UGIX: A GIS INDEPENDENT USER INTERFACE ENVIRONMENT

J.F. Raper and M.S. Bundock
Dept. of Geography,
Birkbeck College,
7-15 Gresse St., London W1P 1PA

ABSTRACT

Work has begun on the design and specification of a Standard User Interface Environment for use with Geographic Information Systems. The work was prompted by the recognition that many of todays commercially available GIS products are firstly difficult to learn and use, secondly are difficult and time-consuming to customise, and thirdly the knowledge gained in using one product is not readily transferable or applicable to another. Consequently the aims of the research are to produce a prototype environment which is independent of the underlying GIS, provides high level analysis, design and customisation facilities, and presents the user with an adaptable, extensible, easy to learn and easy to use interface. In order to satisfy these aims, the research has the following specific objectives:

- to identify the common functional components that must be supported within a generic spatial language for GIS operations
- to define the form of a generic spatial language processor to support the functions while permitting input from a variety of sources such as voice, WIMP and command line interfaces
- to build a prototype generic user interface environment with interfaces to a number of commercially available GIS products
- to test user response to the interface.

This paper outlines the work performed to date and discusses in more detail the architecture of the user interface environment, the conceptual model proposed within the graphical user interface, the identification of a "standard" set of common GIS functions and the approach taken for interface customisation.

INTRODUCTION

Expansion of the Problem

The Use Environment The user environment is a vital element of any GIS. Long ignored as an esoteric aspect of GIS design while GIS development was driven by the need to extend functionality, the user environment is now beginning to attract its due attention. The development of the Universal Geographic Information eXecutive (UGIX) (Rhind, Raper and Green 1989) is a response to the widely expressed need to improve the usability of GIS, especially through the improvement and extension of the user interface. However, the implementation of a GIS user environment involves considerably more than the improvement of the human-computer interaction (HCI) process. Since GIS are conceptually complex and involve diverse operations ranging from data modelling to geometric transformations, improving the HCI cannot be a complete solution to the improvement of GIS use.

A number of general problems afflict many commercially available GIS which can be characterised as failings of the user environment. One of the most pervasive is the blurring of the distinction between goals, tasks and system functions in the language and process of interaction with the system. This means that the user cannot easily comprehend the structure of the user environment. In the design of UGIX the following definitions are used, and the concepts implemented in the interface:

GOAL	a user target for spatial data manipulation expressed in terms of application-specific outcomes e.g. finding which stands of trees in a forest will come to maturity in each of the next 5 years
TASK	a spatial data manipulation procedure expressed in terms of system implementable steps e.g. searching for certain spatially referenced items in a database and displaying the results in a map
FUNCTION	a low-level system operation to manipulate spatial data e.g. plotting a symbol at specified X,Y coordinates on an output device.

In this scheme tasks and functions refer to system operations, while goals apply to conceptual operations which are conceived of without reference to a computing environment.

Using this terminology, most GIS offer a command language composed of functions which are spatial tools and algorithms of various kinds. These commands are often modifiable with arguments and the complete expressions used are complex and often obscure. As part of a general movement to improve this situation a number of commercial systems have begun to offer graphical user interfaces (GUI's) built using window-icon-mouse-pointer (WIMP) techniques. However, these developments have illustrated the difficulties inherent in assigning icons or menu items to functions i.e. while the range of options available is now stated, the system structure is still no easier to understand. In particular, it is difficult to convert a goal into a task made up of the appropriate functions. Added to these implicit difficulties are the problems of overfilling the screen with icons or creating very long menus, the use of inappropriate screen metaphors and the lack of activity indicators to indicate the status of an operation to the user.

A further problem is that there are many different user languages for space in use, such as those defined by professionals working with spatial data (see examples in table 1). The table (with reference to two contrasting applications) shows how the relationship between objects in the application domain, user descriptions of space and the basic spatial data types supported by GIS products can be complex and difficult to understand.

The process by which the user links the concept and implementation can lead to confusion and to users making errors in they way they specify operations. A well designed user environment requires an interface which permits the customiser or expert user to link the appropriate user language for space to the system architecture in a way which is transparent to the end users. In other words the interface should allow the user to manipulate objects that are meaningful in terms of the application, like sub-divide a parcel rather than split a polygon.

Application Entity Types	Spatial Data Type	Comment
Land surveying		
Monument	Point	Fixed point located by physical mark
Centroid	Point	Centre point to which reference code is linked
Metes	Line	Boundaries of parcel defined by distance & direction
Bounds	Line	Boundaries defining position of adjoining parcels
Strip	Line	Corridor of fixed width either side of a centreline
Abuttal	Line	Boundary of a parcel on an adjoining parcel
Easement	Line/polygon	Corridor or area of land parcel set aside
Parcel	Polygon	Unit of land ownership
Aliquot	Polygon	Subdivision of parcel
Tract	Polygon	Land segregated by resurvey
Mine Surveying		
Face	Line	Section of mine boundary used for excavation
Entry	Line/polygon*	Centreline/section of tunnel forming access to face
Cross-cut	Line/polygon*	Centreline/section of tunnel at right angle to entry
Pillar	Polygon	Area of unexcavated material within mine
Room	Polygon*	Section of tunnel ending in face

* Polygon defined by closure of an open polygon in specified circumstances e.g across end of a tunnel

Table 1
Examples of user language for space for land and mine surveying

User environments of all forms have long lacked good visualisation tools for the spatial database data model. Ideally the user should be able to see the entity types and their interrelationships graphically expressed so that they can formulate queries more easily. Finally, the poor quality of help systems for many GIS has also become a major drawback for many use environments. Frequently the help is simply a formal statement of the command syntax and arguments, and not an explanation of its wider usage.

Customisability Commercial GIS software packages are normally designed to be fairly general purpose in nature - they are not designed for a specific well-defined application within a particular organisation. Consequently, they need to be adapted to fit the specific application and user requirements of the organisation within which they are implemented. This adaptation of the as-supplied system is termed *customisation*. The term *customisability* is used to describe the ease and extent to which a system may be customised.

The objectives of the customisation process are to provide a system for the user that supports both the data model and functionality demanded by the application requirements, that presents to that user an interface specific to the user's application, language and experience, which is uncluttered by non-required functions, icons and menus, is easy to learn and easy to use.

