerance of 5m. But if one Pl is at one side of its zone and the other at the other then they are only 1m apart. Not good, as you have no access for eqpt. The only way you could get a 5m tol. is to lay the 1st pipeline and re survey precise posn., relocate the 2nd plan then lay it.

But the client dwg is wrong cos they showed the top od the platform not the bottom, so we had more space. and we can get a 10m gap. They diverge to 50m then run parallel to the wells.

LEE : where is the export line?

CHAS : that's this one. It goes straight.

One reason they stay together is for the ease of anchor handling, another is that it si easier in the future for other people to say " there are Pls in this CORRIDOR here", Pls all over would cause AH probs in the future so it helps everyone else to keep a corridor. But wouldn't go out of the way to do this just for other peoples convenience. It has to be better and cost less etc too.

This deep water shipping channel (DC) is guaranteed 25m deep. We couldn't put a platform in it but we can put well heads in as long as they are not too tall. In prelim design they thought that instead of going at an angle across the deep water shipping lane they would go straight across, minimising the time in the channel. This does not affect the pipeline once installed but any vessel movements in the deep water shipping lane have to be coordinated with shipping movements. you couldn't have 3 vessels wandering in the deep water shipping lane cos there might be a bloody great tanker coming along.

LEE : so they might snag your anchor lines or something ?

CHAS : Well no, they would be sure to stay far enough away to avoid that, its far too expensive. So if you had as few ops as possible within the deep water shipping lane it would be better, but as they are putting in well heads anyway, and each will req a jack up rig in the middle of the deep water shipping lane. So you cant avoid, or even minimise it , so just ignore it ! To avoid you would have to move the wells somewhere else.

LEE : so then you would have to slant drill ?

CHAS : Well no, you cant drill that far. You can drill a km or so I think, dep on the depth. The coords are in trans merc proj, but ACs in geographical coords.

LEE : Do you do much work for BGC ?

CHAS : No, Did an assessment for Rough for reuse of pipelines. Also did Morecombe for HGB. Flat, straight, shallow. platform inst cons were much more important in this case.

end of tape 2 side 2 AT LAST................ !

B. PAPER MODEL OF THE PIPELINE ROUTING PROCEDURE

Two aspects of pipeline route design had to be considered prior to the development of an PIRATE, these being the work flow associated with the design process, and the rules used to make judgements during the design. The following text is derived from knowledge elicitation interviews with a pipeline engineer from JP Kenny & Partners. It gives the pipeline design work flow as he perceives it, illustrated through a particular example pipeline project called the Pickerill Field Pipeline in the UK section of the North Sea. After the process was written down, it was returned to the engineer for verification, and changes suggested by him have been incorporated in the text below.

1. Engineers receive contract, start and finish points, details of the material to be transported (oil or gas), provisional pipe diameter and Codes of Practice to be followed. Admiralty Charts (ACs) and Telecom Cable route charts are also available for the North Sea region.
2. Identify end points of the pipeline on AC. Consider a straight line route with respect to :

   Bathymetry
   Existing Installations/pipelines

   In particular identify areas where :

   Dredging - required for lay barge access
   Steep Slopes - may cause overstressing etc.
   Other Known Uses are made of the sea bed (Eg. anchor zone)
   Existing Pipelines and other man made obstacles
   Areas Of Known Geologic Difficulty

3. Consider routes which avoid the obstacles hit by the straight line route.
4. Assess these alternatives with respect to :

   Reduced cost due to avoiding obstacle
   Increased costs due to extra pipeline length
   Any other problems hit by the diverted pipeline

5. Possibly amend any alternative routes to avoid the obstacles encountered.
6. Discard non-feasible or non-economic alternative routes and select those for further study.

7. Produce preliminary comparative costings between selected routes and assess these prior to requesting surveys.

[Time taken for points 1 to 5 approx. 2 weeks, further time for points 6 and 7 approx. 2 weeks per alternative line. this includes drawing production]

In this preliminary costing the basic comparison includes:-

Material Costs
Dredging Costs
Lay Barge Costs
Pipeline/cable Crossings
Trenching Costs (Preliminary as sea bed sediments not known)
Blasting Costs
Diver Costs (other than basic diver support)

8. The specification for the required detailed surveys are decided upon and the survey is requested.

9. Survey results obtained - Results of the survey are generally over a 600m wide corridor unless otherwise requested. For the Pickerill pipeline the following types of data were obtained:-

- A contour map showing :

  - **Surface Conditions**
  - **Wrecks**
  - **Other Unidentified/identified Obstructions**
  - **Line Of The Pipeline**
  - **Ship Tracks**

- A contour map of the depth of the first sub-surface change in soil type.

- A sectional profile along the route showing :

  - **Sub Bottom Profiler Results**
  - **Soil Conditions At Various Depths**

- Borehole results were also given at an average of 500 metre intervals along the pipeline route.

10. Once the survey results have been obtained an off-shore survey review is produced by JPKenny, which is essentially a summary of the survey and the way it affects the pipeline routes proposed. Particular attention is paid to the more accurate bathymetry now available. In the Pickerill case the passage of construction vessels and related dredging requirements could only at this stage be accurately estimated.

Also the knowledge of sediment occurrences now makes a detailed analysis of trenching possible. Obstacles which are positively identified along the pipeline route may involve moving the route slightly to avoid the problem, though usually the route remains within the surveyed corridor.

The presence of bedform activity can also be more accurately identified and pre-sweeping requirements assessed.

11. Some route optimisation may be attempted at this stage (for example on the Pickerill line in areas where dredging is complex and significant). In practice, however, the assessment of large numbers of alternatives is not attempted as it is too tedious and time consuming. Only clearly advantageous lines are addressed in isolation.

12. A full cost analysis of the chosen alternative(s) is then produced. The client, who receives this, usually decides finally on which route alternative to use. Once again, depending on client requirement, these costings may be comparitive rather than absolutely accurate, using last years prices and other simplifications.

13. A full report and all necessary drawings are produced, and areas of unusual bedforms may be analysed for pipe stress and spanning problems. Environmental conditions, though not normally critical in the North Sea except in shallow water, will also be considered in the report.

C. PAPER MODEL OF THE PIPELINE ROUTING RULES

This appendix gives a summary of the rules found during knowledge acquisition for the PIRATE system design, Chapter 5. The rules relate to pipeline routing, and essentially give a *paper model* of the knowledge needed for pipeline design. The rules are given in natural English, and specific values in brackets refer to the particular limits used for the Sable Island Gas Pipeline project, which is described in Chapter 8.

Overall route optimisation is based on minimising cost whilst achieving an acceptable standard of safety. Each feature or constraint on a particular route is considered with respect to the planning, installation and operation of the pipeline. Essentially it must be discovered whether it is safe for a pipeline to traverse a feature, and this being so how much it will cost. Costs are both capital and revenue, applying to the installation and future operation of the pipeline.

Routes are compared under the following categories :

Bathymetry
Sediment Transport
Underlying and Surficial Geology
Environment
Fishing and Man Made Obstructions
Platform Approach
Future Tie-ins to Other Lines

The following sections deal with the specific rules which are relevant under each of the above categories. The format of the rules under each category is a bulleted list. In this list a dash (-) stands for factors affecting the category or limiting values, whilst a double dash (=) gives the resulting effects that the existence of a category will have. Values mentioned in brackets are specific values taken from the Sable Island Pipeline Project, and are considered typical.

1. BATHYMETRY

DEPTH :-

Maximum Depth

- 200 metres

=Beyond this special pipelay vessels are needed and pipe laying becomes difficult

Minimum Depth

- Draught of lay vessel (10 metres)

=Beyond this dredging will be required to allow the lay vessel to pass. Alternatively other methods of pipe laying could be considered, for example pulling the pipe from the shore

Wave Affected Depth

- Max affected depth = 1/2 Wavelength of surface waves (60 metres)

=Less than this depth means that the sea bed will be affected by wave activity. This has implications for both the occurrence of active bedform features, megaripples for example, and the stability of an un-trenched pipeline. Stresses caused by wave currents would need to be analysed

SLOPES

Steep Slopes 1

- General (no specific classification figure found)
- Within 2.5 kilometres of the route

=Anchor handling difficulties. Steep slopes can cause snagging of anchor cables. More prevalent in shallow water where undulations and slopes can cause cables to snag as the lay vessel moves forward.

Steep Slopes 2

- Greater than 3^{o} in fine sediments
- Greater than 6^{o} in coarse sediments
- On the pipe route itself

= If the pipe crosses a steep slope, it should be directed to do so perpendicular to the line of the slope so as to minimise the risk of movement due to slope slippage. Steep slopes can cause overstressing in the pipeline which would require analysis. Also there is the possibility of free spans occurring due to rapid slope changes along the pipe, which will present difficulties with vortex shedding due to currents and may snag fishing gear. Steep slopes also necessitate an increased lay tension from the pipelay vessel, possibly leading to a larger pipelay vessel being needed.

2. Sediment Transport and Bedform Activity

SEDIMENT TRANSPORT:-

Sediment Overspill Areas

- Slope steepness

- Sediment type
- Current and wave effects
- Occurrence near enough to affect the route

= If sediments are moving onto the pipe route it can cause movement, overstressing and excessive burial. If sediments are moving away from the pipe route it can cause exposure of the buried line, localised spanning, vortex and scour problems.

BEDFORM ACTIVITY

Sand Ridges and Sand Waves

- Wave height (3 - 7 metres)
- Wave length (300 - 3000 metres)
- very low curvatures
- longitudinal, the crest runs in the same direction as the current

= Generally sand ridges have little effect. The crest profile may need pre-sweeping to avoid overstressing or spanning. Also if they are very mobile they could cause re-exposure problems.

Sand Ribbons

- Wave Height (minimal)
- Longitudinal

= Sand ribbons have no effect on the route, but can indicate areas of high current activity.

Megaripples

- Wave Height (0.5 - 1.5 metres)
- Wave length (15 - 29 metres)
- Transverse to currents
- Curvatures relatively high

= Megaripples can cause overstressing, spanning and re-exposure of the pipeline. The pipeline is generally trenched to below the level of the megaripple troughs to avoid re-exposure.

3. Surficial and Underlying Geology

SURFICIAL GEOLOGY

Surficial Sediment

- Sediment type
- Sediment thickness
- Sediment properties

= Affects ease and cost of dredging and trenching. It may also affect pipeline stability. Affects classification of steep slopes, formation of mobile bedforms, anchor stability and holding force.

Surficial Boulder Fields

- Rock type
- Density of boulders (number per 100 square metres of sea bed)
- Average boulder size

= Affects laying of pipe. All surficial boulders in the way of the pipeline must be removed by lifting or blasting. Boulders can also cause anchor cable snagging, and if mobile can be a continual threat to the pipeline, which in this case should be trenched where possible.

