User Interface Issues in Geographic Information Systems

Juliano Lopes de Oliveira
Claudia Bauzer Medeiros

Relatório Técnico IC–96-06

Julho de 1996

User Interface Issues in Geographic Information Systems

Juliano Lopes de Oliveira Claudia Bauzer Medeiros

July 1, 1996

Abstract

Recently, much research effort has been employed in the area of Geographic Information Systems due to the vast potential for applications of this technology. Simultaneously, user interface subsystems of software products have received attention since the interface has marked influence in software acceptance. This paper presents an overview of research done in the intersection of these areas. The main approaches and the current problems of user interfaces for Geographical Information Systems are discussed and analyzed. This study concludes with open problems and new research directions for future work in this area.

1 Introduction

Geographic Information Systems (GIS) group tools and methods for the management and retrieval of georeferenced data. These data refer to geographic phenomena and includes their physical location and spatial relationships [Car89]. Applications of GIS technology vary from worldwide scale (e.g., global natural resource management) to local concerns (e.g., city planning). Examples of such applications appear in areas such as infrastructure management, surveying and mapping, transportation and logistics, election administration and redistricting, natural resources management, and many others [MGR91, ABC+91].

The economical and technical perspectives of GIS have also been noticed by the research community, and although the first developments have begun in the early 1960s, only recent advances in hardware and software technology have made possible the implementation of systems with adequate functionality. The main characteristics expected on the new generation of GIS are:

- Data is managed by geographic database systems: the systems should provide all the advantages of conventional Database Management Systems (DBMS) extended with the capability of manipulating georeferenced data. There is a trend for implementation of such geographic database systems using object-oriented DBMS.
- The architecture of GIS is open: therefore, the set of functions and spatial analysis tools provided by GIS are extensible, allowing smooth integration with other systems.
- There is intensive use of graphics capabilities: these capabilities are used not only for static output (such as paper maps), but also for dynamic and interactive displays.

- Georeferenced data originates from many different sources (e.g., satellite images, digitalized maps, and census tables): as a consequence, multiple representations of the same entity of the real world may coexist in the geographic database.

The *user interface system* (or *interface*, for short), has gained much attention since software developers realized that usability is a key factor for the success of a product. In fact, for the majority of the users, the interface *is* the system, and therefore the choice of a software product is based on the ease of use of its user interface subsystem.

The economical and technical importance of the user interface is now well established in the computer science research community, and many conferences and projects all over the world deal specifically with human factors and user interface issues. Open problems still exist in this area, and they are increased when merged with the properties and idiosyncrasies of GIS.

This paper discusses the barriers for the development of adequate user interface systems for GIS. The next section shows the particularities of GIS that increase the complexity of its user interface. Section 3 describes architectural alternatives for implementation of GIS user interfaces. In section 4 the existing languages for the interaction with GIS are analyzed. Section 5 addresses human factors and cognitive aspects of GIS user interfaces. Finally, section 6 presents the conclusions of this work.

2 Distinctive Features of GIS User Interfaces

The fundamental question to be answered in this section is: why GIS interfaces are more complex than other DBMS interfaces. In fact, if we assume that GIS have a spatial DBMS component, the user interfaces of GIS may be considered in the light of conventional DBMS user interfaces. In this sense, all the desired features of DBMS user interfaces should be present in GIS user interfaces. [Oli94] presents a non-exhaustive list of such features that includes:

- access to the whole set of DBMS features;
- automatic display generation;
- support to different abstraction levels for data visualization;
- data access via DBMS only;
- support to database integration; and
- full representation of the DBMS data model.

A brief discussion of these features can give a flavor of the additional complexity of GIS user interfaces. First, GIS are open systems with extensible functionality, and so should be the user interface in order to provide access to all the GIS functions.

Automatic display generation is very simple in conventional DBMS. In object-oriented DBMSthe problem is complex, but suitable solutions have been presented [MM91, OA93a].

In GIS, however, this task is incomparably harder, involving very complex steps (for instance transformation of arbitrary alphanumeric data into graphic format, or cartographic production).

Generalization, which is an open research area, is unavoidable in user interfaces offering different levels of visualization. There are two main types of generalization [LR93]: *abstract*, which deals with schema manipulation to provide different views of the database, and *cartographic*, which manipulates geometric objects and symbols to improve the readability of spatial data presentations. In DBMS, only abstract generalization is involved; in GIS, cartographic generalization contributes to the complexity of the problem.

The volume of data manipulated in GIS is usually very high and the interface has to provide large buffers to temporally store and manipulate the data retrieved from the spatial DBMS. Efficient management of buffers is thus a typical DBMS problem that the GIS interface must deal with. In conventional DBMS interfaces this problem does not arise.

Database integration, or schema integration, is difficult in conventional DBMS because of the different semantics of data. An important feature of GIS is the support to multiple representation of the same real world entity. For instance, a given road may be represented as a satellite image or as a graph for a transportation application. Therefore, in GIS the integration problem is harder because, beside the conflicting semantics of data, it is possible to occur incompatibility within the representations of the same data.

Finally, the conceptual model of the underlying GIS cannot be used as the representation model of the interface (contrasting, for instance, with relational DBMS interfaces). Data modeling in GIS involves spatial concepts usually expressed as low level data structures, such as lists of coordinates, which are not adequate for human spatial cognition. Graphical representations are therefore used as an intermediate model, and the burden of translation between models is undertaken by the user interface system.

There are many other difficulties for developing GIS interfaces, and they are studied in the literature in three main areas: architecture, language, and human factors. Figure 1 relates the three areas in a diagram, showing the overlappings among them. Papers dealing with architectures for GIS interfaces try to optimize the communication between the GIS and the interface system, through a internal language L_i. The main goal is to build a representation model at the interface level that can be mapped to and from GIS data models in an efficient way. In analogy to relational DBMS, the internal language, L_i, corresponds to the relational algebra while the external language, L_e, could be compared to SQL.

In general, research at the architecture level does not completely describe an external language L_e. This language is used to convey the interaction of the user with the representation model of the interface, and it is sufficiently complex to be treated as a separate field of research. The objective is to provide efficient mappings between the user mental model and the interface representation model, with emphasis on the latter model.

Efforts in the area of human factors share the objectives of those on external languages. The approach, however, is much more abstract. The definition of a mental model of the user is the main concern. This user model is mapped to the representation model, and vice-versa. Rather than being directed by a specific language, these mappings emphasize the user mental model. The target of the research is to understand how the user thinks and, based on this knowledge, to develop the appropriate representation model in the interface.

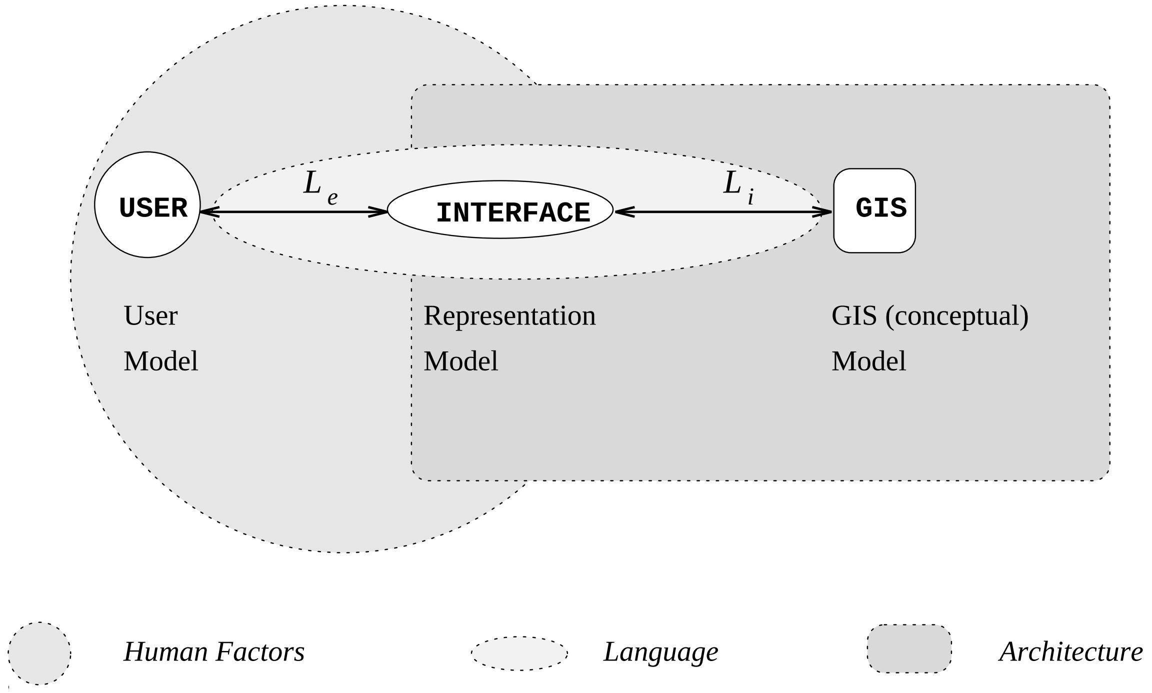

Figure 1: Research Areas in GIS User Interface

To summarize, this section has shown that building a user interface for GIS is a complex problem, which is partitioned in three areas of study. The first area tries to minimize the semantic distance between the GIS data model and the interface representation model. The other two areas have different approaches to minimize the semantic distance from the user mental model to the interface representation model. The design of a user interface for GIS has to establish a tradeoff solution for these conflicting necessities. The main solutions in each of these areas will be presented in the following sections.