At present the base products delivered by GIS vendors are little more than a box of low-level spatial tools. These general purpose tools do not directly satisfy the user's functional requirements which are determined by organisational and application specific objectives. Furthermore, the tools will often have little meaning or applicability to the end user who must be educated in the language, interface and conceptual model supported by the product. The customisability of existing GIS products is poor, especially in the areas of database

design and implementation, task definition and user interface design and development. The result is that effective customisation of a GIS product to satisfy corporate GIS requirements involves enormous expense and effort. This problem is so acute that it is not unusual to see organisations struggling to use a system that is uncustomised and uncustomisable with the available resources. Effectively, the application requirements are largely discarded so that the functions the GIS supports become the application.

The customisation process incorporates all the normal stages of the familiar systems development life-cycle, including planning, analysis, design, construction, implementation and operation. The analysis stage incorporates both data analysis (resulting in the development of data models) and function analysis which involves both process and event analysis. The design phase incorporates logical and physical database design, task design and user interface design. Construction refers to the actual development of the physical database, tasks and user interfaces, while implementation is concerned with delivering the working system to the user environment.

Non-Transferability of Skills. Each GIS product on the market today incorporates its own distinctive environment, being substantially different from virtually all other available products. Each system tends to have its own unique command language, icon set, menu organisation and form layouts. The methods of interaction with the system vary considerably, even for such simple actions as selecting an object, obtaining help information or indicating confirmation of an action. Each vendor tends to use their own set of jargon, often in a manner which is inconsistent with other GIS vendors.

Even worse, the underlying system architectures show through and must be understood by the user before effective system usage is possible. In the absence of any other strong conceptual model for the system (as might be presented in a fully customised environment), the underlying architecture (files, layers, coverages, points, lines and polygons) forms the basis of the mental model developed by the user. The application problem (e.g. forest resource management) becomes mapped to the problem of manipulating the components of the GIS architecture (i.e., the coverages, polygons etc.). Consequently the skill set acquired by a user is specific to the jargon and architecture of a particular product. Since each GIS uses different jargon and different architectures, the user's knowledge of one system is not readily transferable to another.

Expansion of the Objectives

Identification of Common Functional Components The first phase of the research involved identification of a set of common (generic) functions that should be supported within the UGIX interface. These functions must be independent of any underlying GIS implementation strategy (e.g. object, relational or sheet based) and dependent rather on the goals of the user. These functions will be accessed via a set of icons, menus and forms within the GUI, and as a set of commands, operators and procedures within the command language. A "standard" set of icons and command names will be provided for the generic functions which may be modified (when required) within the customised environment. It should be noted that the generic functionality will not be implemented within UGIX, but rather, it will be accessible

through UGIX. This distinction is important, since it restates the concept of separating the application from the user interface.

By identifying the functions required to satisfy tasks, and tasks to satisfy generic goals, and providing access to these via icons and commands, the interface should become more consistent and less dependent on the underlying GIS data structures and architecture. It is believed that this will make the system easier to learn and use, and once learned will provide the user with knowledge that should be more readily transferable to other systems.

Definition of a Generic Spatial Language Processor A command language for interaction with the GIS database that supports the generic functions is being developed. It is intended that the spatial language will be embedded within both 3GL and 4GL languages which will provide the program logic and control structures. A spatially extended form of SQL (SQL-SX) has been designed to provide a standard, transportable language suitable for database definition, query, insertion, deletion and update. SQL-SX is to be supplemented by the set of generic GIS functions identified above. The overall architecture for the spatial query processor has now been sketched out. It includes layers for SQL-SX, an equivalent iconic query language, an inter-process communications interface and a customisation environment.

Development of a Prototype Generic Use Environment To test the feasibility of the concepts described here and the usability of such an interface, a prototype use environment must be developed. Further, it must be interfaced to a number of commercially available GIS products to prove that the user interface can be detached from the underlying GIS product architectures. The more difficult aspects of the development are likely to be:

- creating an efficient system to map between the functions supported within the generic spatial language and the matching functions supported by the underlying GIS,
- hiding the user interface supplied with the underlying GIS.

Testing User Response The resulting prototype must be evaluated to determine that it does indeed provide a superior user interface environment to the standard interfaces provided by each of the underlying GIS products. To test the acceptability of the UGIX environment, a number of trials are proposed. Methods for evaluation of user interfaces include:

- formal analytical methods where the interface is evaluated in isolation from users (Grudin 1989)
- empirical methods where users are requested to perform the same tasks using different interfaces, and the performances measured and compared
- ethnographic methods where actual users are observed and the context and users observations are elicited (Crellin, Horn and Preece 1990).

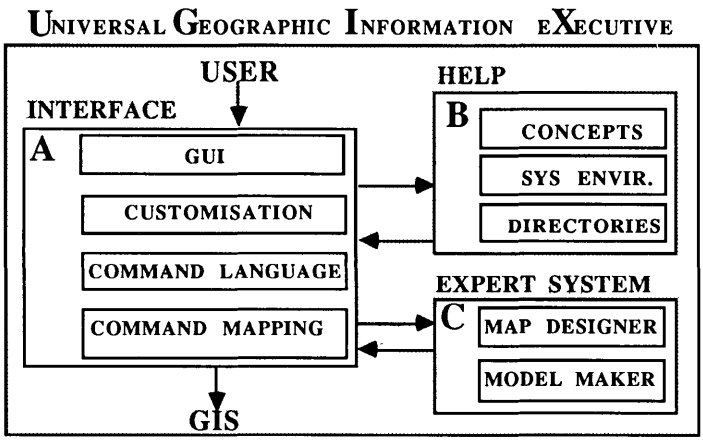
During design of the graphical user interface we propose to explicitly adopt accepted guidelines (e.g. proposed ISO guidelines Williams 1989, Strijland 1990) as an aid to user interface specification. Analytical methods, operating on data gathered by automated monitoring of user interaction, may be used subsequently to determine interface effectiveness, identify frequent command usages,

common errors and the relationships between use patterns and error occurrences. Chen (1990) describes the use of monitoring facilities built into the Xt toolkit to automatically gather appropriate information. Empirical studies are also proposed to compare the effectiveness of the new environment in direct comparison to the interface provided by the underlying GIS. Finally, user acceptability will be evaluated based on user interviews. It is intended that this evaluation will lead to suggestions for improvement for subsequent developments.

Background to UGIX

System Overview The UGIX system design as described by Rhind, Raper and Green (1989) contains 3 main modules, viz. (A) containing the screen interfaces, dialogues and command processor; (B) containing a help and information system for a GIS; and (C) an expert system shell or high level system access module. The structure of UGIX is illustrated in figure 1. This section describes the approach taken in the UGIX project, through first and second generation implementations. The major distinction between the generations is that the first aims to improve the usability of a specific GIS implementation, while the second aims to provide a generic user environment supporting transfer of skills between GIS and allowing easier customisation.