Surficial Rock Outcrops

- Rock Type
- Protrusion height
- Roughness

= Basically they need to be avoided as the pipe will require support over the outcrop or a trench must be blasted through it. Depending on the protrusion height and the extent of the outcrop, anchor handling becomes very difficult. Spanning, overstressing and pipe stability will all be problems.

UNDERLYING GEOLOGY

Underlying Sediment

- Depth of interface with surficial sediment
- Sediment type
- Sediment thickness
- Sediment Properties

= Depending on the interface depth the underlying sediment can affect trenching, dredging, formation of mobile bedforms and anchor holding strength.

Underlying Solid Rock

- Depth of interface with surficial sediment
- Rock type
- Rock contiguity
- Roughness

= Depending on the interface depth this can affect trenching, dredging, formation of mobile bedforms and anchor holding strength.

Underlying Boulder Fields

- Minimum depth below surface
- Mean depth below surface
- Rock Type
- Density of boulders (number per 100 square metres of sea bed)
- Mean Boulder size

= Depending on depth below surface, boulders can affect trenching and dredging activities.

4. Environment

No specific rules were elicited for environmental effects, but the following factors need to be considered:

Currents caused by

- Wind shear
- Waves
- Tides
- Residual currents (for example the Gulf Stream)

Routes must avoid shoaling caused by breaking waves. Environmental stability for the pipeline most be provided, by trenching or concrete coating for example. The wave induced component does not affect depths in excess of 1/2 the wavelength. Tops of banks and other sharp relief features can cause increased current loading.

5. Man Made and Natural Hazards

Fishing

Bottom Feeder Fishing

- Fish type
- size/type of trawler gear
- Sensitivity of the fish habitat

= Trenching may be required in these areas to prevent snagging of trawler boards and other fishing gear.

Shellfish Beds

- Harvesting Methods
- Sensitivity of the environment

= Effect of pipeline on the environment and in particular the habitat of the shellfish. Collection methods may also interfere with or be obstructed by the pipeline

Man Made Obstructions

Wrecks

- Size
- Location
- Ownership/value/ protection status

- Protrusion above the sea floor
= Must avoid wrecks as they are a direct obstacle to the pipeline which are difficult to destroy. They can also cause anchor cable snagging, such that if wrecks are to be avoided the modified route will usually pass close (50 - 100 metres) rather than farther away where the anchor cables from the pipelay vessel would be more likely to snag. Pipelay vessel anchors can have a spread of over two kilometres.

Wellheads

- Location
- Protrusion above the sea floor
- Ownership/operating Status
- Access clearance requirements
- Exclusion zone radius
= Wellheads need to be avoided.

Cables and Existing Pipelines

- Surface/buried
- Ownership
- Used/decommissioned/permanently disused
- Ease of crossing
- Permission to cut through
= Crossing existing cables and pipelines is expensive and should be avoided wherever possible. Permanently disused cables may be cut.

D. A Typical Pipeline Cost Breakdown

The following cost breakdown is for the Base Case Route of the Sable Island Pipeline Project, used as the case study to test the PIRATE system. The actual values of the component costs are based on historical data used by JP Kenny engineers, rather than current commercial costs and rates. This section is intended not only to give the reader an appreciation of the magnitude of financial investment needed to install a pipeline, but also to show the proportion of the cost conferred by each part of the project.

Example Costs For The 263 Kilometre Base Case Route

Materials

Line Pipe	$72,135,000
Corrosion Coat	$21,418,000
Concrete Coat	$17,698,000
Other Items	$20,349,000
Total Material Costs	$131,600,000

Construction Costs

Lay Vessel

Mob/Demob	$6,900,000
Number of Days Needed	
263km/1.95km per day = 135	
Add extras, contingency and weather downtime	
135 days + 53 days = 188 days	
Day Rate = $460,000	
Operating Cost	$86,480,000
TOTAL	$93,380,000

Dive Support Vessel

Mob/Demob	$2,400,000
Operating Time = 51 days + 35% downtime	
Day Rate = $120,000	
Operating Cost	$8,280,000
TOTAL	$10,680,000

Trenching Spread

Mob/Demob	$3,000,000
Output Quantity = 3.5km/day (single pass)	
Day Rate = $200,000	
No. of days needed = 89 + 25% downtime	
Operating Cost = 111 x $200,000 =	$22,200,000
TOTAL	$25,200,000

Sea Bed Preparation (Blasting)

Mob/Demob	$2,400,000
Time needed = 60 days + 30% downtime	

Day rate = $200,000	
Operating Cost	$15,600,000
TOTAL	$18,000,000
Shore Approach - TOTAL	$5,000,000
Surveys	
Pre-lay	$1,490,000
As laid	$3,300,000
As built	$1,650,000
TOTAL	$6,440,000
Testing Finished Line	$5,061,000

SUMMARY COSTS :

Total Material Costs	:	$	121,500,000
Taxes and Duties	:	$	10,100,000
TOTAL	:	$	131,600,000
Total Installation	:	$	164,740,000
Total Materials and Installation	:	$	296,340,000
Insurance @ 4%	:	$	11,850,000
Contingency @ 13%	:	$	40,060,000
Project Management/Engineering @ 8%	:	$	27,860,000
TOTAL MARINE PIPELINE COST	:	$	376,110,000

E. GOLDWORKSII RULES USED IN THE SABLE ISLAND CASE STUDY

```
(define-rule megga-trench-rule-1
         (
  :explanation-string "If a megaripple field is passed through then the normal remedial
action is trenching. The trench depth is the sum of the pipe diameter, the minimum
cover required above the pipeline, and the wave height of the megaripples in the field.
The pipe has to be trenched to avoid spanning, vortex shedding and possible collision
damage with anchors, trawl boards etc. In this case the trench requires only one pass
of the trencher, according to the maximum cutting depth of GENERIC-TRENCHER."
  :direction :backward
  :dependency t)
  (instance ?attrib is meggaripples
           with wave-height ?ht)
  (instance generic-trencher is trenchers
           with max-cutting-depth ?cut-d)
;  (instance generic-pipe is pipe
;           with diameter ?dia)
;CHANGED ABOVE TO BELOW 24TH NOV. 1990
  (instance design-para-1 is design-parameters
           with chosen-pipe-diameter ?dia)
  (instance design-para-1 is design-parameters
           with min-cover ?cover)
  (bind ?reqd-cut
        (+ ?cover ?dia ?ht))
  (<= ?reqd-cut ?cut-d)
  (bind ?data-list (list 'trenching ?reqd-cut 1))
  then
  (instance ?attrib is meggaripples
           with remedial-action trenching)
  and-then
  (instance ?attrib is meggaripples
           with data-1 ?data-list)
  (comment
   "In the case of trenching, data-1 stores the depth of required cut as the second
element and the third element is the number of passes required with the generic
trencher"))
```

```
(define-rule megga-trench-rule-2
         (
   :explanation-string "If a megaripple field is passed through then the normal
remedail action is trenching. The trench depth is the sum of the pipe diameter, the
minimum cover required above the pipeline, and the wave height of the megaripples
in the field. The pipe has to be trenched to avoid spanning, vortex shedding and
possible collision damage with anchors, trawl boards etc. In this case the trench
requires more than one pass of the trencher, according to the maximum cutting depth
of GENERIC-TRENCHER. The number of passes is given in the third element of the
data-1 list"
   :direction :backward
   :dependency t)
  (instance ?attrib is meggaripples
           with wave-height ?ht)
  (instance generic-trencher is trenchers
           with max-cutting-depth ?cut-d)
;  (instance generic-pipe is pipe
;          with diameter ?dia)
; CHANGED ABOVE TO BELOW 24TH NOV. 1990
  (instance design-para-1 is design-parameters
           with chosen-pipe-diameter ?dia)
  (instance design-para-1 is design-parameters
           with min-cover ?cover)
  (bind ?reqd-cut
        (+ ?cover ?dia ?ht))
  (> ?reqd-cut ?cut-d)
  (bind ?no-of-cuts
        (+ .999 (/ ?reqd-cut ?cut-d)))
  (bind ?int-no-of-cuts
        (truncate ?no-of-cuts))
  (bind ?data-list (list 'trenching ?reqd-cut ?int-no-of-cuts))
;;;NB changed last arg above from ?no-of-cuts 21/11/90
  then
  (instance ?attrib is meggaripples
           with remedial-action trenching)
  and-then
  (instance ?attrib is meggaripples
           with data-1 ?data-list)
  (comment
   "In the case of trenching, data-1 elem 2 =depth of reqd cut, data-1 elem 3 is number
 of passes required with the generic trencher"))
```

```
(define-rule boulder-trench-rule-1
         (
 :explanation-string "If a boulder field is passed through then the normal remedail
action is trenching. The trench depth is the sum of the pipe diameter and the minimum
cover required above the pipeline. The pipe has to be trenched to avoid possible
collision damage with other boulders migrating. In this case the trench requires only
one pass of the trencher, according to the maximum cutting depth of GENERIC-
TRENCHER."
 :direction :backward
 :dependency t)
 (instance ?attrib is boulder-field
          with density-of-boulders ?density)
 (instance generic-trencher is trenchers
          with max-cutting-depth ?cut-d)
; (instance generic-pipe is pipe
;          with diameter ?dia)
;CHANGED ABOVE TO BELOW ON 24TH NOV. 1990
 (instance design-para-1 is design-parameters
          with chosen-pipe-diameter ?dia)
 (instance ?design is design-parameters
          with min-cover ?cover)
 (bind ?reqd-cut
        (+ ?cover ?dia))
 (<= ?reqd-cut ?cut-d)
 (bind ?data-list (list 'trenching ?reqd-cut 1))
 then
 (instance ?attrib is boulder-field
          with remedial-action trenching)
 and-then
 (instance ?attrib is boulder-field
          with data-1 ?data-list)
 (comment
  "In the case of trenching, data-1 elem 2=depth of reqd cut, data-1 elem 3 is number
of passes required with the generic trencher"))
```

```
(define-rule boulder-trench-rule-2
          (
  :explanation-string "If a boulder field is passed through then the normal remedail
action is trenching. The trench depth is the sum of the pipe diameter and the minimum
cover required above the pipeline. The pipe has to be trenched to avoid possible
collision damage with other boulders migrating. In this case the trench requires
multiple passes of the trencher, according to the maximum cutting depth of
GENERIC-TRENCHER."
   :direction :backward
   :dependency t)
  (instance ?attrib is boulder-field
           with density-of-boulders ?density)
  (instance generic-trencher is trenchers
           with max-cutting-depth ?cut-d)
; (instance generic-pipe is pipe
;          with diameter ?dia)
;CHANGED ABOVE TO BELOW ON 24TH NOV. 1990
  (instance design-para-1 is design-parameters
           with chosen-pipe-diameter ?dia)
  (instance ?design is design-parameters
           with min-cover ?cover)
  (bind ?reqd-cut
        (+ ?cover ?dia))
  (> ?reqd-cut ?cut-d)
  (bind ?no-of-cuts
        (+ .999 (/ ?reqd-cut ?cut-d)))
  (bind ?int-no-of-cuts
        (truncate ?no-of-cuts))
  (bind ?data-list (list 'trenching ?reqd-cut ?int-no-of-cuts))
  then
  (instance ?attrib is boulder-field
           with remedial-action trenching)
  and-then
  (instance ?attrib is boulder-field
           with data-1 ?data-list)
  (comment
   "In the case of trenching, data-1 elem 2=depth of reqd cut, data-1 elem 3 is number
 of passes required with the generic trencher"))
```

```
(define-rule diver-support-for-boulders-1
          (
 :explanation-string "When boulder fields have a boulder density greater than 40
boulders per 100 square metres, intensive diver support is required to clear the
pipeline route."
 :dependency t
 :direction :backward)
 (instance ?bou is boulder-field
          with density-of-boulders ?den)
 (> ?den 40)
 (bind ?att (list 'diver-support 'intensive))
 then
 (instance ?bou is boulder-field
          with remedial-action diver-support)
 and-then
 (instance ?bou is boulder-field
          with data-1 ?att))
```