3 GIS User Interface Architectures

As it was already mentioned, modern GIS make intensive use of DBMS facilities, and nowadays the DBMS is considered the heart of a GIS. The coupling of DBMS to GIS has been done according to three architectures [MP94]:

1. Proprietary systems: a special-purpose DBMS is tightly coupled with the GIS spatial modules;
2. Layered systems: a standard DBMS is used for spatial data access functions, which are usually implemented by external packages;
3. Extensible systems: third generation DBMS (extensible relational and object-oriented) embed the spatial dimension and provide language facilities to support spatial functions.

Recent research is biased to the third architecture, since the underlying DBMS of extensible systems provide powerful data models, efficient support to non-conventional (spatial) data, and uniform access facilities for both aspatial (alphanumeric) and georeferenced data.

The architecture of the geographic information system may or may not influence the architecture of the user interface system, depending on the type of connection between the two systems. The remainder of this section is divided in three subsections: the first analyses the connection of user interfaces to GIS; the second discusses interface architectures proposed in the literature; the last presents some concluding remarks.

3.1 Coupling User Interfaces and GIS

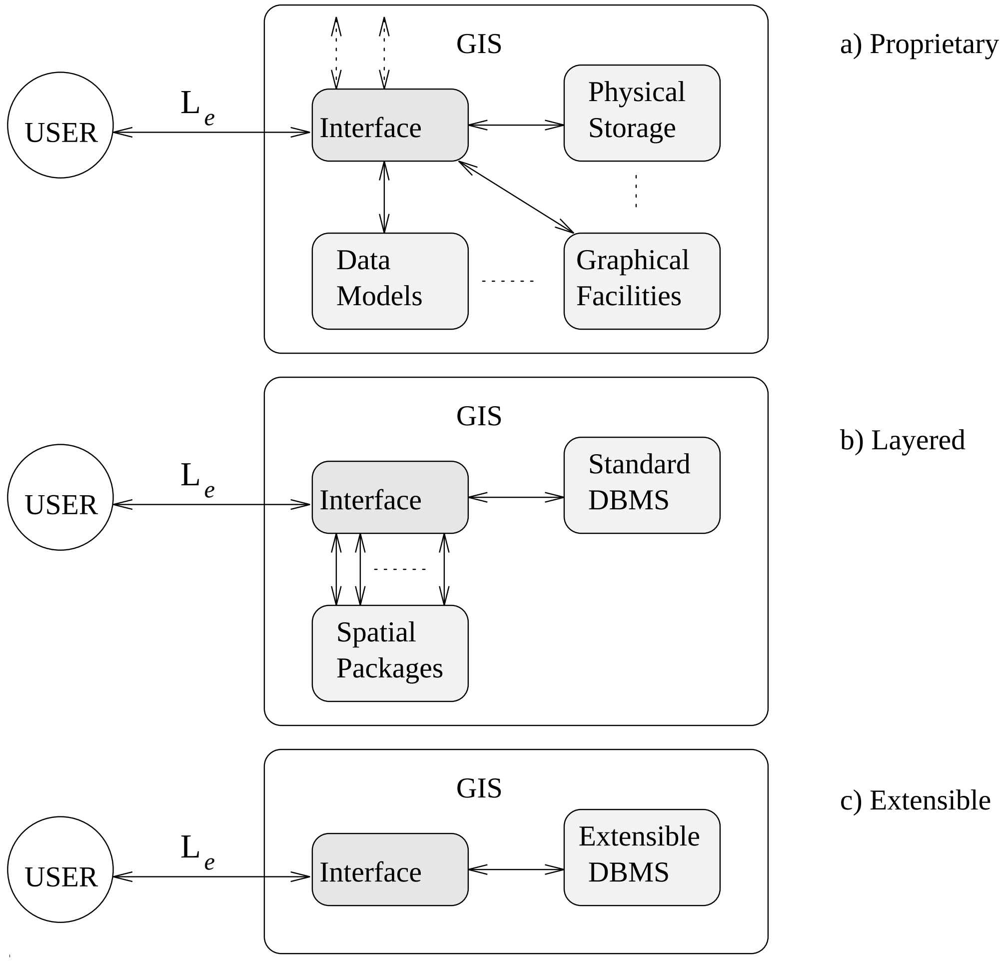

Figure 2: Strong Integration of User Interface with GIS

There are two basic approaches to integrate a user interface system to a DBMS, and in particular to a GIS: *strong integration* and *weak integration* [Voi94]. In the former, the user interface is part of the geographical system, sharing its data model and taking advantage of the knowledge about the internal data structures. As a consequence, there is great difficulty in using data from different sources, and it is not possible to adapt the same user interface to different GIS.

In this approach, the interface must "know" the internal details of the GIS architecture, which has, thus, great influence in the interface architecture. If the GIS is based on a proprietary DBMS, the interface architecture has to provide modules for interaction with each main component of the GIS (figure 2, part (a)).

If the GIS has a layered architecture, the interface architecture has to provide at least two different types of modules for communication with the standard DBMS and with the external packages that deal with spatial functions (figure 2, part (b)).

Strong integration with extensible systems demands a single module in the interface architecture to handle the knowledge about the extensible DBMS(figure 2, part (c)). This module, however, is very complex since all the DBMS features are represented there.

It can be observed in figure 2 that the degree of encapsulation of the interface increases from proprietary to extensible GIS architectures. The second approach for integration, weak integration, is based on maximum encapsulation of the user interface (figure 3).

In weak integration, the user interface considers the GIS an external module (and vice-versa), and is therefore adaptable to more than one system. The major disadvantage of this approach is that it demands the definition of communication and data conversion protocols between the user interface system and the geographical system. In strong integration this is not necessary since the interface can directly access the information in the GIS.

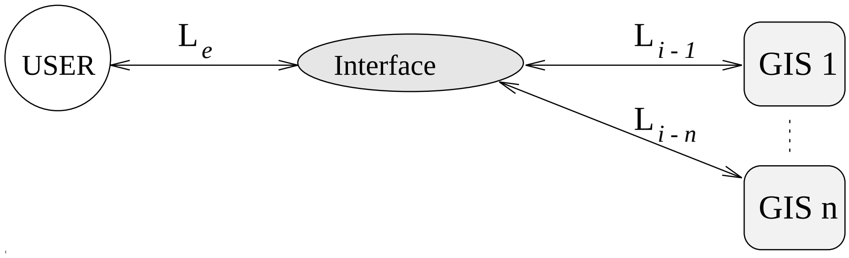

Figure 3: Weak Integration of User Interface with GIS

In spite of this disadvantage, weak integration is used in the majority of the interface systems proposed in the literature. The main reasons for this choice are [OM95]:

- There is a world-wide trend towards the development of open systems, which can be integrated with other products, and GIS are among the systems that pursue this kind of architecture (e.g., [AYA+92, VvO92, GR93, PMP93, VS94]).
- It provides independence and improves specialization of functionality of each component (user interface and geographical data management).
- It allows importing specialized packages, such as graphical libraries, into the user interface.
- It allows the progressive inclusion of new services and functions in the GIS, without affecting the interface.

- No specific modules are necessary in the interface architecture, since no special knowledge about the underlying GIS is maintained.
- The communication between the systems is based on the internal language of the GIS. A external driver in the interface manages this communication. This driver has always the same function, that is, it is always responsible for mapping the GIS data model to the representation model of the interface, through the GIS internal language. Therefore it is not difficult to build a new driver for the integration of a different GIS.

It is worth noting that there is no absolute division between strong and weak integration; as a mather of fact, the passage from strong to weak integration is gradual, and the integration with extensible GIS (figure 2 (c)) could be considered an intermediate solution.