Figure 1. The three primary modules of the UGIX architecture.



The first generation approach to interface design within the UGIX project has been to prototype using Hypercard for the Apple Macintosh, where the Hypercard application (complete with in-built communications software) acts as a client to a host processor running the GIS application software (Raper, Linsey and Connolly 1990). This approach is similar to the one used by Cowen and Love (1988) to create an interface to the South Carolina Historic Preservation Office GIS database. Hypercard with its standard set of buttons, scrolling boxes and cards makes use of the GIS less daunting for the less technically-aware user. In addition, with the rich graphics environment available in Hypercard it is possible to show a graphic to illustrate the effect of various options available at any one point. It is also desirable to display all the commands available to the user in one place, with a pop-up explanation for each option.

Screen design has involved the standardisation of button and text field formats as well as card and background layout for different areas of activity such as:-

- Introduction and explanation (using a map guide);
- Map and file selection (using standard Macintosh file selection dialogue);
- Session screens for command processing;
- Help environment (UGIX (B) based on GISTutor version 2);
- A Gallery for maps and images generated in the GIS (along with button to redraw them).

Screen metaphors have been developed for each of these areas to make location in the system a graphical attribute. The interface also displays an activity index to give continuous feedback to the user on the status of the session. Currently this system interface 'shell' is being implemented for the GIS ARC/INFO, and is known as 'HyperArc': it is currently under test with users at 'novice' and 'competent' levels of expertise. In addition to feedback on the use of the system, the aim of the evaluation phase is to define a core area of functionality in common use to help optimise the UGIX system structure.

An important early objective in the development of HyperArc was the creation of file handling procedures to harmonise the user's concept of maps with that of ARC/INFO. This establishes that maps are both 'views' of spatial data and sheets within a series i.e., spatial tiles. Thus, search routines to find maps with particular names, to sort maps by type (e.g. point or polygon based), to access the map tiling system and to select the part of the sheet to view have all been created. In the first generation of the UGIX project the user specifies spatial queries using these system implementation concepts which are made comprehensible to the user diagrammatically (user testing is helping to refine this aspect). Hence HyperArc forces the user to work with ARC/INFO concepts, but tries to connect them with the user's view of the problem under study. This is ultimately restrictive to the user since the data structure is fixed, and maps are files which the user needs to manipulate in some way.

A basic principle of the UGIX design is that in order to make a GIS easy to use the process of making a database selection, displaying a map or carrying out spatial analysis must be broken down into a series of logical parts, linked by a pathway for the user to follow. Following such a path and gaining experience with the alternative options is an excellent way to improve a user's end-to-end understanding of the components of spatial data processing. Appropriate information needed for a user to make a decision is also retrieved before indicating the command options, for example only maps with the correct specifications are presented (e.g. with topological relations already created), when this is necessary for the operation.

Another UGIX design principle is that improving access to existing GIS can be achieved by converting the current function-orientation of the native system interface (primitive and implementation-specific operations) to a task-oriented interface (sequences of high level spatial operations) usable by a spatially aware user. The second generation of the UGIX project aims to build on the experience of constructing such task-oriented interfaces to create generic interfaces capable of communication with any GIS.

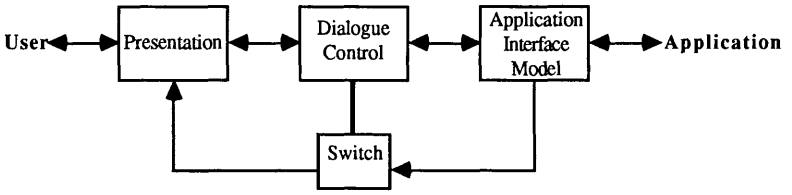
However, in order to implement such an interface in a generic way requires a new form of software architecture which is independent of specific implementations, does not enforce a particular data model, and adheres to the standards in the user community which are most crucial to the success of GIS in heterogeneous computing environments. To achieve this a layered model is suggested that protects the user interface from the actual implementation mechanisms provided by each GIS vendor. Each layer within the model will perform a particular task and have a well defined interface to the layers both above and below. Some of the layers within UGIX will be able to communicate directly with the underlying GIS at a matching level.

UGIX(A)

Design Overview

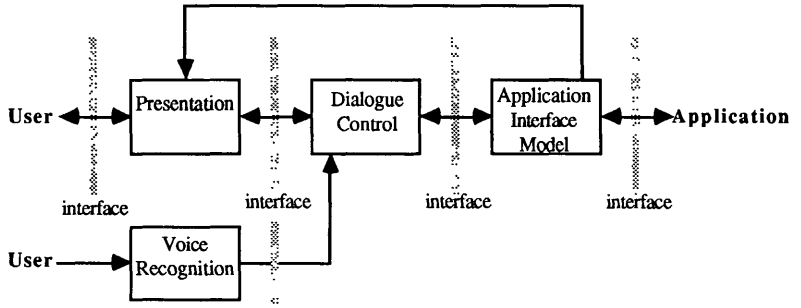
Separation of the user interface from the application is not a new concept. Early work resulted in what is often known as the "Seeheim model" (Green 1985) developed during a 1983 workshop on architectures for interactive systems at Seeheim. Subsequently, the identification of components of the overall system corresponding to semantic, syntactic and lexical aspects, and the relationships between them has lead to various alternative architectures. The development of general purpose software for managing the user interface as a separate process has lead to the comcept of the user interface management system (UIMS) (Pfaff 1985). Here the application and the user interface software are quite separate and communicate via a well-defined protocol.

Figure 2. The Seeheim model



The overall architecture for UGIX(A) is similar to the Seeheim model in many aspects. The requirement for a high bandwidth communications channel from the GIS application is supported to allow efficient graphics display and manipulation. Figure 3 illustrates the overall structure proposed for the UGIX environment.

Figure 3. Overview of UGIX(A) architecture



Description of Components

The presentation layer incorporates a standard user interface toolkit (e.g. Motif, OpenLook etc.) a widget design facility, a screen design facility and a screen execution facility. The widget and screen design facilities operate within the constraints of the toolkit, and will be implemented as a set of executable screens. The customisation environment itself and the actual resulting end-user application, will also simply be a set of screens with which the user may interact.