RULES FOR DEFINING DESIGN PARAMETERS

```
(define-rule min-cover-requirement-british-1
          (
 :direction :backward
 :dependency t
 :explanation-string "All pipelines with a diameter of 0.45m or less need a minimum
cover of 0.3 metres" )
 (instance ?des-par is design-parameters
          with chosen-pipe-diameter ?dia
          with national-code-used british)
 (<= ?dia .45)
 then
 (instance ?des-para is design-parameters
          with min-cover 0.3))
```

```
(define-rule min-cover-requirement-british-2
         (
 :direction :backward
 :dependency t
 :explanation-string "All pipelines with a diameter greater than 0.45m need a cover of
.4 metres" )
 (instance ?des-par is design-parameters
         with chosen-pipe-diameter ?dia
         with national-code-used british)
 (> ?dia .45)
 then
 (instance ?des-para is design-parameters
         with min-cover 0.4))
```

```
(define-rule min-cover-requirement-canadian-1
         (
 :direction :backward
 :dependency t
 :explanation-string "All pipelines with a diameter of 0.45m or less need a minimum
cover of 0.4 metres in the Canadian code (example - verify figures)" )
 (instance ?des-par is design-parameters
         with chosen-pipe-diameter ?dia
         with national-code-used canadian)
 (<= ?dia .45)
 then
 (instance ?des-para is design-parameters
         with min-cover 0.4))
```

```
(define-rule min-cover-requirement-canadian-2
          (
 :direction :backward
 :dependency t
 :explanation-string "All pipelines with a diameter greater than 0.45m need a cover of
.5 metres in the Canadian code (example - verify figures)" )
 (instance ?des-par is design-parameters
          with chosen-pipe-diameter ?dia
          with national-code-used canadian)
 (> ?dia .45)
 then
 (instance ?des-para is design-parameters
          with min-cover 0.5))
```

```
(define-rule min-cover-requirement-united-states-1
          (
 :direction :backward
 :dependency t
 :explanation-string "All pipelines with a diameter of 0.6m or less need a minimum
cover of 0.25 metres in the United States code (example - verify figures)" )
 (instance ?des-par is design-parameters
          with chosen-pipe-diameter ?dia
          with national-code-used united-states)
 (<= ?dia .6)
 then
 (instance ?des-para is design-parameters
          with min-cover 0.25))
```

```
(define-rule min-cover-requirement-United-States-2
          (
 :direction :backward
 :dependency t
 :explanation-string "All pipelines with a diameter greater than 0.6m need a cover of
.35 metres in the United States code (example - verify figures)" )
 (instance ?des-par is design-parameters
          with chosen-pipe-diameter ?dia
          with national-code-used united-states)
 (> ?dia .6)
 then
 (instance ?des-para is design-parameters
          with min-cover 0.35))
```

```
(define-rule min-cover-requirement-Norwegian-1
          (
 :direction :backward
 :dependency t
 :explanation-string "All pipelines with a diameter of 0.45m or less need a minimum
cover of 0.3 metres in the usual Norwegian code as complied by Det Norske Veritas
(example - verify figures)" )
 (instance ?des-par is design-parameters
          with chosen-pipe-diameter ?dia
          with national-code-used norwegian)
 (<= ?dia .45)
 then
 (instance ?des-para is design-parameters
          with min-cover 0.3))
```

```
(define-rule min-cover-requirement-Norwegian-2
          (
 :direction :backward
 :dependency t
 :explanation-string "All pipelines with a diameter of greater than 0.45m need a
minimum cover of 0.4 metres in the usual Norwegian code as complied by Det
Norske Veritas  (example - verify figures)" )
 (instance ?des-par is design-parameters
          with chosen-pipe-diameter ?dia
          with national-code-used norwegian)
 (> ?dia .45)
 then
 (instance ?des-para is design-parameters
          with min-cover 0.4))
```

```
(define-rule total-trenching-british-1
          (
 :direction :backward
 :dependency t
 :explanation-string "Pipelines less than .45 metres in diameter have to be trenched
throughout according to British codes of practice")
 (instance ?des-par is design-parameters
          with national-code-used british)
 then
 (instance ?des-par is design-parameters
          with total-trenching-limit-diameter 0.45))
```

```
(define-rule total-trenching-canadian-1
        (
 :direction :backward
 :dependency t
 :explanation-string "Pipelines less than 0.3 metres in diameter have to be trenched
throughout according to Canadian codes of practice (example - verify figures)")
 (instance ?des-par is design-parameters
        with national-code-used canadian)
 then
 (instance ?des-par is design-parameters
        with total-trenching-limit-diameter 0.3))
```

```
(define-rule total-trenching-united-states-1
        (
 :direction :backward
 :dependency t
 :explanation-string "United States codes of practice do not place a figure on the
minimum diameter at which trenching is required. It is left to the designers descretion
(example - subject to verification)")
 (instance ?des-par is design-parameters
        with national-code-used united-states)
 then
 (instance ?des-par is design-parameters
        with total-trenching-limit-diameter 0))
```

```
(define-rule total-trenching-norwegian-1
          (
 :direction :backward
 :dependency t
 :explanation-string "Pipelines less than 0.45 metres in diameter have to be trenched
throughout according to Det Norske Veritas (subject to confirmation)")
 (instance ?des-par is design-parameters
          with national-code-used norwegian)
 then
 (instance ?des-par is design-parameters
          with total-trenching-limit-diameter 0.45))
```

```
(define-rule countries-use-british-code-1
          (
 :direction :backward
 :dependency t
 :explanation-string "The country uses the British code of practice for the design and
routing of pipelines. (verification required)")
 (or (instance ?des-par is design-parameters
               with national-juristiction united-kingdom)
    (instance ?des-par is design-parameters
               with national-juristiction netherlands)
    (instance ?des-par is design-parameters
               with national-juristiction greece)
    (instance ?des-par is design-parameters
               with national-juristiction morocco)
    (instance ?des-par is design-parameters
               with national-juristiction tunisia)
    (instance ?des-par is design-parameters
               with national-juristiction india))
 then
 (instance ?des-par is design-parameters
          with national-code-used british))
```

```
(define-rule countries-use-canadian-code-1
          (
 :direction :backward
 :dependency t
 :explanation-string "The country uses the Canadian code of practice for the design
and routing of pipelines. (verification required)")
 (instance ?des-par is design-parameters
          with national-juristiction canada)
 then
 (instance ?des-par is design-parameters
          with national-code-used canadian))
```

```
(define-rule countries-use-united-states-code-1
          (
 :direction :backward
 :dependency t
 :explanation-string "The country uses the United-states code of practice for the
design and routing of pipelines. (verification required)")
 (or (instance ?des-par is design-parameters
               with national-juristiction united-states)
    (instance ?des-par is design-parameters
               with national-juristiction mexico)
    (instance ?des-par is design-parameters
               with national-juristiction brazil)
    (instance ?des-par is design-parameters
               with national-juristiction panama)
    (instance ?des-par is design-parameters
               with national-juristiction saudi-arabia)
    (instance ?des-par is design-parameters
               with national-juristiction united-arab-emerates))
  then
 (instance ?des-par is design-parameters
          with national-code-used united-states))
```

```
(define-rule countries-use-norwegian-code-1
          (
  :direction :backward
  :dependency t
  :explanation-string "The country uses the Norwegian code of practice for the design
and routing of pipelines. (verification required)")
  (or (instance ?des-par is design-parameters
                 with national-juristiction norway)
     (instance ?des-par is design-parameters
                 with national-juristiction lybia)
     (instance ?des-par is design-parameters
                 with national-juristiction tunisia)
     (instance ?des-par is design-parameters
                 with national-juristiction iran)
     (instance ?des-par is design-parameters
                 with national-juristiction iraq))
  then
  (instance ?des-par is design-parameters
           with national-code-used norwegian))
```

F. THE PIRATE GIS DATA TABLE SCHEMA

This appendix gives the detailed structure of the relational tables within the PIRATE GIS. The dBase format is used, which means that all fields, whether chatacter or numeric, are stored as ASCII text. Hence a real number will take up one character for every digit it has, plus the decimal point and any sign. Field numbers, names, type, width and number of decimal places are given.