3.2 Review of Proposed Architectures

The GIS user interface architecture has to define four important aspects. The first and most important issue is the integration approach, which was discussed above. Second, the main components (modules) of the interface system should be identified, and their functionality and interoperability should specified. Next, the interface architecture should define the interface representation model and the mapping mechanisms to and from the GIS data model. Finally, the division of tasks between GIS and interface should be clearly defined. These four aspects guide the analysis of the architectures presented bellow.

3.2.1 Voisard [Voi91]

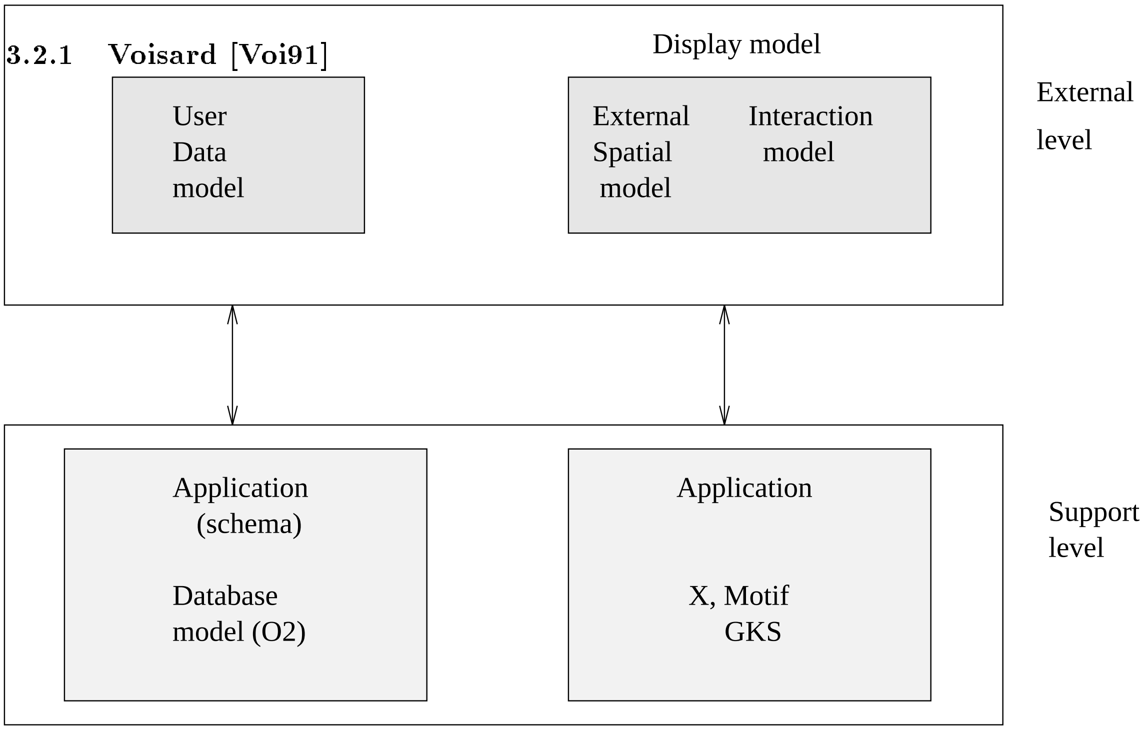

Figure 4: Architecture adapted from [Voi91]

Voisard assumes, in [Voi91], that due to the diversity of geographic applications, there is no user interface model general enough for customizing the presentation of all applications. For this reason she proposes a toolbox approach for describing particular GIS user interfaces.

An interface toolbox for GIS involves the manipulation of several models: the user data model (e.g., map metaphor), the database model (usually relational or object oriented), the data representation model (the interface intermediate model), and the map display model (visual objects manipulated in the user interface). In [Voi91] the architecture of a toolbox is described, with predefined user and database models and great emphasis on the map display model.

According to [Voi91], defining a general toolbox independent of the database data model is bound to fail, but given a data model, it is realistic to define an external spatial model independent of the application, with a mapping from one representation to the other. The proposed toolbox architecture is shown in figure 4.

The map display model has two levels: the higher one is the interaction level, which contains interaction objects such as windows and menus. A special purpose window, called Mapget, is used to display maps and to react on user events (e.g., mouse clicks). The lower level is the external spatial level, containing visual objects such as polygons, arcs and points. These objects come from a mapping of the database model.

In the interaction level, maps are organized in layers, which can be opaque or transparent. This feature is useful to visualize several maps without superimposing them. A layer-stack manages layers in a given order, and provides operations such as shift-up or shift-down to change the order of the layers. An application is basically a set of layer-stacks.

The external spatial level deals with images, which have sets of visual objects with geometric and graphic attributes. The objects are mapped from the database model, and the image uses a meta database for legend. An image is displayed in a Mapget through a presentation. A map corresponds to a single image (static definition of geographic objects), but may have several presentations (dynamic definition of visual parameters).

The proposed map display mechanism can be performed in five steps: 1) read database and meta database; 2) convert geographic data to external spatial model; 3) define interaction objects (mapgets, layers, etc); 4) compute presentation based on (user's and environment's) parameters; 5) link presentations to mapgets.

Although the paper proposes a generic architecture, only the map display model is detailed. The database model chosen was the O2 data model, but the mappings from one model to the other are not defined. In a posterior work, the author considered these drawbacks and proposed an extension of this architecture. Section 3.2.5 presents this extended architecture.

3.2.2 Abel et al. [AYA$^+$92]

[AYA$^+$92] presents an architecture model for GIS. The model has three types of entities: Collections (an aggregate of entity instances), Views (user-visible representations of data collections), and Operations (functions that act on collections and views to generate other collections and views). The system is controlled by a special operation, the user interface

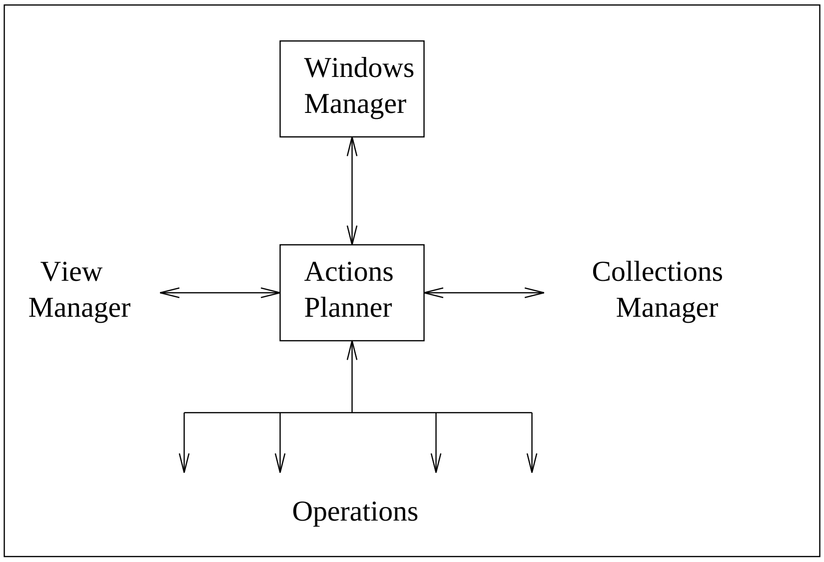

Figure 5: Architecture adapted from [AYA+92]

subsystem, which manages the specification of operations by the user and translates them to sequences of actions by the other subsystems.

All visible elements of the interface can have operations applied to them, typically by clicking on a feature and selecting an operation from a pop-up menu. Operations are grouped by entity classes. For instance, maps are entities with operations like zoom and pan while the geographic entities represented in the map may have operations like report.

The user interface architecture is shown in figure 5. The window manager deals with graphical operations under the X11 windowing system. Again, this is an implementation option; the architecture is independent from the underlying window system. The Actions Planner performs actions required by users. Operations may be initiated through a graphical user interface or by typing commands in a textual interface. Every graphical command has an equivalent textual command, but all commands which can be textually expressed need not have graphical equivalents. This is an important problem of GIS query languages, and we shall return to this subject later in this text.

An Operation can be a primitive action or a sequence of actions including loops and conditionals specified in a scripting language. This approach to the user interface differs from that of traditional graphical user interfaces. The operations are attached to objects, not to control panels. Therefore each object presents only the applicable operations. Moreover, maps are objects of the interface, with associated operations, rather than simply results of operations.