The screens will be designed in terms of a set of windows, a set of widgets within each window and a set of forms. The behaviour of the windows, widgets and forms will be described in terms of the 4GL command language. Interaction with the screens will cause the widgets to react in the predefined manner and the execution of GIS tasks in terms of the spatial command language embedded within the 4GL.

Equivalent commands may be issued directly via a command line interface, via widget interaction or using voice input. Each method should result in the execution of the same generic functions within the dialogue control component. The voice recognition facility issues either individual spatial command language tokens which may be used to build a complete command, or entire commands. Entire commands may be abbreviated into a spoken shorthand consisting of just a few words rather than requiring the user to speak the full command syntax for a particular task.

The application interface module accepts generic spatial language commands and maps them onto the command language of the underlying GIS. It then issues these commands to the GIS via an inter-process communications mechanism. The GIS responses may include alphanumeric information (which may be used to fill a form), status information (error and function status) and/or graphics. To support a highly interactive graphics environment requires that a high speed channel be provided to display the graphical data. However, alphanumeric and status data may be routed through the dialogue control module for further processing and display.

Figure 4 illustrates the proposed architecture of UGIX(A) in more detail.

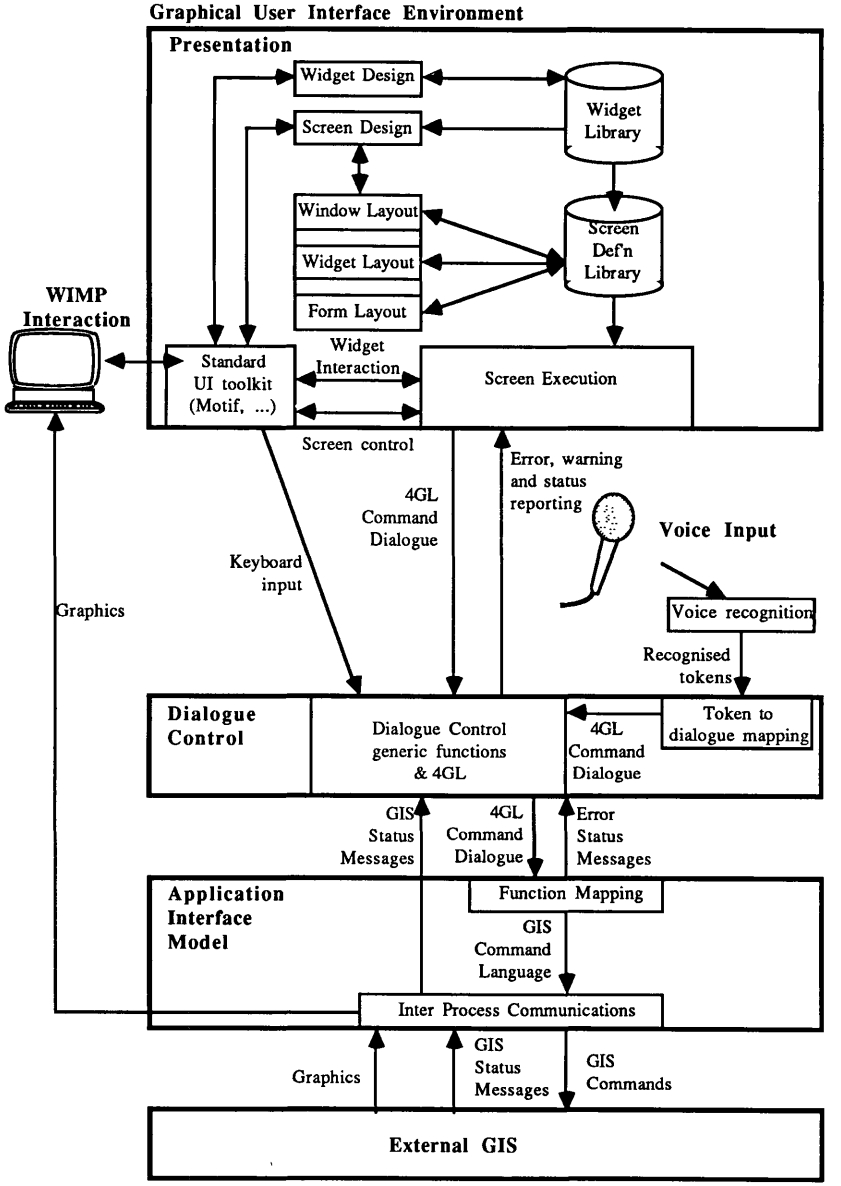
The Graphical User Interface

Wilson (1990) reviews the use of graphical user interfaces within GIS and the applicability of the desktop metaphor. He suggests guidelines for building suitable user interfaces. The GUI is described as having three components:

- an underlying conceptual model,
- a command structure comprising codes, function keys, buttons etc. with which to create a syntax, and
- the visible screen graphics, such as command lines, menus and icons.

To date, within the GIS world, most emphasis has been placed on the development of a command syntax and the design of menus, icons and screens. However, as yet there are no agreed standards for interaction with the interface unlike the PC world where techniques such as double clicking on an object to activate it, F1 function key for help etc. are commonly adopted.

Figure 4. More detailed breakdown of UGIX architecture



This lack of standardisation has lead to a lack of transferability of knowledge, long learning periods and generally difficult system usage.

Initial attempts at improving usability concentrated on reducing the number of interactions, menus, forms and icons that the user had to deal with. This approach either reduced the available

functionality or produced menu hierarchies that were difficult to use. An alternative approach is to use existing knowledge of a related field that may be applied to the new problem domain.

The Underlying Conceptual Model A major contributing factor towards the non-standardisation of the GIS user interface is the lack of an underlying conceptual model for the interface. It has been suggested (Gould & McGranaghan 1990) that the primary mechanism by which a user learns to use GIS is by metaphoric learning. Here the user is able to treat the unfamiliar environment like another familiar one thus reducing the overall learning period. The general cognitive process may be partitioned into metaphoric, analogical and modelling processes. The differences between the three processes and their implications for computer systems design are reviewed by Wozny (1989). The concepts of metaphor and analogy are closely related: analogy implies that one domain behaves like another, whereas with metaphor, the target domain is more directly mapped onto the other and hence becomes the other. Consequently, the use of metaphor within the user interface is preferable since it allows a user to interact with an unfamiliar system as if it ***is*** an environment with which they are familiar. This effectively reduces the learning time, reduces stress caused by unfamiliarity (i.e. makes for a happy user) and provides a conceptual framework for the new environment which may be built upon. For infrequent users, the use of metaphor may be more important, since they may never progress beyond the metaphor presented to develop a mental model of there own (Wozny 1989).