The Main Raster Data Table - BIGCELL.DBF

FIELD	FIELD NAME	TYPE	WIDTH	DECIMAL PLACES
1	CELL_CODE	CHARACTER	2	
2	ELEVATION	NUMERIC	7	2

The Combination Table - COMBCODE.DBF

FIELD	FIELD NAME	TYPE	WIDTH	DECIMAL PLACES
1	CURR_CODE	CHARACTER	2	
2	PREV_CODE	CHARACTER	2	
3	ATT_CODE	CHARACTER	2	

The Feature Polygon Vector Table - ALINE.DBF

FIELD	FIELD NAME	TYPE	WIDTH	DECIMAL PLACES
1	X	NUMERIC	10	3
2	Y	NUMERIC	10	3
3	LINE_ID	NUMERIC	5	
4	ATTRIB_ID	NUMERIC	4	

The Contour Vector Table - CONTLINE.DBF

FIELD	FIELD NAME	TYPE	WIDTH	DECIMAL PLACES
1	X	NUMERIC	10	3
2	Y	NUMERIC	10	3
3	Z	NUMERIC	7	2
4	LINE_ID	NUMERIC	5	
5	CONT_ID	NUMERIC	4	

The Boulder Field Non-Spatial Property Table - BOULDER.DBF

FIELD	FIELD NAME	TYPE	WIDTH	DECIMAL PLACES
1	ATT_ID	CHARACTER	2	
2	DENSITY	NUMERIC	10	3
3	ATT_NAME	CHARACTER	30	

The Megaripple Field Non-Spatial Property Table - MRIP.DBF

FIELD	FIELD NAME	TYPE	WIDTH	DECIMAL PLACES
1	ATT_ID	CHARACTER	2	
2	WAVE_DIR	NUMERIC	10	3
3	WAVE_HEIGHT	NUMERIC	10	3
4	WAVE_LEN	NUMERIC	10	3
5	ATT_NAME	CHARACTER	30	

The Pipe Route Table - PIPE.DBF

FIELD	FIELD NAME	TYPE	WIDTH	DECIMAL PLACES
1	PIPE_ID	NUMERIC	3	
2	X	NUMERIC	10	3
3	Y	NUMERIC	10	3

The Parameters Table - PARAMS.DBF

FIELD	FIELD NAME	TYPE	WIDTH	DECIMAL PLACES
1	X_CELL	NUMERIC	6	
2	Y_CELL	NUMERIC	6	
3	CELL_SIZE	NUMERIC	10	3
4	X_ORIGIN	NUMERIC	10	3
5	Y_ORIGIN	NUMERIC	10	3

The Pipeline Feature Clash Result Table - SEGMENT.DBF

FIELD	FIELD NAME	TYPE	WIDTH	DECIMAL PLACES
1	PIPE_ID	NUMERIC	3	
2	SEG_ID	NUMERIC	4	
3	ATTRIB_ID	CHARACTER	2	
4	START_CH	NUMERIC	10	3
5	FINISH_CH	NUMERIC	10	3

The Long Section Results Table - DTMOUT.DBF

FIELD	FIELD NAME	TYPE	WIDTH	DECIMAL PLACES
1	PIPE_ID	NUMERIC	3	
2	SEG_ID	NUMERIC	4	
3	ELEVATION	NUMERIC	7	2
4	START_CH	NUMERIC	10	3
5	FINISH_CH	NUMERIC	10	3