3.2.3 ESRI [Env92]

The ARC/INFO user interface architecture is strongly connected to the GIS. In fact, the user interface components are distributed in the set of tools that compose the geoprocessing system. Nevertheless, the user interface is coherent because every tool is based on the same

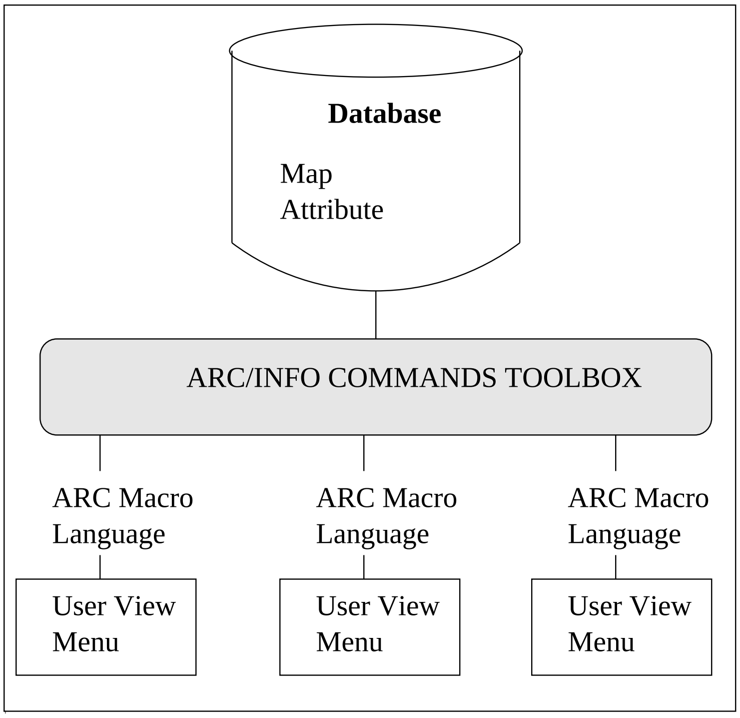

Figure 6: Architecture adapted from [Env92]

data model.

The ARC/INFO data model is a hybrid georelational data model that uses a topological data structure to store a collection of coverages, which are basic units of vector data storage and can represent several types of geographic features [Env92].

The tools and applications of ARC/INFO build specific data models on the base generic data model. For instance, ARC/INFO provides separate tools for data entry and edit, data analysis, data management, custom application building, and cartographic production. Each of these has its particular data model, based on the generic model, and its own user interface components. Moreover, ARC/INFO offers a user interface toolkit allowing application programmers to create application-specific graphical user interfaces through standard widgets.

ARC/INFO is currently one of the most widely spread commercial GIS. Its user interface architecture, however, does not give much contribution for a general solution of the problem, for two main reasons. First, each tool has its own strongly connected user interface component. Thus, it is difficult to separate the application and the user interface components of each tool. Second, the tools are strongly based on a particular data model. The concepts of the ARC/INFO data model are not easily mapped to a generic GIS data model.

3.2.4 Pissinou, Makki and Park [PMP93]

The user interface architecture proposed in [PMP93] consists of several modules which provide generic services at the user level, which are translated into different procedures at

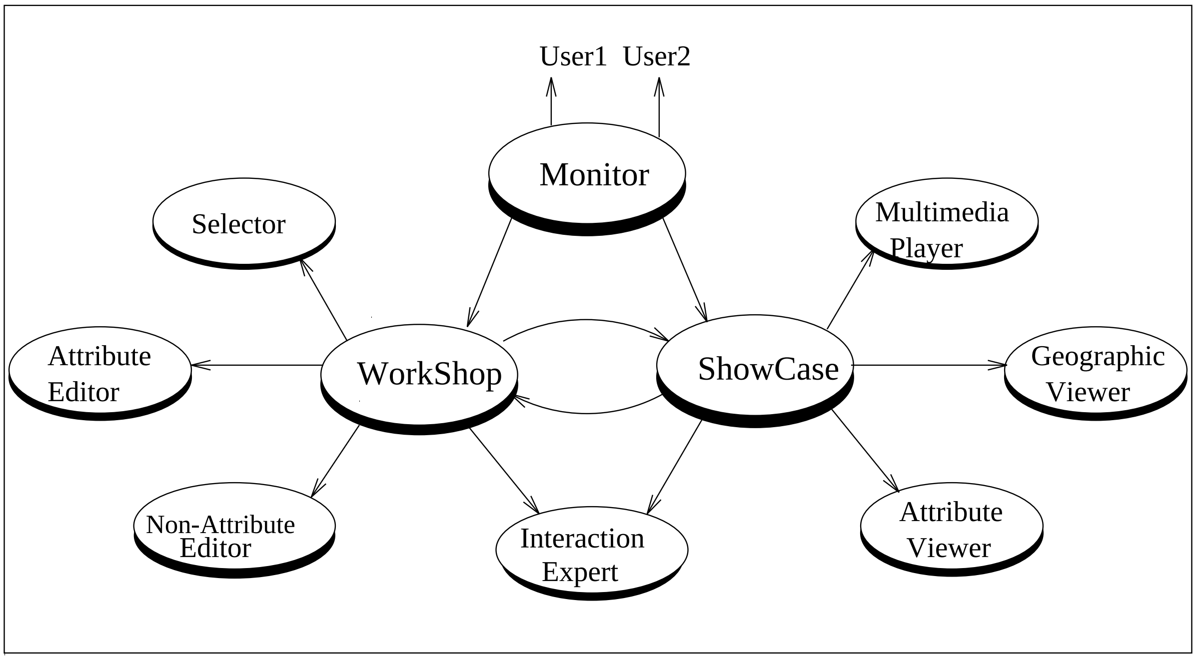

Figure 7: Architecture adapted from [PMP93]

lower levels of the GIS architecture. The objective of the interface is to require minimal knowledge from the user for accessing the GIS functionality.

The architecture is *multiagent*, that is, it is structured into a collection of specialized and complete information processing systems (the agents) that produce and react to events. The main agents of the architecture are the Monitor, the Showcase and the Workshop (figure 7).

The Monitor's main function is to control the overall consistency of the agents. It can detect any operation of any agent in the system, decide if the operation has a side effect, in which case it sends an update message to the affected agents. The agent receiving the message is responsible for its interpretation. This behavior is useful in distributed and cooperative work. Distributed displays allow the creation of multiple views of the same collection of objects. In order to control the consistency of a multiuser distributed environment, the monitor maintains a dynamic *semantic view* dictionary of the system.

The Showcase agent is responsible for coordinating its cooperative agents, while the Workshop agent provides a complete user interface to manipulate the GIS domain specific databases. Since agents share common properties they can be grouped into classes which define the protocol which can be understood by an agent's children (the processes spawned by that agent).

The proposed user interface forms the second layer of the four layer GIS architecture proposed by [PMP93]. The first layer comprehends high level GIS tools (used for knowledge discovery, data integrity and control, hypermedia management and decision support). The layers below the user interface are the object oriented deductive temporal GIS model and the GIS engine. The former represents both spatial and temporal knowledge about georeferenced entities, providing procedures to perform operations such as comparison, derivation, prediction, validation, and visualization of data. The latter includes a rule manager, an explanation manager, a data access module, a meta-data manager and a multimedia manager in order to provide close integration of data access with the inference engine and efficient support to multimedia storage and display.

Since [PMP93] does not provide insight information in how the four levels are connected, it cannot be asserted whether the user interface is strongly or weakly integrated with the GIS.