Existing graphical user interfaces for non-GIS applications have often been developed using the desktop metaphor as the underlying conceptual model. The desktop metaphor is suitable for many business related applications since the activities performed by the computer based application have direct equivalents with the manual methods. However, it may not be readily applicable to many GIS applications due to the lack of spatial and mapping related activities that normally occur on and around a desk.

The wide variation in GIS applications and the variation in experience of GIS users indicates that a single conceptual model is unlikely to satisfy or be applicable to all situations. If we perceive GIS to be an enabling technology for the integration of spatial and aspatial data, we must then consider it to be equivalent to a DBMS in generality, and hence not suited to a single model. In contrast, a GIS customised to suit a particular narrow application (e.g. mains fault analysis in the Water industry) may provide a situation where an applicable underlying conceptual model may be utilised.

Wilson (1990) pointed out that some GIS applications may have no equivalent manual method. However, this does not imply that a conceptual model on which to base the user interface cannot be found. Rather it implies that analogy or metaphor may be suitable techniques for development of the conceptual model.

Current GIS technology imposes on the user a conceptual model of geographic space that is a function of the internal structures supported by the GIS (e.g. layers, points, lines, polygons). What we should be aiming for is a user interface that permits the system customiser to present a conceptual model to the user that is relevant and applicable to the both the user's background and the application in hand.

The strength of the desktop metaphor as used within the Macintosh and other PC environments for the underlying conceptual model, is that it provides an organising framework within which other operations and metaphors may exist. Gould and McGranaghan (1990) have extended this idea to suggest the need for an organising metaphor within which there may be other nested metaphors (which may themselves be organising metaphors). This approach has promise since it provides a structure within which applicable and relevant metaphors may be applied, rather than trying to apply a single metaphor to all situations.

The Organising Metaphor within UGIX The current thinking for the UGIX GUI is to develop an environment supporting nested metaphors. The proposed overall organising metaphor is a building, within which there are a set of rooms, each accessible via a door. It should be noted that the idea of using the room/building metaphor has been independently conceived by a number of different groups including researchers at Xerox Palo Alto Research Center and University of Waterloo (Chan and Malcolm, 1984), and even built into a number of existing products (e.g. Rooms from ENVOS and even X11 rev 4 attempts to provide a Rooms-like system).

Within UGIX, each room may possess its own organising metaphor. Most rooms will be directly accessible from the entrance hall although some special-purpose rooms may require access from within another room. On entering the system the user is located in the entrance hall, a neutral, public space through which the user moves to particular environment. Doors provide access to the environments the user has access to. The door metaphor is a strong metaphor for access into and out of different environments (Cátedra 1990) and provides features such as locks, opening and closing. These features may be used directly for access control, entering and leaving. Within each room, a single type of activity occurs, and a single lower-level organising metaphor is employed.

For example, one room provides an environment where the desktop metaphor is supported. Here general filing, correspondence and interfacing to external non-GIS packages (e.g. word processing, spreadsheets) takes place. Access to the aspatial part of the GIS database is available via card indexes, file folders etc. and alphanumeric reports can be created and printed.

A second room contains the drafting office where the map, drafting table, map cabinet and light table are the principle metaphors. Now access to the GIS database is via the map. Maps may be taken out of the filing cabinet, updated, viewed and copies taken for development proposals etc. Eventually these may be replaced in the map cabinet following approval by the chief draftsman/engineer. Note that operations not consistent with the map metaphor may not be applied here.

A third room contains a library. Within the library books are kept providing reference material, reports, archives and documentation. Updating of system documentation is performed here.

A fourth room contains the development and customisation environment. A workstation metaphor is supported which provides direct access to the 4GL development and customisation environment.

There is one special door that leads off the entrance hall - an external door. Through the door, blue sky and clouds may be glimpsed,

and on entering this environment the real world metaphor is used for access to the GIS database. Here there are no seams, maps or files - only a continuous world containing objects. This is where the experienced GIS user works and it is also the environment in which virtual reality may one-day be accessible (Jacobson 1990).

Users are provided with keys that are able to unlock only some doors. It is feasible to consider more specialised rooms leading off of others. For example the database administrator and system manager may be in their very own room accessible from the development and customisation area.

The concepts of buildings, rooms and doors are internationally and culturally neutral, providing an almost universally understood concept. A further advantage exhibited by this metaphor is that it provides an easy facility for extension. To add new environments involves simply adding another room (with door!) to the building. Within each new environment a different organising metaphor may be used to support functions not supported elsewhere. The possibilities of this metaphor are seemingly endless - e.g. leaving the system may simply be performed by turning off the light switch in the entrance hall, or alternatively going through the door that leads out into the night.

The Iconic Query Language Access to the functionality provided within each environment (room) will be predominantly via icons, menus and forms. The icons should fit the organising metaphor for that environment so that they have relevance and preferably direct applicability. Consequently, the drafting office might be designed specifically for experienced cartographers and hence might support map cabinets from which maps may be extracted, drafting boards on which map updates and viewing may be performed and light tables on which map overlay operations may be carried out.

Consistency and simplicity are key considerations when attempting to design a user interface, be it for a GIS or for a dishwasher. A concise and simple syntax for manipulating the icons and database objects is required which is both consistent and meaningful in terms of the metaphors used. Existing GUIs such as that used on the Macintosh are not fully consistent. Consistency between applications has been encouraged since Apple provide a set of guidelines for developers to follow (ref Apple Mac developers guide). Consequently most packages available for the Mac have a similar look and feel so that knowledge of one application/package provides useful knowledge on the use of others.

Certain other aspects of the Mac interface are far less consistent. In particular the order in which the objects and the operators are selected varies from one type of operation to another. Objects to be manipulated are usually selected first, and then the operation to be applied is selected (e.g. discarding data by moving it to the wastebasket, applying a different font and ruler to a section of text). However, sometimes the operation to be performed is selected first and then the objects to which it is to be applied are identified (e.g. select the print function and then indicate which pages to print). Hence, even though operations that are common between applications are normally presented in a very similar manner the syntax for different operations may vary within an application.