G. THE FULL PIRATE FRAME HIERARCHY

The following is the full PIRATE frame hierarchy in it's Lisp form. Instances that are necessary for the working of PIRATE have also been included. GoldWorksII only provides the Lisp format for hard copy output of the frame hierarchy. The Plates in Chapter 7 indicate the structural relationship of the more important frames in tree form, which is more simple to follow than the Lisp given here. This appendix, then, is only intended for the reader with a sound grasp of GoldWorksII and Lisp.

```
(DEFINE-FRAME MODEL-ATTRIBUTE
   ()
   (ATTRIBUTE-ID :EXPLANATION-STRING "the value for the attribute-id
is read in from the feature database and corresponds to the attribute
representation in the spatial database")
   (NAME :EXPLANATION-STRING "the name is used by the base instances
to construct the instance names. It is a system slot")
   (MIN-X-Y :EXPLANATION-STRING "This slot is not in current use")
   (MAX-X-Y :EXPLANATION-STRING "this slot is not in current use")
   (POLYGON-COORD-LIST :DEFAULT-VALUES ((NIL NIL))
    :CONSTRAINTS (:LISP-TYPE LIST)
    :EXPLANATION-STRING "The polygon coordinates are obtained from
the ALINE.DBF database during initialisation")
   (DATA-SOURCE-AND-ACCURACY :CONSTRAINTS (:LISP-TYPE STRING)
    :EXPLANATION-STRING "this value is user derived")
   (DATABASE-NAME :CONSTRAINTS (:LISP-TYPE STRING)
    :EXPLANATION-STRING "the database name is set at the time of
database construction and should not be changed")
   (BASE-INSTANCE :DEFAULT-VALUES (NO)
    :CONSTRAINTS (:ONE-OF (YES NO NEW))
    :EXPLANATION-STRING "For each model attribute frame there is a
single base instance which holds the control for all subsequent
instance creation")
   (DB-SLOT-LIST :CONSTRAINTS (:LISP-TYPE LIST)
    :EXPLANATION-STRING "The values here give the slots corresponding
to the database field structure. Change if the database fields are
changed")
   (TEXT-NAME :EXPLANATION-STRING "A pretty name that can be set by
the user")
   (REMEDIAL-ACTION :MULTIVALUED T
    :EXPLANATION-STRING "The action needed to lay a pipeline through
the feature. Found by the rules")
   (DATA-1 :MULTIVALUED T
    :EXPLANATION-STRING "A data field to augment the remedial action
description"
    :when-modified (remedial-action-dependency-daemon))
;   (DATA-2 :MULTIVALUED T
;    :EXPLANATION-STRING "A data field to augment the remedial action
description")
;   (DATA-3 :MULTIVALUED T
;    :EXPLANATION-STRING "A data field to augment the remedial action
description")
)


(DEFINE-FRAME MAN-MADE-FEATURE
   (:IS MODEL-ATTRIBUTE)
   (TYPE :CONSTRAINTS (:ONE-OF (NOT-IN-CURRENT-USE ABANDONED IN-
CURRENT-USE)))
   (OWNER)
   (SURFACE-BURIED :CONSTRAINTS (:ONE-OF (SURFACE BURIED)))
   (BURIAL-DEPTH)
   (PERMISSION-TO-CUT :CONSTRAINTS (:ONE-OF (YES NO CONDITIONAL)))
   (PERMISSION-TO-CROSS :CONSTRAINTS (:ONE-OF (YES NO CONDITIONAL)))
   (USE :CONSTRAINTS (:ONE-OF (OIL GAS TELECOM OTHER)))
   (COST-TO-CUT)
   (COST-TO-CROSS))


(DEFINE-FRAME LINEAR-FEATURE
   (:IS MODEL-ATTRIBUTE))


(DEFINE-FRAME PIPELINES
   (:IS (MAN-MADE-FEATURE LINEAR-FEATURE)))


(DEFINE-FRAME CABLES
   (:IS (MAN-MADE-FEATURE LINEAR-FEATURE)))


(DEFINE-FRAME POINT-FEATURE
   (:IS MODEL-ATTRIBUTE)
   (HEIGHT-ABOVE-BED)
   (MIN-RADIUS-FOR-AVOIDANCE))


(DEFINE-FRAME WELLHEADS
   (:IS (MAN-MADE-FEATURE POINT-FEATURE)))


(DEFINE-FRAME WRECKS
   (:IS (MAN-MADE-FEATURE POINT-FEATURE))
   (CARGO-CARRIED :CONSTRAINTS (:ONE-OF (SAFE UNSAFE UNKNOWN)))
   (SIZE-IN-TONNES))


(DEFINE-FRAME WAVE-BASED-FEATURE
   (:IS MODEL-ATTRIBUTE)
   (WAVELENGTH :EXPLANATION-STRING "The wavelength of the waveform in
metres. Set by the user at the CAD interface when specifying the
polygon it relates to")
   (FREQUENCY :EXPLANATION-STRING "Frequency in waves per unit
time/distance may be used instead of the wavelength. User specified")
   (WAVE-HEIGHT :EXPLANATION-STRING "The trough to crest height in
metres. User specified at CAD interface")
   (WAVE-DIRECTION :EXPLANATION-STRING "Bearing in degrees of wave
travel direction with respect to grid North. User specified at CAD
interface."))
```

```
(DEFINE-FRAME BEDFORM-FEATURE
   (:IS WAVE-BASED-FEATURE))


(DEFINE-FRAME SAND-RIDGES
   (:IS BEDFORM-FEATURE))


(DEFINE-FRAME SAND-WAVES
   (:IS BEDFORM-FEATURE))


(DEFINE-FRAME SAND-RIBBONS
   (:IS BEDFORM-FEATURE))


(DEFINE-FRAME MEGGARIPPLES
   (:IS BEDFORM-FEATURE)
   (TEXT-NAME :DEFAULT-VALUES ("Megaripples"))
   (DB-SLOT-LIST :DEFAULT-VALUES ((ATTRIBUTE-ID FREQUENCY WAVE-
DIRECTION WAVE-HEIGHT WAVELENGTH NAME)))
   (DATABASE-NAME :DEFAULT-VALUES ("c:\\windows\\db\\mrip.dbc"))
   (TRENCH-DEPTH :EXPLANATION-STRING "Depth of the trench required to
overcome obstacle. Set by rules"))


(DEFINE-FRAME ENVIRONMENTAL-FEATURE
   (:IS MODEL-ATTRIBUTE)
   (FREQUENCY-OF-OCCURRANCE))


(DEFINE-FRAME FLOW-FEATURE
   (:IS MODEL-ATTRIBUTE)
   (DIRECTION)
   (VELOCITY))


(DEFINE-FRAME CURRENT
   (:IS (ENVIRONMENTAL-FEATURE FLOW-FEATURE)))


(DEFINE-FRAME WIND
   (:IS (ENVIRONMENTAL-FEATURE FLOW-FEATURE))
   (POLYGON-COORD-LIST :CONSTRAINTS (:LISP-TYPE LIST)))


(DEFINE-FRAME TIDES
   (:IS (ENVIRONMENTAL-FEATURE FLOW-FEATURE))
   (TIDAL-RANGE)
   (HIGH-LOW-WATER-TIME-INTERVAL))


(DEFINE-FRAME WAVES
   (:IS (ENVIRONMENTAL-FEATURE WAVE-BASED-FEATURE)))


(DEFINE-FRAME SLOPE-FEATURE
   (:IS MODEL-ATTRIBUTE)
   (SLOPE-DIRECTION)
   (SLOPE-VALUE)
   (SLOPE-LENGTH))


(DEFINE-FRAME SLOPE-GREATER-THAN-3
   (:IS SLOPE-FEATURE))


(DEFINE-FRAME SLOPE-GREATER-THAN-6
   (:IS SLOPE-FEATURE))


(DEFINE-FRAME SOIL-TYPE-FEATURE
   (:IS MODEL-ATTRIBUTE)
   (TOP-BOUNDARY-DEPTH)
   (BOTTOM-BOUNDARY-DEPTH)
   (SOIL-TYPE :CONSTRAINTS (:INSTANCE-OF SOIL)))


(DEFINE-FRAME SURFICIAL-SOIL-TYPE
   (:IS SOIL-TYPE-FEATURE))


(DEFINE-FRAME UNDERLYING-SOIL-TYPE
   (:IS SOIL-TYPE-FEATURE))


(DEFINE-FRAME FISHING-GROUND
   (:IS MODEL-ATTRIBUTE))


(DEFINE-FRAME BOULDER-FIELD
   (:IS MODEL-ATTRIBUTE)
   (TEXT-NAME :DEFAULT-VALUES ("Boulder Fields"))
   (DB-SLOT-LIST :DEFAULT-VALUES ((ATTRIBUTE-ID DENSITY-OF-BOULDERS
NAME)))
   (DATABASE-NAME :DEFAULT-VALUES ("c:\\windows\\db\\boulder.dbc"))
   (DENSITY-OF-BOULDERS :EXPLANATION-STRING "The density of boulders
per 100 sq. metres. Set by user at CAD interface")
   (AV-SIZE-OF-BOULDERS :EXPLANATION-STRING "The average size of the
boulders (tonnes). NOT CURRENTLY SET AT CAD INTERFACE!")
   (ROCK-TYPE :EXPLANATION-STRING "The rock type of the boulders. NOT
CURRENTLY SET AT CAD INTERFACE!"))


(DEFINE-FRAME DEPTH-LIMIT
   (:IS MODEL-ATTRIBUTE)
   (DEPTH-LIMIT-MIN)
   (DEPTH-LIMIT-MAX)
   (WORST-DEPTH))


(DEFINE-FRAME DEPTH-MAXIMUM
   (:IS DEPTH-LIMIT)
   (LIMITING-DEPTH))
```

```
(DEFINE-FRAME DEPTH-MINIMUM
   (:IS DEPTH-LIMIT))


(DEFINE-FRAME PROPOSED-PIPELINE
   ()
   (PIPE-ID)
   (PIPE-NAME)
   (ORDERED-SEGMENT-LIST)
   (SEGMENT-DIR-LIST)
   (VALIDITY)
   (ASSESSMENT-STATUS :DEFAULT-VALUES (:NOT-VALID)
    :CONSTRAINTS (:ONE-OF (:NOT-ASSESSED :ASSESSED :NOT-VALID)))
   (VALIDITY-LIST)
   (global-chain-list))


(DEFINE-FRAME PIPELINE-SEGMENT
   ()
   (PIPE-ID :DEFAULT-VALUES (1))
   (SEGMENT-NUMBER-WITHIN-PIPELINE)
   (START-COORD :DEFAULT-VALUES (NIL))
   (FINISH-COORD :DEFAULT-VALUES (NIL))
   (START-CHAINAGE)
   (FINISH-CHAINAGE)
   (START-CHAINAGE-OF-ATTRIB-HIT :CONSTRAINTS (:LISP-TYPE LIST))
   (FINISH-CHAINAGE-OF-ATTRIB-HIT :CONSTRAINTS (:LISP-TYPE LIST))
   (DISTANCE-OF-ATTRIB-HIT :CONSTRAINTS (:LISP-TYPE LIST))
   (ATTRIB-HIT :EXPLANATION-STRING "This slot gives a list of
attributes in the order they are hit along the pipeline. The other
slots of start/finish chainages etc relate to this list"
    :CONSTRAINTS (:LISP-TYPE LIST))
   (M)
   (C)
   (SEGMENT-ID)
   (GRAPHIC-HOTSPOT-1)
   (GRAPHIC-HOTSPOT-2)
   (GRAPHIC-WINDOW)
   (ORIGIN :CONSTRAINTS (:ONE-OF (CAD-DATABASE GW-INTERACTIVE BOTH)))
   (ORIGINAL-START-COORD)
   (ORIGINAL-FINISH-COORD)
   (ATTRIB-VALIDITY :DEFAULT-VALUES (NOT-GENERATED)
    :CONSTRAINTS (:ONE-OF (VALID NOT-VALID NOT-GENERATED))
    :when-modified (segment-dependency-daemon))
   (PIPELINE-MEMBERS :DEFAULT-VALUES (NIL)
    :CONSTRAINTS (:LISP-TYPE LIST))
   (A-LIST)
   (A-START)
   (A-FINISH)
   (A-DIST)
   (ATTRIB-MULTI-INST :CONSTRAINTS (:INSTANCE-OF MODEL-ATTRIBUTE)
    :MULTIVALUED T)
   (consolidated-remedial-actions)
   (long-section-list))


 (DEFINE-FRAME SOILS
   ()
   (SOIL-NAME)
   (SHEAR-STRENGTH)
   (SUBMERGED-DENSITY)
   (LIQUIFACTION-POSSIBLE))


(DEFINE-FRAME DATABASE-STATUS
   ()
   (SEGMENT-NUMBER :DEFAULT-VALUES (0))
   (RECORD-STATUS :DEFAULT-VALUES (FIRST)
    :CONSTRAINTS (:ONE-OF (NOT-READ-YET READ FIRST)))
   (DB-STATUS :DEFAULT-VALUES (CLOSED)
    :CONSTRAINTS (:ONE-OF (OPEN CLOSED FINISHED)))
   (DATABASE-NAME :CONSTRAINTS (:LISP-TYPE STRING))
   (PARENT-FRAME-NAME :CONSTRAINTS NIL))


(DEFINE-FRAME PIPE-DBF-STATUS
   (:IS DATABASE-STATUS)
   (DATABASE-NAME :DEFAULT-VALUES ("c:\\windows\\db\\pipe.dbf"))
   (TOP/TAIL-STATUS :DEFAULT-VALUES (NOT-READY)
    :CONSTRAINTS (:ONE-OF (NOT-READY READY DONE))))


(DEFINE-FRAME POLYGON-DBF-STATUS
   (:IS DATABASE-STATUS))


(DEFINE-FRAME CLASH-DBF-STATUS
   (:IS DATABASE-STATUS))

(DEFINE-FRAME long-section-DBF-STATUS
   (:IS DATABASE-STATUS))


(DEFINE-FRAME GRID-PARAMETERS
   ()
   (X-ORIGIN :DEFAULT-VALUES (0))
   (Y-ORIGIN :DEFAULT-VALUES (0))
   (CELL-SIZE :DEFAULT-VALUES (0))
   (X-EXTENT :DEFAULT-VALUES (0))
   (Y-EXTENT :DEFAULT-VALUES (0)))


(DEFINE-FRAME GRAPHIC-INTERFACE-CONTROL
   ()
   (ZOOM-SCALE)
   (SCROLL-SPEED)
   (HIGHLIGHT-COLOR-CHOSEN)
   (HIGHLIGHT-COLOR-OPTIONS :MULTIVALUED T)
   (HIGHLIGHT-COLORS :MULTIVALUED T)
   (GRAPHIC-WINDOW)
   (GRID-STATUS :DEFAULT-VALUES (OFF)
    :CONSTRAINTS (:ONE-OF (ON OFF))))


(DEFINE-FRAME PIPELINE-SEGMENT-CONTROL
   ()
   (LATEST-SEGMENT-ID)
   (NAME)
   (TEXT-NAME)
   (CURRENT-SEGMENT-NUMBER)
   (SELECTED-SEGMENT-LIST :DEFAULT-VALUES (NIL))
   (CURRENT-PIPE-NUMBER :DEFAULT-VALUES (1)))


(DEFINE-FRAME PIPELINE-CONTROL
   ()
   (TEXT-NAME :DEFAULT-VALUES ("pipeline"))
   (CURRENT-SEGMENT-NUMBER :DEFAULT-VALUES (1)))


(DEFINE-FRAME PLANT
   ()
   (EQPT-NAME)
   (MOBILISATION-COST)
   (DAY-COST)
   (%-DOWNTIME)
   (DAY-OUTPUT-RATE)
   (DAY-OUTPUT-UNIT :CONSTRAINTS NIL))


(DEFINE-FRAME SEA-GOING-VESSELS
   (:IS PLANT)
   (DEPTH-MAX)
   (DEPTH-MIN)
   (SEASON-START)
   (SEASON-FINISH)
   (AVAILABILITY-EARLIEST)
   (AVALIABILITY-DURATION)
   (VESSEL-TYPE))


(DEFINE-FRAME LAY-VESSELS
   (:IS SEA-GOING-VESSELS)
   (DAY-OUTPUT-UNIT :DEFAULT-VALUES (LINEAR-METRES)))


(DEFINE-FRAME TRENCHERS
   (:IS SEA-GOING-VESSELS)
   (DAY-OUTPUT-UNIT :DEFAULT-VALUES (CUBIC-METRES))
   (max-cutting-depth))


(DEFINE-FRAME DREDGERS
   (:IS SEA-GOING-VESSELS)
   (DAY-OUTPUT-UNIT :DEFAULT-VALUES (CUBIC-METRES)))


(DEFINE-FRAME DIVE-SUPPORT-VESSELS
   (:IS SEA-GOING-VESSELS)
   (DAY-OUTPUT-UNIT))


(DEFINE-FRAME ROCK-DUMP-VESSELS
   (:IS SEA-GOING-VESSELS)
   (DAY-OUTPUT-UNIT :DEFAULT-VALUES (TONNES)))


(DEFINE-FRAME SURVEY-VESSELS
   (:IS SEA-GOING-VESSELS)
   (SURVEY-SPEED))


(DEFINE-FRAME OTHER-EQPT
   (:IS PLANT))


(DEFINE-FRAME UNMANNED-SUBMERSIBLES
   (:IS OTHER-EQPT))


(DEFINE-FRAME MANNED-SUBMERSIBLES
   (:IS OTHER-EQPT))


(DEFINE-FRAME TRENCHING-PLOUGHS
   (:IS OTHER-EQPT))


(DEFINE-FRAME MATERIALS
   ()
   (UNIT-MEASURE :CONSTRAINTS (:ONE-OF (UNIT-ITEM LINEAR-METRE
SQUARE-METRE CUBIC-METRE METRIC-TONNE)))
   (COST-PER-UNIT)
   (NAME)
   (WEIGHT-PER-UNIT)
   (SUPPLIERS :MULTIVALUED T)
   (MATERIAL-NAME))


(DEFINE-FRAME PIPE
   (:IS MATERIALS)
   (DIAMETER :explanation-string "Although commonly referred to in
inches, the diameter MUST be input in metres")
   (WALL-THICKNESS)
   (MATERIAL-TYPE)
   (SPOOL-LENGTH))


(DEFINE-FRAME CONCRETE-COAT
   (:IS MATERIALS))


(DEFINE-FRAME ROCK-DUMP
   (:IS MATERIALS))


(DEFINE-FRAME SACRIFICIAL-ANODE
   (:IS MATERIALS))


(DEFINE-FRAME VALVE
   (:IS MATERIALS))


(DEFINE-FRAME PIPELINE-TALLY
   ())


(DEFINE-FRAME PIPE-SEGMENT-TALLY
   ())


(DEFINE-FRAME LOGISTIC-CONTROL
   ()
   (LOTUS-ROW)
   (LOTUS-COL)
   (LOTUS-END-ROW)
   (LOTUS-END-COL)
   (SPREADSHEET)
   (ORDERED-SLOT-LIST)
   (STATUS :DEFAULT-VALUES (READ)
    :CONSTRAINTS (:ONE-OF (READ NOT-READ))))


(DEFINE-FRAME PLANT-CONTROL
   (:IS LOGISTIC-CONTROL)
   (SPREADSHEET :DEFAULT-VALUES ("c:/gw2/lee/plant.wrl"))
   (NAME))


(DEFINE-FRAME MATERIAL-CONTROL
   (:IS LOGISTIC-CONTROL)
   (LOTUS-ROW :DEFAULT-VALUES (NIL))
   (LOTUS-END-ROW :DEFAULT-VALUES (NIL))
   (LOTUS-END-COL :DEFAULT-VALUES (NIL))
   (LOTUS-COL :DEFAULT-VALUES (NIL))
   (SPREADSHEET :DEFAULT-VALUES ("c:/gw2/lee/material.wrl"))
   (NAME))


(DEFINE-FRAME REPORT-CONTROLLER
   (:DOC-STRING "The report controller is the fundamental part of the
analysis of the pipeline. The daemons attached to these slots react