3.2.5 Voisard [Voi94]

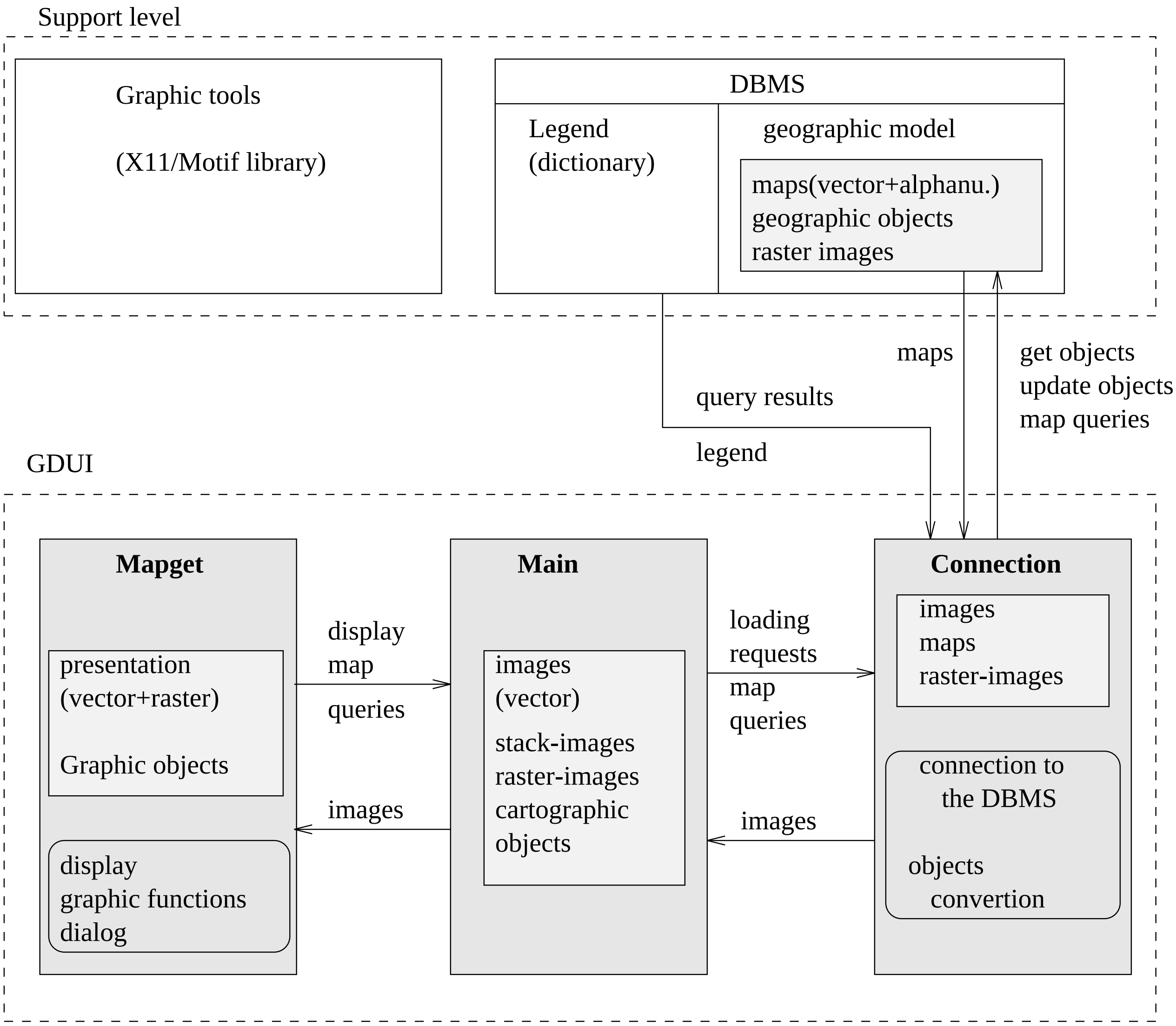

Figure 8: Architecture adapted from [Voi94]

[Voi94] proposes a toolkit based solution for the design of user interfaces for GIS. The architecture is an extension of a previous work of the author (see section 3.2.1). Towards this solution Voisard classifies GIS interface functions in three categories:

- Visual representation: display spatial (raster and vector) data; display description (alphanumeric) data; overlay maps; build and attach legend to map (color, pattern, text, symbol); change map scale; display different representations of data.

- Map manipulation: select geographic objects from the map; draw geometric objects to parametrize operations; modify (spatial and description) properties of objects; formulate queries; browse in meta data level.

- Database interaction: get (geometry and description of) objects to be displayed; define representation based on the object's properties; update objects in the database; query the database.

This categorization of functions provides the requirements for the interface architecture. Unhappily, the paper does not make a clear partition of the operations into the identified categories. It is not clear, for instance, which query facilities are provided by the DBMS and by the interface, since query operations appear as both Map manipulation and Database interaction functions.

The main contribution of [Voi94] is a detailed analysis of the integration problem. After a rich discussion of this subject, the paper concludes that the requirements of GIS interface and the drawbacks of the weak and strong approaches demand an alternative solution: to offer tools for developing customizable interfaces within a general database environment. This proposal is the basis of the two level architecture presented in figure 8.

The support level contains two orthogonal modules. The graphic tools module is used mainly for visual representation functions. In [Voi94] it is represented by the X11/Motif library, but this is an implementation decision. The architecture is independent from the graphic package used in the graphic tools module.

The second module of the support level is the DBMS. This module provides the general database environment of the alternative solution for the integration problem. However, it can be easily seen from figure 8 that the architecture imposes some restrictions on the DBMS model. The DBMS must provide the legend dictionary and the geographic model defined by the architecture. The alternative solution proposed may use any DBMS which can provide these services. Since the architecture of the interface is not completely independent from the architecture of the geographic system, we can consider this alternative solution as a variation with minimum dependence of the strong integration approach.

The second level of the architecture is represented by a geographic data user interface (GDUI) kernel composed by three main modules. The Mapget module is responsible for the user dialog and for the presentation of maps in the interface windows. Therefore, it is strongly connected to the Graphic Tools module of the support level. The second module is called *Main*, and it deals with abstract views of maps, controlling the communication between the Mapget and the Connection modules. The latter is the real "main" module of the architecture. It knows how to convert data from database to interface and vice-versa. It is also in charge of handling all exchanges with the DBMS. This module is not so complex as in a weak integration, since the DBMS is forced to offer a predefined geographic model.

3.2.6 Oliveira and Medeiros [OM95]

The architecture proposed by Oliveira and Medeiros [OM95] is based on the weak integration approach. The user interface can be coupled to any GIS, and this feature demands a rich representation model to map concepts used in different GIS. The object-oriented GIS model of [CFS+94] was adopted.

This data model supports both the field and the object views of the geographical world. It consists of four levels of abstraction: the real world level (the real geographical phenomena); the conceptual level (an abstract view of these phenomena, in which operations are independent from the representation of the data); the representation level (where operations are specialized to each particular representation of a geographical entity); and the physical level (which deals with issues that provide efficient storage and retrieval of data, such as spatial indexes and access methods).

Georeferenced entities are conceptually classified into geo-objects (object view) and geo-fields (field view). Each such class has its own high-level operations and particular representations. The interface representation model supports the multiple representation paradigm, in which a given georeferenced phenomenon may be perceived differently according to application needs. By supporting this model, the interface architecture establishes a framework for two important end-users requirements: users can manipulate multiple representations at the interface level; and users can define distinct presentations (display characteristics) for each representation.

The proposed architecture, shown in figure 9, has three main components:

- The *user dialog* module manages the user interaction with the interface system. It is responsible for two main tasks: the creation and management of presentations, and the translation of user's requests into operations of the underlying geographic database model (and vice-versa). These tasks are performed, respectively, by the presentation manager and by the interaction manager. The binding with the graphic toolkit is another important task within the user dialog module.

  The definition and management of presentations are handled, in the screen, through two areas: a control area for query definition and a display area for result visualization. This task involves graphical display operations (display area) and dynamic construction of widgets (control area). Each representation of the data can be visualized through different presentations (e.g., graphical, diagrams or tabular).

  The second main task is the translation of direct manipulation actions of the user into high-level conceptual operations on georeferenced data, managed by the data model module.

- The *data model* module is responsible for providing to the user a view of the underlying database which is compatible with the adopted representation model. The conceptual manager is responsible for the object-oriented schema that describes geographical entities. The representation manager records the different representations associated with each conceptual entity.

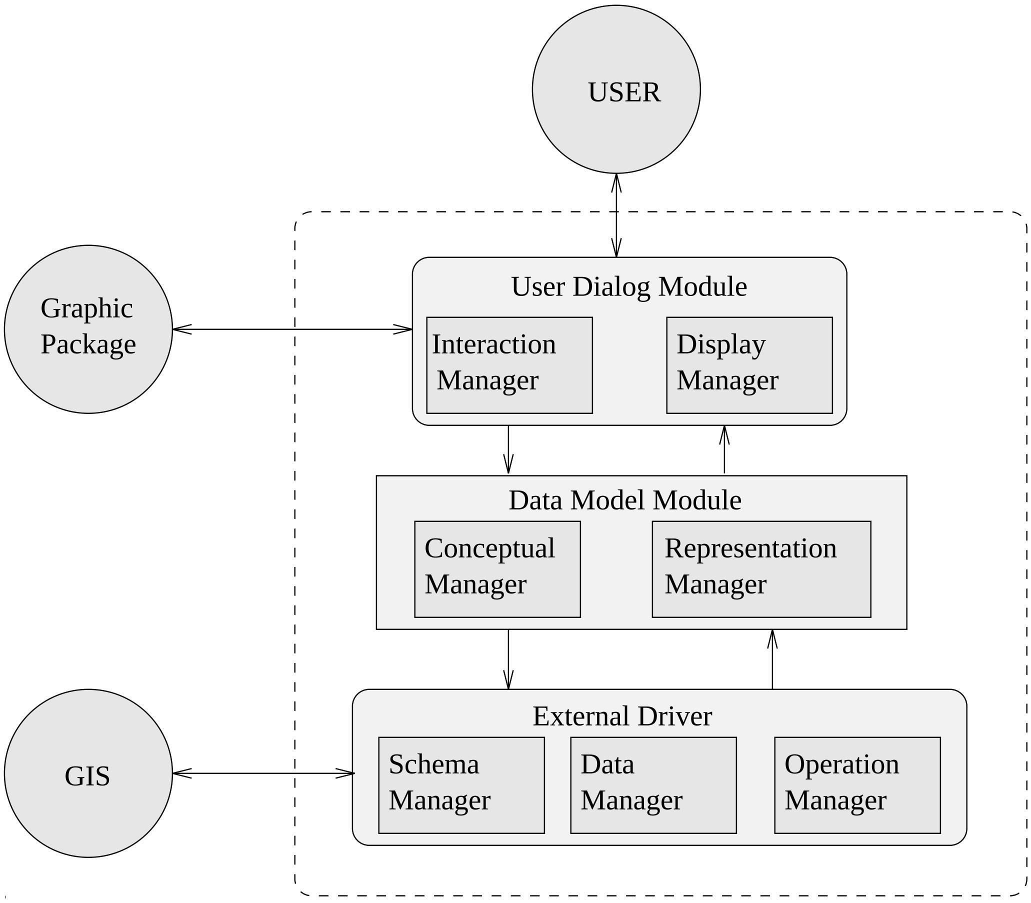

Figure 9: Architecture adapted from [OM95]