A further level of inconsistency, most frequently observed by novices, is the use of the wastebasket. Why is the trash can used to eject the disk when it is normally used for deleting data? Most novices are unable to find a method for ejecting the disk, since the use of the wastebasket for deleting documents and folders implies that if the diskette is moved to the wastebasket, all folders and documents on the disk will be deleted. This latter example indicates where the use of a metaphor has been extended beyond its applicability and used in an inconsistent manner.

The impact of the English language on the syntactic ordering of operations, parameters and objects (verbs, modifiers and nouns) may not have relevance to the iconic interface. Although it is known that language structures our concept of space (Talmy 1990), it is not thought that language will adversely impact the syntactic structure of the iconic interface. In English we generally use a noun-verb-modifier ordering to state facts (e.g. Jack closed the door), but a verb-noun-modifier ordering for instructions or commands (e.g. close the door quietly please). Most of the operations performed within the GIS tend to be instructional in that the user is commanding the system to perform some action (e.g. modifying, deleting, reporting), supporting the adoption of a verb-noun-modifier ordering.

However, most iconic interfaces require that the objects are selected prior to identification of the action (i.e. noun-verb-modifier, or object-action ordering). Even though this ordering is not common within the English language for instructional sentences, it does feel natural for English speaking users of the iconic interface.

Perhaps the most important aspect of this ordering is that object selection and the operations to be performed on those objects are effectively separated. They have become two discrete instructions issued by the user. Furthermore, object selection is common to virtually all operations and becomes independent of those operations, meaning that a single set of object selection techniques can be applied throughout.

One significant disadvantage to the use of object-action syntax ordering is that the selection process may select objects for which the operation may be invalid. If the operation to be performed is identified first, the object selection process can use knowledge of the operation to ensure that only appropriate (valid) objects are selected. Within existing GUIs, this problem is partially overcome by disabling functions for which the selected data is inappropriate. However, if it is not obvious which of the selected objects is causing the function to become unselectable, much operator frustration is likely to ensue.

Within the UGIX GUI, we recommend the use of object-action ordering as the basic syntactical construct for icon interaction. Object selection will be performed first. Subsequent identification of an action will apply that action to the selected set of objects.

The Command Level Interface

The command level interface incorporates a 4GL command language, a function mapping facility and an inter-process communications facility. It accepts commands from the GUI and the voice recognition system in terms of 4GL command sequences. It can also be accessed directly to perform ad hoc functions and applications development.

Smallworld Magik, an object-oriented (OO) development facility from Smallworld Systems has been selected for the development of the prototype.

Smallworld Magik: An object-oriented development environment. The objective is to support a single command line and development environment in which application development, database definition, ad hoc queries, menu, form and icon commands, database access and graphics are all available. It is also desirable that the full power of the underlying GIS, DBMS and UIMS are available. An object-oriented development language has been selected since it offers the opportunity for high programmer productivity and a structured development approach. The use of OO techniques such as code re-usability, inheritance and encapsulation can reduce the overall development effort for a complex system. The Magik language (Chance, Newell and Theriault 1990) is a hybrid of the procedural and OO approaches and supports its own interactive development environment. It is fairly readable (certainly more so than 3GLs such as C and C++), comes with a comprehensive set of standard object classes, methods and procedures, and provides the ability to transfer applications between hardware and operating system platforms with a minimum of effort. It utilises the X-Windows standard for all interaction with the workstation.

For the UGIX development, we are extending Magik by adding a new set of language constructs to support a spatially extended version of SQL.

SQL-SX: A spatially Extended version of SQL The relational language SQL (Date 1989). forms a suitable base on which to develop spatial extensions due to:

- its widespread acceptance by database users
- its availability within a large number of commercially available DBMS (relational and non-relational)
- its acceptance as an international standard
- its ongoing development, thus ensuring a long-term future.

The use of relational database management system (RDBMS) technology within the existing GIS user community is virtually universal. Further, it is likely that RDBMS will be the major data management technology for at least the 1990s, and that SQL will be the major language for interaction with those databases. The investment by user organisations in training staff in the use of SQL is significant so there is consequently a sizeable body of expertise available both within GIS user organisations and in the general computing industry. The use of a non-standard query language for GIS implementations does not appear commercially viable, nor practical in the near future.

A number of GIS vendors are already developing spatially extended versions of SQL and have reported their work in the research literature including Kork (Ingram & Phillips, 1987), Intergraph (Herring, Larsen & Shivakumar, 1988), Wild Heerbrugg (now Prime) (Charlwood, Moon & Tulip, 1987) and GeoVision (Westwood, 1989). Each has attempted to provide facilities supporting spatial predicates and spatial data manipulation facilities within SQL (or SQL-like query languages). Unfortunately, the basic query language in each case has been an incomplete implementation of the ANSI/ISO SQL standard (ISO 1987, ISO 1989), and the spatial extensions were fairly minimal and

elementary. To make matters worse, the extensions in general do not maintain consistent syntactic and semantic constructs with the rest of SQL. For example, spatial predicates are not in general supported within the WHERE clause, but rather within a separate clause.

Other researchers including Pong-Chai Goh (1989), Sacks-Davis, McDonell and Ooi (1987) and Egenhofer (1987) have also provided useful contributions towards the development of spatial extensions to SQL. However, until recently there has been no proposal for a standard set of extensions put forward for discussion. In our recent paper (Raper and Bundock 1990) we proposed a set of spatial extensions for SQL that could form the basis for an agreed standard between GIS vendors and the GIS user community. The spatial extensions are based on the existing proposals for SQL2 and SQL3 being studied by the combined ISO-ANSI SQL standards working group (ISO-ANSI 1989). These proposals include a number of object-oriented concepts, including support for abstract data types, methods, operators and functions. In particular, the detailed proposal in support of abstract data types (Kerridge 1989) if implemented, would provide the framework in which to develop spatial data types, spatial operators and spatial functions, while remaining completely within the SQL standard.

The extensions necessary to make SQL usable within GIS applications for query of both spatial and aspatial data include:

- spatial data type(s) e.g. point, node, line, polygon
- spatial operators (predicates) e.g. at, inside, encloses, connected_to
- spatial functions e.g. area_of, length_of
- long transaction management statements
- report specification facilities - both textual and graphical

In addition, functional requirements demand that:

- the spatial data types should be displayed graphically as a result of being SELECTed
- the data dictionary and DDL support the extensions
- spatial access control (protection) may be provided by inclusion of spatial data types, predicates and functions within VIEW definitions
- spatial integrity maintenance may be provided by support of spatial data types, predicates and functions within the CONSTRAINT clause.