to a REQUIRED entry, creating, firing and retracting attempts to
produce the respective reports")
   (SEGMENT-CLASH-REPORT :WHEN-MODIFIED (SEGMENT-CLASH-REPORT-DAEMON)
    :DEFAULT-VALUES (NOT-GENERATED)
    :CONSTRAINTS (:ONE-OF (COMPLETE REQUIRED NOT-GENERATED))
    :EXPLANATION-STRING "This slot controls the generation of a
feature clash report for the selected segment. Setting the slot to
REQUIRED activates the daemon.")
   (SEGMENT-TECH-REPORT :WHEN-MODIFIED (SEGMENT-TECH-REPORT-DAEMON)
    :DEFAULT-VALUES (NOT-GENERATED)
    :CONSTRAINTS (:ONE-OF (COMPLETE REQUIRED NOT-GENERATED))
    :EXPLANATION-STRING "This slot controls the generation of the
technical report for the clashes of the chosen segment. Setting the
value to REQUIRED activates the daemon.")
   (SEGMENT-PLANT-REPORT :WHEN-MODIFIED (SEGMENT-PLANT-REPORT-DAEMON)
    :DEFAULT-VALUES (NOT-GENERATED)
    :CONSTRAINTS (:ONE-OF (COMPLETE REQUIRED NOT-GENERATED))
    :EXPLANATION-STRING "This slot controls the generation of hte
plant and materials report for the chosen segment. Setting the value
to REQUIRED activates the daemon/")
   (SEGMENT-COST-REPORT :WHEN-MODIFIED (SEGMENT-COST-REPORT-DAEMON)
    :DEFAULT-VALUES (NOT-GENERATED)
    :CONSTRAINTS (:ONE-OF (COMPLETE REQUIRED NOT-GENERATED))
    :EXPLANATION-STRING "this slot controls the generation of a cost
report for the chosen segment. Setting the value to REQUIRED
activates the daemon.")
   (PIPELINE-CLASH-REPORT :DEFAULT-VALUES (NOT-GENERATED)
    :CONSTRAINTS (:ONE-OF (REQUIRED COMPLETE NOT-GENERATED))
    :WHEN-MODIFIED (PIPELINE-CLASH-REPORT-DAEMON)
    :EXPLANATION-STRING "The control slot for pipeline/attribute
clash reports. Can be set to REQUIRED from anywhere in the system, to
initiate the report.")
   (PIPELINE-TECH-REPORT :WHEN-MODIFIED (PIPELINE-TECH-REPORT-DAEMON)
    :DEFAULT-VALUES (NOT-GENERATED)
    :CONSTRAINTS (:ONE-OF (REQUIRED COMPLETE NOT-GENERATED))
    :EXPLANATION-STRING "This is the slot that controls the
generation of the pipeline technical report (ie. the remedial works
required etc.). Setting the report to REQUIRED activates the
daemon.")
   (PIPELINE-PLANT-REPORT :WHEN-MODIFIED (PIPELINE-PLANT-REPORT-
DAEMON)
    :DEFAULT-VALUES (NOT-GENERATED)
    :CONSTRAINTS (:ONE-OF (REQUIRED COMPLETE NOT-GENERATED))
    :EXPLANATION-STRING "this slot controls the generation of a
plant/materials report for the selected pipeline. Setting the value
to REQUIRED activates the daemon.")
   (PIPELINE-COST-REPORT :WHEN-MODIFIED (PIPELINE-COST-REPORT-DAEMON)
    :DEFAULT-VALUES (NOT-GENERATED)
    :CONSTRAINTS (:ONE-OF (REQUIRED COMPLETE NOT-GENERATED))
    :EXPLANATION-STRING "This slot controls the generation of a cost
report for the selected pipeline. Setting the value to REQUIRED will
initiate the daemon.")
   (CURRENT-PIPELINE :CONSTRAINTS (:INSTANCE-OF proposed-pipeline)
    :EXPLANATION-STRING "This is the instance name of the pipeline
currently being evaluated by the system. It is instantiated when the
current pipeline is highlighted in the graphics interface")
   (CURRENT-SEGMENT :CONSTRAINTS (:INSTANCE-OF PIPELINE-SEGMENT)
    :EXPLANATION-STRING "This is the instance name of the pipeline
segment under consideration. It's value comes either from pipeline
analysis rules, or directly by highlighting the appropriate segment
at the graphics interface."))


(DEFINE-FRAME SEGMENT-TECHNICAL-REPORT
   ()
   (TRENCH-DEPTH-MAX :EXPLANATION-STRING "The maximum depth of trench
needed so far on the segment, which does not exceed the max. single
pass depth.")
   (TRENCH-DEPTH-ALLOWABLE :EXPLANATION-STRING "This is the maximum
depth that the trencher can dig to in a single pass. The value can
either be set by the user or gained from the plant tables.")
   (SEGMENT-LENGTH :EXPLANATION-STRING "The length of the segment in
metres. Calculated from the original coordinates of the endpoints of
the pipe segment.")
```

```
   (SEGMENT :EXPLANATION-STRING "The name of the pipeline-segment to
which this report refers. Used for identification purposes."
    :CONSTRAINTS (:INSTANCE-OF PIPELINE-SEGMENT))
   (DREDGE-LENGTH :EXPLANATION-STRING "the length of the segment over
which dredging work is required.")
   (DREDGE-DEPTH :EXPLANATION-STRING "The depth below the sea bed
which the final dredged surface needs to be. ")
   (DREDGE-VOLUME :EXPLANATION-STRING "An estimate of the volume of
dredged material that needs to be shifted for this segment. Found
either using the segment-profile, or by using the dredge length and
average dredge depth.")
   (LAY-BARGE-TYPE :EXPLANATION-STRING "States whether the lay barge
required is a reel vessel or a standard lay vessel. This slot could
be moved to a global parameter frame, but the type can be set by the
user or inferred from the pipe size.")
   (DIVER-SUPPORT-LENGTH :EXPLANATION-STRING "The length of pipeline
which requires diver interaction, either for placing mats for
crossings, clearance of obstacles etc. Note that the time needed
depends on the problem and so time may have to be assessed
separately.")
   (BLASTING-LENGTH :EXPLANATION-STRING "The pipe length through
which blasting is required.")
   (BOULDER-CLEARING-LENGTH :EXPLANATION-STRING "The pipe length
through which boulder clearance has to take place (by blasting or
brute force).")
   (BOULDER-NUMBER-PER-100M :EXPLANATION-STRING "An estimate of the
number of boulders which need clearing per 100 metre run on this
section of pipeline. Found using the boulder density, length etc.")
   (PRE-SWEEP-LENGTH :EXPLANATION-STRING "The length of the pipeline
over which pre-sweeping is required. ")
   (TRENCH-DEEP-LENGTH :EXPLANATION-STRING "This is the length of the
pipeline segment over which deep trenching is needed. Deep trenching
is such that the trenching vessel has to make more than one pass to
achieve the depth required.")
   (TRENCH-DEEP-DEPTH :EXPLANATION-STRING "This slot gives the
maximum depth needed for deep trenching. Deep trenching is trenching
in excess of the allowable depth for trenching with a single pass of
the trenching vessel (given in the TRENCH-DEPTH-ALLOWABLE slot).")
   (STABILITY-PROBLEM-LENGTH :EXPLANATION-STRING "Gives the length of
the segment over which there may be problems with pipeline stability
using the standard configuration. This may indicate that a heavier
concrete coat is required, or that the pipeline may need to be
secured or trenched.")
   (BED-PROFILE-VECTOR-LIST :EXPLANATION-STRING "This slot stores the
bed profile along the pipeline in terms of a nested list of x,z
coordinates. Obtained from the GIS DTM.")
   (TRENCH-LENGTH :EXPLANATION-STRING "The length of the pipeline
which requires trenching at a single pass. Multiple pass trenching
lengths are held in the DEEP slots.")
   (CURR-ACTION :EXPLANATION-STRING "This is the remedial action
currently under consideration for inclusion in the segment technical
report. The data fields and length over which the action is required
is included in the other CURR- slots. Set from the matching on the
multivalued slot in the pipeline-segment instance.")
   (CURR-DATA-1 :EXPLANATION-STRING "The first data field for the
current remedial action")
   (CURR-DATA-2 :EXPLANATION-STRING "The second data field for the
current remedial action")
```

```
   (CURR-DATA-3 :EXPLANATION-STRING "The third data field for the
current remedial action")
   (CURR-LENGTH :EXPLANATION-STRING "The length of pipeline segment
over which the current remedial action acts."))


(DEFINE-INSTANCE GENERAL-REPORT
  (:IS OUTPUT-WINDOW)
 (MESSAGE-LINE "GENERAL REPORT")
 (Y 50)
 (X 50)
 (WIDTH 500)
 (HEIGHT 300))


(DEFINE-INSTANCE NODE-DESCRIPTION-1
  (:IS OUTPUT-WINDOW)
 (TITLE "Node Description")
 (Y 40)
 (X 13)
 (WIDTH 535)
 (HEIGHT 246))


(DEFINE-INSTANCE SCROLLING-REPORT
  (:IS OUTPUT-WINDOW)
 (CLEAR-BEFORE-NEW-DISPLAY :NO)
 (MESSAGE-LINE "DETAILED REPORT")
 (Y -11)
 (X -11)
 (WIDTH 654)
 (HEIGHT 158))


(DEFINE-INSTANCE SEGMENT-DESCRIPTION-1
  (:IS OUTPUT-WINDOW)
 (TITLE "Segment Description")
 (Y 80)
 (X 150)
 (WIDTH 499)
 (HEIGHT 299))


(DEFINE-INSTANCE GRAPHIC-ERROR-WINDOW-1
  (:IS POPUP-CONFIRM)
 (ANSWER :YES)
 (INSTRUCTIONS)
 (TEXT-ATTRIBUTES
   (:COLOR :RED))
 (TRIGGER-INFERENCING :NO)
 (CANCEL-BUTTON :NO))


(DEFINE-INSTANCE GRAPHIC-MESSAGE-WINDOW-1
  (:IS POPUP-CONFIRM)
 (ANSWER :YES)
 (INSTRUCTIONS)
   (TEXT-ATTRIBUTES
   (:COLOR :RED))
 (TRIGGER-INFERENCING :NO))


(DEFINE-INSTANCE CHOOSE-HIGHLIGHT-COLOR
  (:IS POPUP-CHOOSE)
 (ELEMENTS
   ((BLUE *BLUE-BRUSH*)
    (RED *RED-BRUSH*)
    (GREEN *GREEN-BRUSH*)
    (YELLOW *YELLOW-BRUSH*)))
 (ANSWER *YELLOW-BRUSH*)
 (INSTRUCTIONS "Choose highlight color")
 (USER-BUTTONS
   ((:BUTTON-TEXT "KILL DISK" :EVALUATE
      (PRIN1 "No I won't !!")))))


(DEFINE-INSTANCE JOIN-SEGMENT-CHOICE-1
  (:IS POPUP-CHOOSE)
 (ELEMENTS
   (("Start Point Only" 1)
    ("Both Start And Endpoint" 2)))
 (ANSWER 1)
 (INSTRUCTIONS
    "One or both endpoints EXISTING? Probe EXISTING first")
 (TRIGGER-INFERENCING :NO))


(DEFINE-INSTANCE POPUP-ASK-FOR-ZOOM-SCALE
  (:IS POPUP-ASK-USER)
 (REMEMBER-LAST-ANSWER :YES)
 (DEFAULT-ANSWER 1.8)
 (LAST-ANSWER 4.2)
 (ANSWER 4.2)
 (TARGET-INSTANCE INTERFACE-CONTROL-1)
 (TARGET-SLOT ZOOM-SCALE)
 (INSTRUCTIONS "Please input new zoom scale (1.0 to 5.0)")
 (TEXT-ATTRIBUTES
   (:COLOR :BLUE))
 (TRIGGER-INFERENCING :NO)
 (CANCEL-BUTTON :NO)
 (BACKGROUND-COLOR :YELLOW))


(DEFINE-INSTANCE BIGCELL-DBF
  (:IS DBASE-ACTION)
 (FILE-NAME "c:\\windows\\db2\\bigcell.dbf"))


(DEFINE-INSTANCE COMBCODE-ATT-DBF
  (:IS DBASE-ACTION)
 (FILE-NAME "c:\\windows\\db\\combcode.dbf")
 (INDEX-FILE-NAME "c:\\windows\\db\\cc_att.ndx")
 (KEY-FIELD-NAMES "att_code"))


(DEFINE-INSTANCE COMBCODE-CURR-DBF
  (:IS DBASE-ACTION)
 (FILE-NAME "c:\\windows\\db\\combcode.dbf")
 (INDEX-FILE-NAME "c:\\windows\\db\\cc_curr.ndx")
 (KEY-FIELD-NAMES "curr_code"))


(DEFINE-INSTANCE COMBCODE-PREV-DBF
  (:IS DBASE-ACTION)
 (FILE-NAME "c:\\windows\\db\\combcode.dbf")
 (INDEX-FILE-NAME "c:\\windows\\db\\cc_prev.ndx")
 (KEY-FIELD-NAMES "prev_code"))


(DEFINE-INSTANCE PARAMS-DBF
  (:IS DBASE-ACTION)
 (ACTION :OPEN-FILE)
 (FILE-NAME "c:\\windows\\db\\params.dbf")
 (ERROR-CODE NIL))


(DEFINE-INSTANCE PIPE-DBF
  (:IS DBASE-ACTION)
 (ACTION :CLOSE-FILE)
 (FILE-NAME "c:\\windows\\db\\pipe.dbf")
 (ERROR-CODE NIL))


(DEFINE-INSTANCE SEGMENT-DBF
  (:IS DBASE-ACTION)
 (ACTION :CLOSE-FILE)
 (FILE-NAME "c:\\windows\\db\\segment.dbf")
 (ERROR-CODE NIL))

(DEFINE-INSTANCE DTMOUT-DBF
  (:IS DBASE-ACTION)
 (ACTION :CLOSE-FILE)
 (FILE-NAME "c:\\windows\\db\\dtmout.dbf")
 (ERROR-CODE NIL))


(DEFINE-INSTANCE 123-MATERIAL
  (:IS 123-ACTION)
 (SPREADSHEET-NAME "c:/gw2/lee/material.wr1"))


(DEFINE-INSTANCE 123-PLANT
  (:IS 123-ACTION)
 (ACTION :READ-VALUE)
 (VALUE)
 (COLUMN B)
 (ROW 5)
 (END-COLUMN NIL)
 (END-ROW NIL)
 (SPREADSHEET-NAME "c:/gw2/lee/plant.wr1")
 (ERROR-CODE NIL))


(DEFINE-INSTANCE MR1
  (:IS MEGGARIPPLES)
 (ATTRIBUTE-ID 0)
 (NAME MEGGARIPPLE)
 (BASE-INSTANCE YES))


(DEFINE-INSTANCE BF1
  (:IS BOULDER-FIELD)
 (ATTRIBUTE-ID 0)
 (NAME BOULDER)
 (BASE-INSTANCE YES))


(DEFINE-INSTANCE PIPE-STATUS-1
  (:IS PIPE-DBF-STATUS))


(DEFINE-INSTANCE POLYGON-STATUS-1
  (:IS POLYGON-DBF-STATUS)
 (DATABASE-NAME "c:\\windows\\db\\aline.dbc"))


(DEFINE-INSTANCE CLASH-STATUS-1
  (:IS CLASH-DBF-STATUS)
 (DATABASE-NAME "c:\\windows\\db\\segment.dbf"))

(DEFINE-INSTANCE LONG-SECTION-STATUS-1
  (:IS LONG-SECTION-DBF-STATUS)
 (DATABASE-NAME "c:\\windows\\db\\dtmout.dbf"))


(DEFINE-INSTANCE INTERFACE-CONTROL-1
  (:IS GRAPHIC-INTERFACE-CONTROL)
 (ZOOM-SCALE 4.2)
 (SCROLL-SPEED 5)
 (HIGHLIGHT-COLOR-OPTIONS RED YELLOW BLUE GREEN))


(DEFINE-INSTANCE PIPELINE-SEGMENT-CONTROL-1
  (:IS PIPELINE-SEGMENT-CONTROL)
 (TEXT-NAME "pipeline-segment")
 (CURRENT-SEGMENT-NUMBER 1))


(DEFINE-INSTANCE PIPELINE-CONTROL-1
  (:IS PIPELINE-CONTROL))

(DEFINE-INSTANCE EX-CARMEN
  (:IS LAY-VESSELS)
 (DEPTH-MAX 200)
 (DEPTH-MIN 10)
 (SEASON-START 9)
 (SEASON-FINISH 3)
 (AVAILABILITY-EARLIEST
   (6 90))
 (AVALIABILITY-DURATION 4)
 (EQPT-NAME "SS-CARMEN, TOTAL OIL")
 (MOBILISATION-COST 2000000)
 (DAY-COST 200000)
 (%-DOWNTIME 20)
 (DAY-OUTPUT-RATE 2500))

(DEFINE-INSTANCE GENERIC-TRENCHER
  (:IS TRENCHERS)
 (MAX-CUTTING-DEPTH 1.5)
 (DEPTH-MAX 100)
 (DEPTH-MIN 7)
 (SEASON-START 10)
 (SEASON-FINISH 3)
 (AVAILABILITY-EARLIEST
   (4 90))
 (AVALIABILITY-DURATION 36)
 (EQPT-NAME "generic-trencher")
 (MOBILISATION-COST 100000)
 (DAY-COST 25000)
 (%-DOWNTIME 30)
 (DAY-OUTPUT-RATE 2000))


(DEFINE-INSTANCE GENERIC-PIPE
  (:IS PIPE)
 (DIAMETER .33)
 (WALL-THICKNESS 20)
 (SPOOL-LENGTH 12)
 (UNIT-MEASURE LINEAR-METRE)
 (COST-PER-UNIT 1200)
 (NAME "12 inch steel pipe")
 (WEIGHT-PER-UNIT 2000)
 (MATERIAL-NAME STEEL))
```