The main task of the data model module is to support a high level conceptual view of data. It therefore supports browsing on concepts rather than on representations, allowing the user to see multiple representations of the same data. Another important task of this module is to convert conceptual operations into representation dependent operations, which are sent to the external driver.

- The *external driver* converts data from the format used in the GIS to the internal data model of the interface and vice-versa. This is achieved by means of a communication protocol that is based on primitive operations that allow retrieving from the GIS database schemas, class descriptions and data values.

  The approach used in the definition of this module was already used to integrate a user interface to different object-oriented database systems [OA93b]. The interface sends queries to the GIS using the primitive operations (Get-Schema, Get-Class, and Get-Value), and the external driver implements these operations according to the syntax of the underlying geographic DBMS.

Although figure 9 shows only one external driver and one GIS, the architecture is perfectly able to deal with many different GIS, depending on adding new external driver modules.

The architecture of the interface system supports distinct conceptual views of the geographical space. Each conceptual view corresponds to an object-oriented database schema built from the underlying GIS schemas according to the definition of the conceptual level of the representation model.

3.2.7 Rigaux [Rig95]

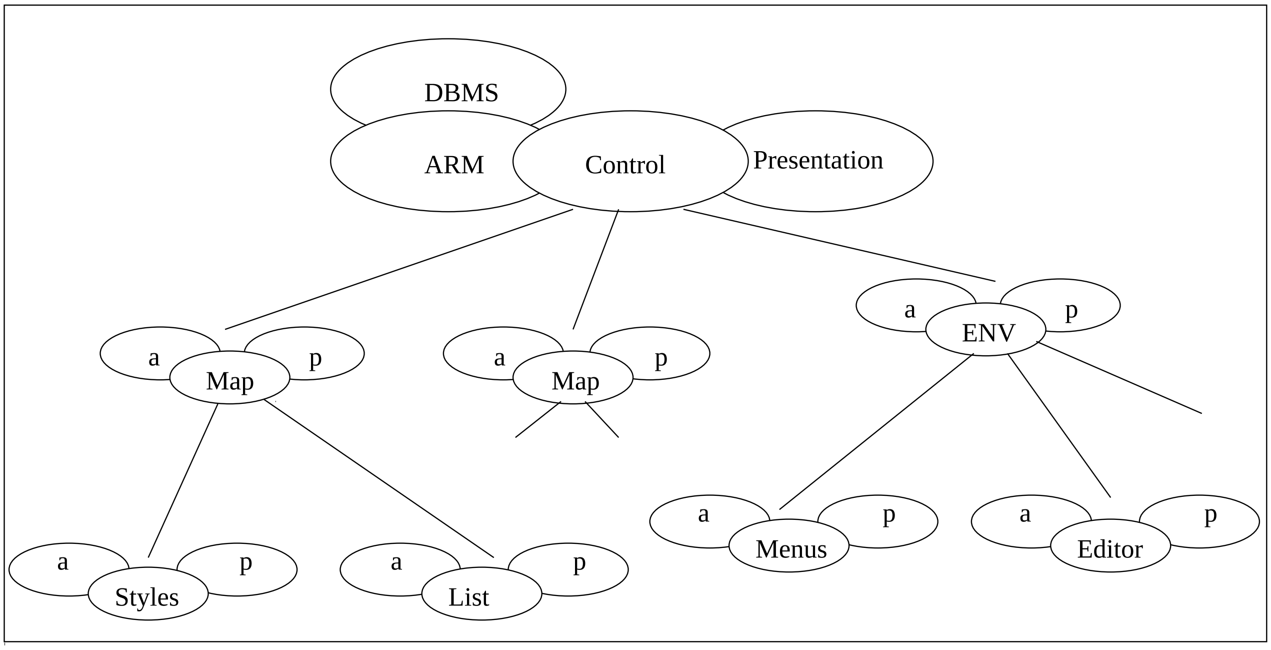

Figure 10: Architecture adapted from [Rig95]

The user interface architecture proposed by Rigaux has three levels of representation in order to provide independence between database and interface [Rig95]. The lower level is the conceptual model of the DBMS, followed by an abstract representation model and finally a user representation model.

Rigaux claims that there are essentially two kinds of operations in the interface: database operations (e.g., query, browse) and graphic operations (e.g., graphic overlay, representation control). The abstract model supports the graphic operations, while the conceptual model deals with database operations. The user model contains objects that are directly perceived by the user (e.g., maps, representation styles, legends).

The levels of the architecture provide a certain degree of independence between user interface and DBMS, but it introduces the necessity of translation between the models. In [Rig95] the translation mechanism is formally defined.

Figure 10 presents the architecture as a composite object, using the PAC interface model. This model is based on the classic decomposition of interactive objects in three components: presentation, abstraction, and control, but permitting recursive decomposition of each object.

The main module of the architecture has an abstraction component formed by the two lower levels (DBMS and abstract models). The user model defines maps, which are complex objects composed by graphic styles and lists of thematic layers. All maps are managed by the higher level control component, which guarantees the coherence of the graphic representations.

The user interacts with the presentation component of the PAC objects. The main presentation objects are the maps. Secondary objects (lists, styles) control the parameters of the presentation. The ENV object represents the environment, that is, all static objects in the interface (windows, buttons, etc).

3.3 Remarks on GIS User Interface Architectures

From the discussion presented above, we can distinguish two kinds of GIS architecture proposals. In one side are those architectures that, once implemented, give the user a direct access tool to the GIS[Env92, PMP93, OM95, Rig95]. On the other side are those architectures that offer a common substract to develop GIS user interfaces for specific applications [Voi91, AYA+92, Voi94].

We could also identify four key aspects of GIS user interface architecture, namely: integration of the interface to the GIS; functionality and interoperability of the main modules; intermediate representation model and mapping to GIS models; and division of tasks between user interface and GIS.

All research dealing with architectures for GIS user interfaces consider one or more of these aspects. However, none of the analyzed papers has proposed concrete solutions for the complete set of fundamental aspects.

4 Languages for Interaction with GIS

Current user interface systems are based on two main paradigms of interaction: textual and visual. Textual interfaces are the oldest and most diffused media for interaction with computers. Operating systems command languages and traditional database query languages are examples of pure textual interfaces.

Visual interfaces are a recent trend and are receiving much attention given their increasing acceptance in the users community. Visual interfaces use graphic objects, colors and symbols, rather than texts as the basis for the user interaction. Direct manipulation interfaces are the most popular visual interfaces; for instance, virtually all window managers are based on this paradigm of interaction.

Since GIS deal with spatially referenced data, it is not difficult to realize that graphical representations are more suitable in most cases for human cognition than texts. Consider, for example, the presentation of regions of interest within a given area. Textual interfaces would represent such data as lists of coordinates while visual interfaces would present polygons within an area. The second option is surely better understood by users. On the other hand, some features are better described by text (such as content of an area, or non-spatial values). Therefore, GIS user interfaces have to use, inexorably, visual and textual paradigms for presentation of query results. The problem that remains open is: how to formulate queries concerning spatial data and how to answer them in a meaningful way.

In the remainder of this section we review several GIS query languages presented in the literature. This will show that, in practice, GIS user interfaces use an *hybrid solution*, combining textual and visual interaction. We finish this section with an analysis of this supremacy of the hybrid solution over the pure visual solution.

4.1 Review of Proposed Query Languages

One important requirement of GIS query languages is that it has to be possible to manipulate the results of a query to directly compose further queries. This distinctive feature and the definition of spatial operators are the main aspects that should be considered in the design of a query language for GIS.