The Function Mapping Facility. This facility accepts UGIX commands in the form of function specifications and data selection specifications, and transforms these to the command language of the underlying GIS. For functions supported by both UGIX and the underlying GIS, the mapping should normally be moderately straightforward. Using OO techniques, the function mappings become methods for the function objects. It is hoped that this technique will provide a straightforward approach to allow the support of multiple underlying GIS products.

Functions in UGIX not supported by the GIS However, not all functions known to UGIX may be available in the underlying GIS. Consequently, there will be holes in the UGIX interface for those unsupported functions. Icons representing unsupported functionality will be displayed in grey, and the matching command level functions will possess a method that reports on the function unavailablity.

Functions in the GIS not supported by UGIX Where the underlying GIS supports functions unknown to UGIX, a general purpose facility will be provided that allows UGIX to send the system dependent command string directly to the underlying GIS. The command string will be explicitly declared and may be associated with icons, menus and forms in the normal way. In this way, special purpose functionality may be readily included in the UGIX GUI, even if it is not considered generic enough to warrant inclusion in the generic function list.

SYSTEM CUSTOMISABILITY

The Requirement for Customisation

Customisation of any GIS is required to allow the system to manage and manipulate the entity types that exist in the problem domain. The system customiser or database administrator, must be able to describe to the system the object classes/entity types that are to be modelled within the resulting system. They must define the names of the object classes, the names and types of attributes the objects possess, the behaviour of the objects with respect to operations on the objects and the inter-relationships between objects. The object class names, attribute names and operation names should each be assigned in terms of the language normally used within the application (i.e. the application specific jargon - e.g. the land surveying examples of table 1) not in terms of the language used by the GIS. Defining the objects, attributes and operations in terms of the user's language allows interaction with the system to use that language.

The user interface may be customised to reflect the names described above and the symbols used by the application. Consequently, the standard icons representing the object classes, attribute types and operations of the application domain must be generated and associated with the names of the matching elements. This permits interaction with the GUI to be performed with icons recognised by the user as being part of the application domain.

The graphical user interface may also be customised to provide a conceptual model using metaphors which inexperienced users may recognise from previous experience - either application experience or experience from other domains. This conceptual model must be appropriate to both the application and the user's background.

The customisation facilities must be continually available, rather than being used just once to create a fixed, static system. Business (research, education, ...) requirements change, resulting in either changes to existing applications or entire new applications being created. The user interface must also remain adaptable to match individual user preferences, and user-specific tasks.

Integration of Task Analysis Methodologies

A variety of methodologies and associated tools are available today to assist system designers and customisers determine the requirements for a new information system. User-centered requirements analysis methodologies provide a structured approach to performing both data and task analyses.

Data analysis results in a detailed description of the object classes, their attributes and the inter-relationships between object classes that might be managed within the final system. It also identifies integrity constraints and any other special behaviour

exhibited by objects when a change of state occurs. Although this information may be recorded on paper in a descriptive manner, it is also possible to save it in a database - often refered to as the data dictionary. Tools that assist the data analysis task will nearly always save this meta-data in a form that may be used at a later date for automating the creation of the target database.

Task analysis results in the development of a detailed description of the goals, processes and user interaction that must occur for the goals to be met. Tools that assist this process may also store the task descriptions in a database in a structured form. Formal mechanisms have been developed to structure this information in such a manner that it may be used later for automating the creation of the user interface. For example Extended Task Action Grammar (ETAG) (Tauber 1990) may provide such a mechanism, although it is likely that the level of detail required to define the target system may be significant.

It is intended that UGIX(A) incorporate tools to assist both data and task analysis. These tools will be used to gather and structure information describing the target environment in such a manner that it may subsequently be used to automatically generate the database definition and the user interface. Further, the information should be in a form to provide input to the help facility supported by UGIX(B) since descriptions of the data, the low level functions and the tasks will all be available in a structured form.

COMMON FUNCTIONAL GIS OPERATIONS

A Methodology for Identification of Key GIS Functions

Analysis of the functions to be supported directly within the interface is based on goal analysis and hierarchical decomposition (top-down) techniques rather than task analysis techniques since the interface, as delivered, must retain its generality. Task analysis might provide a more detailed definition of requirements for a particular (well defined) application but is considered too specific when attempting to define general requirements. Abstraction of the general functionality from a task based analysis might be possible given sufficient access to a wide range of actual users in a wide range of applications. However, operating within a tight set of time and resource constraints, requires the analysis to be performed quickly with minimal contact with real users.

To overcome this limitation, the study initially involved :

- a review of existing research literature,
- a review of a number of GIS tenders,
- a review of the functionality supported by a number of commercially available GIS products.

A number of authors have reported attempts at classifying GIS functionality, including Dangermond (1983) and Maguire and Raper (1990). However, Dangermond based his analysis firmly in terms of the map concept which partitions the world into discrete tiles. Consequently, much of the functionality described is concerned with managing these discrete units, to allow queries across multiple map sheets, to join multiple maps to form a new composite map, and to perform edge matching. This functionality is quite distinct from the goals of the user, being the functionality necessary to support a map-

sheet based GIS implementation. Maguire and Raper describe the functional components of a GIS in a more generic manner. Below each identified high level function they separate the functionality that applies to the spatial data, from that which applies to the attribute (aspatial) data. In the current study we attempt to retain the user's concept of objects within their application domain which may possess spatial and/or aspatial components. This suggested set of generic functions will be presented with the first UGIX prototype.

CONCLUSIONS

The development of a GIS independent user interface environment capable of interfacing with a number of commercially available products while still providing an adaptable, consistent, easily learned and easy to use interface, appears at first sight a difficult task. A structured approach to this problem is however beginning to indicate the feasibility of the project.

Incorporating structured analysis tools into the environment is expected to simplify system customisation while improving the resultant interface. In particular, it should help provide a user interface that incorporates the terminology, icons and conceptual models specific to the application and user's background while developing the interface in terms of a set of standard guidelines.

The usability of the prototype system will be evaluated and compared to the interfaces offered by the underlying GIS products. The comparison will eventually prove or disprove the viability of this approach. If successful, we would like to promote the concepts incorporated within UGIX to initiate discussion between users and vendors on the development of standards and guidelines for GIS user interface design and construction.

ACKNOWLEDGEMENTS.