H. Recursive Lisp Functions For Pipeline Clash Conversion

The following functions are an example of the type of Lisp coding used to decompose the combination code chainages, received as a result of pipeline clash analysis, to feature code chainages. In particular, the RESOLVE-ATTRIBS function is of note, using three branches of recursion to break down the combination code lists and re-build them as feature codes. The other functions support this activity by pre- or post-processing the lists. The notes within the program listing should be sufficient to explain the way in which each function works.

```
THE FOLLOWING FUNCTION[S] READ THE COMBINATION CODES IN THE LISTS OF
AN INDIVIDUAL PIPELINE SEGMENT, AND CONVERT THEM INTO LISTS OF
FEATURE [ATTRIBUTE] CODES AND CHAINAGES. I THINK THIS WILL BE QUITE
COMPLEX, SO I HAVE MY FINGERS CROSSED!!!!! l.j.finniear 30th January,
1990.

What we have is three lists which are co-ordinated by order. They
hold an indeterminate number of combination codes, the start chainage
of the code, and the finish chainage of the code. We need to change
this into three lists comprising the ATTRIBUTE CODE and the
associated chainages.
The three lists may not be in ascending order of chainage, but may be
backwards. I suggest that in this case the lists be reversed during
processing. Three new slots will be required for all pipeline
segments.
;
;    eg ATTRIB-ID   ("c0" "6j" "6q" "6t")
;       START-CO..  (0     21   26   35 )
;       FINISH-C,,  (21    26   35   43 )
;
Note that the finish code is (almost) superfluous. To find out the
extent of each attribute we have to first split up the comb-codes
into component attribute codes. Thus, for each entry we will have a
list of attributes which we know exist between the start and finish
coordinates.
We then need two compare successive pairs of lists to see which of
the attributes are common. We can infer that these attributes extend
over the concatenated range of the two combination codes. Those that
do not can be marked as completed for the segment. Note that multiple
separate occurrances of any attribute will be listed as such, so any
future enquiry about the extent of a feature may have to sum all
occurances.
;

;;THE TEST T2 SI THE FORM. IT WORKS !!!!! L.J. Finniear, 31st Jan,
1990

Typical input list below:
```

```
;(setq t2 '(("a1" 0 10)
;                  ("a2" 0 10)
;                  ("a3" 0 10)
;                  ("a2" 10 20)
;                  ("a3" 10 20)
;                  ("a4" 10 20)
;                  ("a5" 10 20)
;                  ("a2" 20 30)
;                  ("a1" 30 40)
;                  ("a1" 40 50)
;                  ("a2" 40 50)
;                  ("a1" 50 60)))
```