There are two basic approaches to this design problem. First, it is possible to extend a standard database (textual) query language to allow the specification of spatial predicates. Second, it is possible to develop graphical tools to formulate queries by direct manipulation of interface objects. A visual query language is one which express the semantics of the query by a drawing (and not just by a click-in-a-box metaphor to build an alphanumerical query).

Theoretically, the first approach would generate hybrid interfaces combining textual and visual interaction, while the second approach would give origin to pure visual interfaces. Although pure visual interfaces sound interesting, we show in this section that the great majority of existing GIS user interfaces uses the hybrid solution.

4.1.1 Egenhofer and Frank [EF88b, EF88a]

Egenhofer and Frank point out two major deficiencies of the traditional (relational) query languages. First they were designed for the relational data model and can not be easily accommodated to other models; furthermore, tables are far from the spatial mental model of GIS user. Second, they are designed to serve both as interactive and embedded query language, and these contradicting demands are an impediment for designing user-friendly languages.

To solve these problems, the authors propose an object-oriented interactive query language for GIS. The object-oriented model gives the user a suitable vision of the real world, in opposition to relational tables. The model is also semantically rich and can be easily mapped to different models. Since the language is exclusively interactive, it can be optimized for user dialog issues.

The definition of the language is based on three aspects of GIS interaction: interface layout, dialogue and representation. To support the user's analysis, the interface layout must provide visualization of both query formulation and results. Since qualitative analysis is often based on visual comparison, the interface should permit several views of the workspace. The proposed layout has six areas: 1) lexical output; 2) graphical output; 3) lexical input; 4) operations control panel; 5) map features menu; and 6) graphical representation menu. Figure 11 shows these areas.

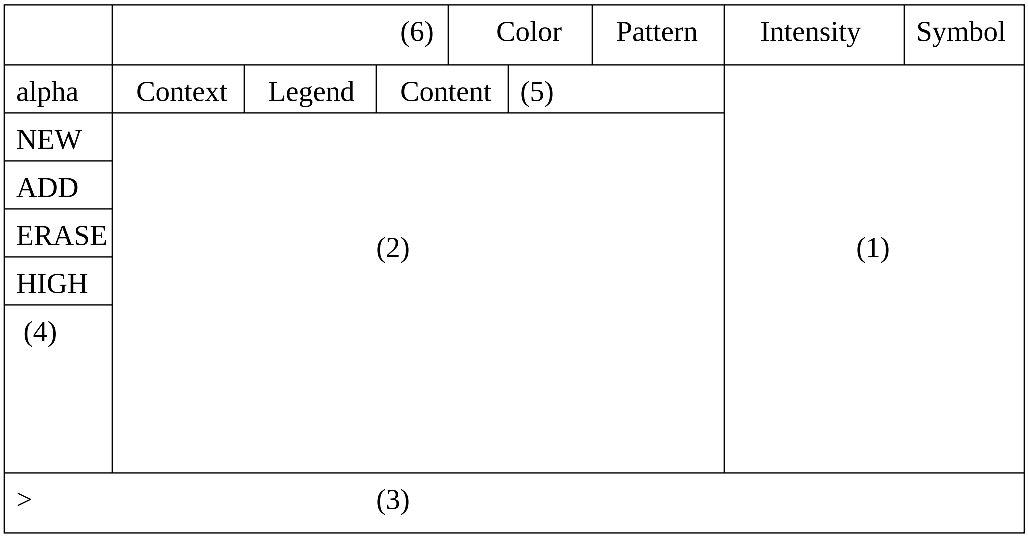

Figure 11: Interface layout adapted from [EF88b, EF88a]

Query formulation is based on either the user's knowledge or on information provided by the system. Therefore, dialogue with GIS must incorporate means to reference objects or regions on maps. The dialogue is performed in the following way: the user inputs keywords for selection and is prompted to select objects; the selection is verified, but the user can alter his choice; finally the user confirms the selection. The interface should provide feedback on object selection, informing the user which class of object is expected. Immediate verification of selected objects is also important to guarantee user's control over system actions.

The knowledge represented in GIS can be represented both lexically and graphically. The lexical representation can be partially covered by a traditional language style: users type queries from a keyboard and the result is presented textually. Graphical representation adds constraints to the interface that are not considered in textual techniques. The limitation of the screen size originates cartographic generalization problems. The authors suggest that useful screen drawings should contain about 2000 to 5000 objects. Since the interface layout has separate areas for lexical and graphical output, queries involving both types of results are allowed.

The proposed query language is based on the addition of four operations to an extended spatial SQL. The *new* operation refreshes the display before performing the next operation; the *overlay* operation adds the result of the current query to previously displayed objects; the *remove* operation erases (only) the contents of the current query from the existing picture; and the *highlight* operation distinguishes the results of the current query from the previous ones. These operations are in the control panel area of figure 11.

With this language, it is possible to combine several query results into a single rendering. This composition of queries, however, makes it difficult to keep track of what is actually presented. An essential feature in such a query environment is a control mechanism allowing the user to check, after a sequence of queries, what a single query for the current drawing would have been. The authors claim that in their interface the *content* button provides a simpler functionality, showing the predicates applied to each class of objects in the current map.

The presentation must be controllable from the interface, allowing users to adopt the most appropriate presentation for their applications. The proposed language allows users to choose colors, patterns, intensities and symbols from pull-down menus (figure 11). This choice composes a map legend that can be visualized through the *legend* button.

Unlike lexical presentation in traditional DBMS, in GIS it is often not sufficient to draw only those objects that were explicitly asked for. It is necessary to consider the selection of appropriate context which depends on the purpose of the drawing, the scale, and the data density. The proposed interface provides a *context* button to specify a context for a drawing.

4.1.2 Güting [Gut88]

Class	Operator	Operands	Result	Syntax
Relational Operators	$+, -, *, /$	NUM x NUM	NUM	(_ # _)
	and, or	BOOL x BOOL	BOOL	(_ # _)
	not	BOOL	BOOL	#(_)
	$\cup, \cap, -, \times, \bowtie$	REL x REL	REL	_ _ #
	σ, π, λ	REL	REL	_ #
	$=, \neq, <, \leq, \geq, >$	NUM x NUM	BOOL	(_ # _)
		STR x STR	BOOL	
		BOOL x BOOL	BOOL	
	count	REL	NUM	(_ #)
	sum, avg, min, max	NUM*	NUM	(_ #)
	extract	REL	ATOM	(_ #)
Geometric Predicates	$=, \neq$	POINT x POINT	BOOL	(_ # _)
		LINE x LINE	BOOL	
		REG x REG	BOOL	
	inside, outside	GEO x REG	BOOL	(_ # _)
	intersects	EXT x EXT	BOOL	(_ # _)
	is_neighbor_of	AREA x AREA	BOOL	(_ # _)
Geometric Relation Transformers	intersection	LINE* x LINE*	POINT*	_ _ #
		LINE* x REG*	LINE*	
		PGON* x REG*	PGON*	
	overlay	AREA* x AREA*	AREA*	_ _ #
	vertices	EXT*	POINT*	_ #
	voronoi	POINT* x REG	AREA*	_ _ #
	closet	POINT x POINT	REL	_ _ #
Returning Atomic Objects	convex_hull	POINT*	PGON	(_ #)
	center	POINT*	POINT	(_ #)
		EXT	POINT	# (_)
Geometric Operators Returning Scalars	dist	POINT x POINT	NUM	# (_ , _)
	mindist, maxdist	GEO x GEO	NUM	# (_ , _)
	diameter	POINT*	NUM	(_ #)
	length	LINE	NUM	# (_)
	perimeter, area	REG	NUM	# (_)

Figure 12: Geo-Relational algebra adapted from [Gut88]

Two main problems in the development of geometric database systems are the design of a user model of data and the specification of efficient strategies for implementation of this model. Güting proposes an extension of the relational algebra, including geometric data types and operators, as a solution for these problems. The author presents an extension of the relational algebra, and shows how it can be implemented by integrating geometric algorithms and data structures. The basic motivation is that standard relational representations and file structures are not efficient representations for geometric objects.

Objects of the proposed *Geo-Relational* algebra can be atomic values such as strings, numbers or boolean, as well as relations. The operators of the algebra include arithmetic operators on numbers, such as addition, and relational operators, such as selection. The basic idea of the framework is the introduction of new data types, and corresponding operators, for geometric objects.

The representation of geometric objects is hidden from users (they are only accessible by the introduced geometric operators). A tuple of a relation describes an object through geometric and non-geometric attributes. A relation describes a homogeneous collection of geometric objects. The data types defined in the algebra are: numbers, strings, booleans, relations, and geometric types (e.g., points, lines, regions, areas, polygons). The distinction among geometric types is usually semantic. For example, PGON and AREA are polygons occurring as an attribute of a relation, but PGON polygons may intersect each other while this is not possible with AREA polygons. Following the same principle, a REG is any type of region (PGON, AREA); an EXT is any extended object (LINE, REG, but not a point); and a GEO is any geometric object.