The authors of this paper acknowledge the help of ESRI in the funding of work leading to the prototyping of HyperArc, the first generation of UGIX. Apple Computer (UK) have also assisted in this work by establishing a laboratory of Macintosh computers which form the Apple Mapping Centre at Birkbeck College. Receipt of an ESRC/NERC grant from the collaborative programme in GIS is acknowledged by MSB. Smallworld Systems have assisted the project with hardware and software tools. We would also like to thank Lesley Bundock for her efforts in reviewing and commenting on various drafts of this paper.

REFERENCES

Apple Computer (1987). *Human interface guidelines.* Addison Wesley, Amsterdam

Cátedra M. (1990). "Through the Door": A view of space from an anthropological perspective. *Proceedings of NATO ASI Cognitive and Linguistic Aspects of Space,* Las Navas, Spain

Chan P.P. and Malcolm M.A. (1984). Learning considerations in the Waterloo port user interface. *Proceedings IEEE First International Conference on Office Automation,* IEEE, New York

Chance A., Newell R.G. and Theriault D.G. (1990). An Overview of Magik, Technical Paper 9/1, Smallworld Systems Ltd, Cambridge, UK.

Charlwood, G. Moon, G. and Tulip, J. (1987). Developing a DBMS for Geographic Information: A Review, *Proceedings Auto-Carto 8*, Baltimore, Maryland.

Chen J. (1990). Providing Intrinsic Support for User Interface Monitoring, *Proceedings Human-Computer Interaction - INTERACT '90*, Cambridge, UK.

Cowen D.J. and Love S.R. (1988). A Hypercard based workstation for a distributed GIS network. *Proceedings GIS/LIS '88*, San Antonio, TX, USA.

Crellin J., Horn T. and Preece J. (1990). Evaluating evaluation: A case study of the use of novel and conventional evaluation techniques in a small company. *Proceedings Human-Computer Interaction - INTERACT '90*, Cambridge, UK.

Dangermond J. (1983). A Classification of Software Components Commonly used in Geographic Information Systems. *Proceedings United States/Australia Workshop on Design and Implementation of Computer-Based Geographic Information Systems*, Amherst, New York

Date C.J. (1989). *A Guide to the SQL Standard*, Second Edition, Addison-Wesley, Reading, Massachusetts

Egenhofer M.J. (1987). An extended SQL syntax to treat spatial objects, *Proceedings 2nd International Seminar on Trends and Concerns of Spatial Sciences*, New Brunswick.

Green M. (1985). Report on Dialogue Specification Tools. In Pfaff, G.E (ed) *User Interface Management Systems*. Springer Verlag. pp 9-20.

Gould M.D. and McGranaghan M. (1990) Metaphor in Geographic Information Systems, *Proceedings 4th International Symposium on Spatial Data Handling*, Zurich, Switzerland.

Grudin J. (1989). The Case Against User Interface Consistency, *Communications of the ACM*, 32, 10.

Herring J.R. Larsen R.C. and Shivakumar J. (1988). Extensions to the SQL Query Language to Support Spatial Analysis in a Topological Data Base, *Proceedings GIS/LIS '88*.

Ingram K.J. and Phillips W.W. (1987). Geographic Information Processing Using a SQL-Based Query Language, *Proceedings Auto-Carto 8*, Baltimore Maryland.

ISO (1987). Report ISO 9075 Information Processing Systems-Database Language SQL

ISO (1989). Report ISO 9075 Information Processing Systems-Database Language SQL

ISO-ANSI (1989). Database Language SQL2 and SQL3 (ISO-ANSI working draft) X3H2-89-252 and ISO DBL FIR-3, J Melton (editor), July 1989

Jacobson R. (1990). Virtual Worlds, Inside and Out. *Proceedings NATO ASI Cognitive and Linguistic Aspects of Space*, Las Navas, Spain

Kerridge J.M. (1989). A Proposal to add User Defined Datatypes to SQL, ISO report ISO/TC97/SC21/WG3-DBL-FIR-3.

Maguire D.J. and Raper J.F. (1990). Design Models and Functionality in GIS, *Proceedings GIS Design Models*, Leicester, UK.

Pfaff G.E. (ed) (1985). *User Interface Management Systems*, Eurographics Seminars, Springer-Verlag, Berlin

Pong-Chai Goh (1989). A GQL for Cartographic and Land Information Systems, *International Journal of Geographic Information Systems*, vol. 3, no. 3, 245-255.

Raper J.F. and Bundock M.S. (1990) UGIX: A Layer Based Model for a GIS User Interface. *Proceedings of NATO ASI Cognitive and Linguistic Aspects of Space*, Las Navas, Spain

Raper J.F. and Green N.P.A. (1989). Development of a Hypertext Based Tutor for Geographical Information Systems. *British Journal of Educational Technology.*

Raper J.F., Linsey T. and Connolly T. (1990). UGIX: A Spatial Language Interface for GIS : Concept and Reality. *Proceedings of EGIS '90*, Amsterdam, The Netherlands

Rhind D.W., Raper J.F. and Green N.P.A, (1989). First UNIX then UGIX, *Proceedings of AutoCarto 9*, Baltimore, MD, USA.

Sacks-Davis R., McDonell K.J. and Ooi B.C. (1987). GEOQL - A Query Language for Geographic Information Systems, Internal Report 87/2, Dept of Computer Science, Royal Melbourne Institute of Technology.

Strijland P. (1990). *Icons for use on screens Part 1: Specifications.* ISO/IEC JTC 1/SC18/WG9 Working Document (4 July 1990).

Talmy L. (1990). How Language Structures Space. *Proceedings of NATO ASI Cognitive and Linguistic Aspects of Space*, Las Navas, Spain

Tauber M.J. (1990) ETAG: Extended task action grammar. A language for the description of the user's task language. *Proceedings Human-Computer Interaction - INTERACT '90*, Cambridge, UK.

Westwood K. (1989). Toward the Successful Integration of Relational and Quadtree Structures in a Geographic Information System. *Proceedings National Conference - GIS - Challenge for the 1990's.*, Ottawa, Canada

Williams J.R. (1989). *Menu Design Guidelines.* ISO TC159/SC4/WG5 WD 9241-14 (17 March 1989)

Wilson P.M. (1990) Get your Desktop Metaphor off my Drafting Table: User Interface Design for Spatial Data Handling. *Proceedings 4th International Symposium on Spatial Data Handling*, Zurich, Switzerland.

Wozny L.A. (1989) The Application of Metaphor, Analogy and Conceptual Models in Computer Systems. *Interacting with Computers: The Interdisciplinary Journal of Human-Computer Interaction.* vol.1 no. 3 pp 273-283