The main recursive function for changing combination code chainages in a list of the above format into individual feature code chainages is gien below. The function calls itself three times, decomposing sections of the original list until adjacent parts can be amalgamated or the root of the list is found :

```
(defun resolve-attribs (overall-list)
  (res-a overall-list (first overall-list) 0 1))

(defun res-a (o-list ml1 count-1 count-2)
  (let*((ml2 (nth count-2 o-list)))
    (cond((eq ml1 nil)
        nil)
       ((eq ml2 nil)
            (cons ml1 (res-a o-list
                         (nth (+ count-1 1) o-list)
                         (+ count-1 1)
                         (+ count-1 2))))
       ((eq (second ml1) (second ml2))
        (res-a o-list ml1 count-1 (+ count-2 1)))
       ((and (equal (first ml1) (first ml2))
             (eq (third ml1) (second ml2)))
        (setq o-list (remove ml2 o-list))
        (res-a o-list (list (first ml1)
                        (second ml1)
                        (third ml2))
             count-1
             (+ count-2 1)))
       (t
        (res-a o-list ml1 count-1 (+ 1 count-2))))))
```

```
;;;NOW WE HAVE A FUNCTION TO GET THE RESOLVED ATTRIBUTES OUT, WE NEED
SOMETHING TO PUT IT INTO THAT FORM IN THE FIRST PLACE. WE START WITH
A COMBINATION CODE LIST WITH MATCHING START AND FINISH COORDS.THE
LISTS ARE REVERSED IF REQUIRED HERE. the function returns the total
attribute/start/finish lists needed for input to the resolve-attribs
function. l.j. finniear. 1st February, 1990.
```

```
;(defun test-list ()
;  (setq comb-list '("6j" "6t" "6r" "6m"))
;  (setq start-list '(0 10 20 30))
;  (setq fin-list '(10 20 30 40)))

(defun comb-to-conversion-form (comb-list start-list finish-list)
  (let*((total-attrib-list '()))
    (cond((eq (length comb-list) 1)
        nil)
       ((> (second start-list) (first start-list))
        (setq comb-list (reverse comb-list))
        (setq start-list (reverse start-list))
        (setq finish-list (reverse finish-list)))
       (t nil))
    (let*((list-len (length comb-list)))
      (dotimes (count list-len)
      (let*((code (nth count comb-list))
            (start (nth count start-list))
            (finish (nth count finish-list))
            (attrib-element '())
            (attrib-list (combcode-to-att-list code)))
        (cond((eq attrib-list nil)
            nil)
             (t
            (dolist (element attrib-list)
              (setq att-sub-list (list element
                                 start
                                 finish))
              (setq total-attrib-list
                    (cons att-sub-list
                        total-attrib-list)))))))))
    total-attrib-list))


;;;THIS FUNCTION RULES THEM ALL, THIS FUNCTION FINDS THEM, THIS
FUNCTION BRINGS THEM ALL AND IN THE DARKNESS BINDS THEM........


(setf *comb-to-att-chainage-call-number* 0)


(defun comb-to-att-chainage (segment)
  (setf *comb-to-att-chainage-call-number*
      (+ *comb-to-att-chainage-call-number* 1))
;  (print "Comb-to-att-chainage function called ..... is this
necessary?")
;  (print "The following global variable value indicates the number
of times comb-to-att-chainage has been called in this session :")
;  (print *comb-to-att-chainage-call-number*)
  (let*((comb-list (slot-value segment 'attrib-hit))
         (start-list (slot-value segment
                              'start-chainage-of-attrib-hit))
         (finish-list
            (slot-value segment
                          'finish-chainage-of-attrib-hit))
         (conv-list (comb-to-conversion-form comb-list
                                  start-list
                                  finish-list))
         (resolved-list (resolve-attribs conv-list)))
    (dolist (element resolved-list)
      (setf (slot-value segment 'a-list)
          (cons (first element)
                  (slot-value segment 'a-list)))
      (setf (slot-value segment 'a-start)
          (cons (second element)
                  (slot-value segment 'a-start)))
      (setf (slot-value segment 'a-finish)
          (cons (third element)
                  (slot-value segment 'a-finish))))
      (setf (slot-value segment 'a-list)
          (reverse (slot-value segment 'a-list)))
      (setf (slot-value segment 'a-start)
          (reverse (slot-value segment 'a-start)))
      (setf (slot-value segment 'a-finish)
          (reverse (slot-value segment 'a-finish)))
      )
  (consolidate-attrib-lists segment))

;(comb-to-att-chainage 'pipeline-segment155)


;### The following function consolidates the lists of attributes,
start and finish chainages produced by the above function. Basically
if there are two entries with the same attrib-id, and they have
either a common start/finish or finish/start chainage value, the
entries are amalgamated into a single entry with non-common chainage
values retained.


(defun consolidate-attrib-lists (segment)
  (let*((a-list (slot-value segment 'a-list))
       (a-start (slot-value segment 'a-start))
       (a-finish (slot-value segment 'a-finish))
       (out-list '())
       (out-start '())
       (sorting-list '())
       (sorted-list '())
       (out-finish '())
       (cnt 0)
       (sort-len 0)
       (att-len (length a-list)))
;    (print " ")
;    (print a-list)
;    (print a-start)
;    (print a-finish)
;    (print " ")
    (dotimes (cnt att-len)
      (let*((al-1 (nth cnt a-list))
          (as-1 (nth cnt a-start))
          (af-1 (nth cnt a-finish)))
      (setq sorting-list (cons (list al-1 as-1 af-1)
sorting-list))))
;    (print sorting-list)
    (setf sorted-list (sort sorting-list
            #'string-lessp :key #'car))
;    (print sorted-list)
;    (print "above was the sorted list")
    (setq sort-len (length sorted-list))
    (dotimes (cnt2 sort-len)
;      (print sorted-list)
;      (print "above is the new sorted list")
      (let*((key-att (first (first sorted-list)))
          (key-list '()))
      (dolist (element sorted-list)
        (cond((equal key-att (first element))
            (setq key-list (cons element key-list)))))
;     (print key-list)
;     (print "above is the key list")
      (setq cons-key-list (produce-cons-key-list
            key-list))
;     (print cons-key-list)
;     (print "above is the produce-cons-key-list result")
      (dolist (element cons-key-list)
        (setq out-list (cons (first element) out-list))
        (setq out-start (cons (second element) out-start))
        (setq out-finish (cons (third element)
            out-finish)))
      (dolist (element key-list)
        (setq sorted-list (remove element sorted-list)))
;     (print "end of function: looping with sawn off
sorted list")
      ))
;    (print out-list)
;    (print out-start)
;    (print out-finish)
    (setf (slot-value segment 'a-list) out-list)
    (setf (slot-value segment 'a-start) out-start)
    (setf (slot-value segment 'a-finish) out-finish))
  (forward-chain))
```

```
;(consolidate-attrib-lists '|pipeline-segment-5|)
;(setq sorted-list '())
;(sort '(("a" 2 3) ("c" 4 5) ("b" 4 5)) #'string-lessp :key #'car)
;(produce-cons-key-list '((" &" 234.45 334.345)
;                   (" &" 123 234.45)
;                   (" &" 334.345 500)
;                   (" &" 600 700)
;                   (" &" 700 800)
;                   (" &" 900 1000)))
```

```
;this function produces a consolidated attribute list given a list of
the attribute occurances. It should not be referenced directly,
except by the above function. l.j. finniear 14th March 1990.

(defun produce-cons-key-list (key-list)
  (cond((= (length key-list) 1)
      key-list)
       (t
      (setf key-list (sort key-list #'< :key #'second))
;     (print "BANG!")
;     (print key-list)
      (let*((a1 " ")
            (s1 0)
            (f1 0)
            (a2 " ")
            (s2 0)
            (f2 0)
            (match-list '())
            (match-list-2 '())
            (cons-list '()))
        (cond
          ((> (length key-list) 1)
           (setq match-list (first key-list))
           (setq a1 (first match-list))
           (setq s1 (second match-list))
           (setq f1 (third match-list))
           (setq key-list (remove match-list key-list))
           (loop
             (setq match-list-2 (first key-list))
             (setq a2 (first match-list-2))
             (setq s2 (second match-list-2))
             (setq f2 (third match-list-2))
             (cond
             ((= f1 s2)
              (setq f1 f2)
              (setq key-list
                  (remove match-list-2 key-list)))
             (t
              (setq cons-list
                  (cons (list a1 s1 f1) cons-list))
              (setq a1 a2)
              (setq s1 s2)
              (setq f1 f2)
              (setq key-list
                  (remove match-list-2 key-list))))
             (cond((= (length key-list) 0)
                 (setq cons-list
                        (cons (list a1 s1 f1) cons-list))
                 (setq cons-list
                        (remove-duplicates cons-list))
                 (return cons-list)))))
          (t
           key-list))
;       (print cons-list)
        ))))


;################
```

COUNTRY HARBOUR
INDIAN HARBOUR
ST. MARYS RIVER
K
SAND RIDGES
LIMIT OF DIGITISING
M
BF2
MR1
MEGARIPPLES ON SAND RIDGES
MR3
MR2
MEGARIPPLES
2
SAND RIBBONS
1
BOULDERS 0·05-1·0/100m²
BF1
COMMUNICATION CABLE (IN SERVICE)
RELIC SAND RIBBONS
ROUTE 1
SAND RIDGES
3
MR4
SAND RIBBONS
MR5
SAND RIDGES
SAND RIBBONS
4
ROUTE 2&3
ROUTE 3
ROUTE 2
MEGARIPPLES
BF7
BF6
BF10
BF3
BF4
5
BF5
MEGARIPPLES
SAND RIBBONS
SAND RIDGES
MR7
SAND RIDGES
SAND RIBBONS
MR6
BOULDER PATCHES
MR8
SAND RIDGE HT. = 4m
LARGE SAND RIDGE
H
J
SAND, NO SEABED FEATURES
BF8
BF9
BF11
BF12
BF13
BF14
BF15
C
SAND, NO SEABED FEATURES
SAND, NO SEABED FEATURES
SAND, NO SEABED FEATURES
B
MEGARIPPLES λ = 15-25m HT = 0·5-1·5m
SAND RIBBONS λ = 70m HT. < 0·25m
MR9
D
SAND RIBBONS λ = 55m HT < 0·25m
SOME BOULDER PATCHES
SAND RIDGES, BOULDER PATCHES, SAND RIBBONS
F
SAND RIBBONS
BF16
A
A'
E
G
SABLE ISLAND
N
LIMIT OF DIGITISING
(100,50)
LEGEND
A B NO RECORDS
GEOPHYSICAL SURVEY LINES SHOWING INCOMPLETE SIDE SCAN SONAR RECORDS
ASSUMED BOULDER FIELDS
2★
KNOWN WRECKS
ABANDONED COMMUNICATIONS CABLES
ASSUMED LIMIT OF BEDFORM ACTIVITY
J P KENNY
JPK

SABLE GAS SYSTEMS LIMITED 2206·32

SEABED FEATURES & MAN MADE HAZARD AREAS WITH ALT. ROUTES INDICATED

FIGURE 8.1