The operators of Geo-Relational algebra are grouped in five classes:

1. Standard relational operators: allow the usual retrieval and data manipulation tasks of relational databases, besides some extensions. For instance, selection and join operators can take an arbitrary algebra expression of result type BOOL as a parameter. The λ operator allows to dynamically extend an existing relation by a new attribute. For each tuple, a value for the new attribute is computed as the result of an expression given as a parameter of the operator. Finally, the *extract* operator allows to extract an atomic value from a relation based on two parameters: a selection condition which identifies a single tuple of the relation and an attribute name for the extracted value.

2. Geometric predicates: compare two geometric objects in different manners. The comparison can be part of the parameter expression of a selection or join operation. These operators can be associated to formal functions. For example:

   (x inside y) := points(x) $\subseteq$ points(y)

3. Geometric relation transformers: take one or more geometric relations as operands and produce a result relation, embedding new geometric objects. The intersection operator can be applied to two sets of lines and it constructs all intersection points between lines of the two sets. The overlay operator combines two partitions of the plane into disjoint regions which are the intersection of a region of the first with a region of the second operand. The vertices operator returns the vertex points of extended objects and the voronoi operator constructs the Voronoi diagram for the given points inside

the given region. Finally the closest operator takes a set of points and returns the point in this set that is closest to the point given in the second operand.

4. Operators returning atomic geometric objects: the first operator of this class constructs the convex hull of a set of points, defined as the smallest convex polygon enclosing all points. The second operator determines the center of either a set of points or an extended atomic geometric object.

5. Geometric operators returning scalars: these operators perform the metric operations suggested by their names. The diameter operator is the only requiring further explanation: the diameter of a set of points is the largest distance between any two points in the set.

Figure 12 presents the operators in each class. The notation ATTR* used on the left hand side of an arrow means a column of a relation of type ATTR; on the right hand side the meaning is that the result relation will have a new attribute of type ATTR. The syntax is defined by two basic rules: 1) use standard (infix or prefix) notation, except on operators taking a relation as an operand, which demands postfix notation; 2) parentheses have to be put as indicated, and also in results originating atomic objects. The symbols "#" and "_" represent the operator and the operand, respectively.

Some operators take parameters besides operands. The distinction between operands and parameters is that operands are objects of the algebra, while parameters are additional information for the operator. For instance, the projection operator (π) takes a relation as an operand and the projection attributes as parameters. Parameters appear in square brackets after the operator symbol.

It is possible to establish a direct correspondence between operators of the algebra and well-know geometric algorithms, allowing an efficient implementation. Nevertheless, the proposed language is not appropriate for end-users. Moreover, the lack of regions with holes is a major limitation for the general application of the algebra in GIS.

4.1.3 Goh [Goh89]

In [Goh89], Goh investigates the use of standards for a generic interface for access to land data banks. The proposed interface is based on SQL for database query and GKS for graphical functions. The objective of the interface is to provide retrieval access to land information in a heterogeneous computing environment.

Four factors influenced the development of the interface:

1. Selective *ad hoc* information retrieval of both textual and graphical data is essential;

2. The interface must be supported by accredited international standards so as to preserve its generic nature and ensure portability;

3. Only relevant graphical data should be retrieved and displayed. Functionality is a major concern, rather than generation of sophisticated output of high quality cartographic standards;

4. Queries are expected to be performed from remote terminals. Thus, minimization of the volume of data in the course of communication is mandatory.

The proposed interface was called GQL. It invokes SQL function calls through application programs. The implementation of GQL requires, therefore, the facility for embedding SQL function calls in a host programming language such as Fortran or C.

The input and output functional capabilities of GKS are organized into nine levels, and GQL can work at the lowest functional level. The basic output primitives (e.g., polyline, fill area, cell array) can have its appearance (color, linetype, fill pattern) controlled by GKS primitive attributes, which can be set by users during an interactive GQL session. The primitive output *text* is an exception. Different systems view text from different perspectives. For example, CAD/CAM systems regard texts as an integral part of the graphical data structure while ARC/INFO can store them either as a graphic element or as part of the textual database.

GQL allows the expansion of a query result to include a graphical component, provided that the particular item has an equivalent graphical definition in the database. The ampersand (&) prefix is the graphic modifier for the desired item, and it is the only addition to the ISO SQL *select* command syntax.

The graphical environment must be set before GQL is invoked for graphical queries. A *define* command is introduced to control this environment. Besides the control of the primitive graphical attributes and the limits of the display (the window), no elaborate interactive graphical editing and manipulation functions, such as scaling and movement of graphical primitives, are permitted. A graphic data dictionary maintains a listing of the entities which have associated graphic components. This allow the user to know which are the entities that could be displayed graphically.

Ideally, a generic query language should support both vector- and raster-based databases with identical query syntax. The query language described in [Goh89] supports only vector-based databases, although the author claims that the extension to support raster data is not complex. Another drawback of GQL is that it restricts the data modeling: it demands that the linkage between the graphic and textual databases be performed through a unique identifier that associates a graphic primitive with a key in a relational table.

4.1.4 Ooi [Ooi90]

[Ooi90] presents GEOQL, an extension of SQL which provides spatial operators to express spatial and aspatial predicates in queries. The following operators are defined: *intersects, adjacent, joins, ends_at, contains, situated_at, within, closest*, and *furthest*.

Since the definition of a closed set of spatial operators is an open research subject, the proposed set limits the expressive power of the language. There are many other types of operators that are not allowed in GEOQL. For instance, topological operators like "east of", metrical operators like "distance", and linguistics or fuzzy operators like "close" and "far".

The proposed language relies on the relational model of the database to guarantee that all aspatial information about entities of a given class will be stored in a relation, and that a given relation holds information about entities of only one class. Therefore, a table name can be used to identify the type of an operand, i.e. a geographic entity class, in a spatial expression. This is important since some operators are not commutative (e.g., *ends_at* and *contains*).

GEOQL provides a windowing facility allowing a query region to be specified by its two corner coordinates pointed in the screen. If there is no predicate involving a window definition, the query predicates are conjuncted with the predicate *window contains geo_obj*, for each geographic entity referenced in the query. The zoom operation is implemented by this mechanism: only the window delimited area is drawn on the output map. Figure 13 shows an example of query in GEOQL.

"Find all cities with a population over 5000 within a radius of 200 km of Mt. Buffalo in the current window drawn in the screen."

```
SELECT CITY.Name
FROM   CITY, MOUNTAIN
WHERE  MOUNTAIN.Name = ''Buffalo'' and
        CITY.Population ≥ 5000 and
        CITY within 200 km of MOUNTAIN and
        window contains CITY and
        window contains MOUNTAIN.
```

Figure 13: Example Query in GEOQL adapted from [Ooi90]

Three types of spatial data are supported: points, lines and regions. No spatial division is imposed on data, that is, queries may range over all the stored information. Aspatial data are stored as relations and spatial data are kept in an external file. An attribute of geographic relations identifies each individual geographic object and is used to retrieve the associated spatial data.

4.1.5 Calcinelli and Mainguenaud [CM91]

The authors note that geographic applications distinguish between network-oriented and thematic-oriented queries due to the lack of common data structures and an unified data manipulation language (DML). They present the unification of two previous graphical DML to avoid this dichotomy of the spatial query languages. Furthermore, they observe that ambiguities may arise from this unification, and propose solutions for three classes of ambiguities: visual semantics, level of abstraction and query labels.

The network-oriented DML *Crog* models spatial data by directed graphs and supports four classes of queries: path evaluation (going from one place to another under constraints); intersection of paths (common sub-paths); inclusion of paths (path evaluation with specific sub-paths); and node manipulations (common places between two paths). Manipulations are based on evaluation of paths, including operations on edges (direct link, transitive link, intersection of links, inclusion of edges), on nodes (intersection of nodes, inclusion of nodes), and on edges and nodes (inclusion of nodes in a path).

A graphical query is a set of labeled and oriented graph-like structures. The labels can be variables or constants, and three types of edges are available to model link, inclusion and intersection. Edges are oriented binary operators which may represent the results of a sub-query (a set of paths).

The thematic-oriented DML *Cigales* defines two basic objects: line and area. Each object is characterized by a set of attributes and object are organized in a simple inheritance hierarchy. The geometrical operators available include: adjacency, inclusion, intersection, path and euclidian distance. To build a query, the user selects basic objects, associates semantics to these objects trough identity labels, and assembles objects by applying spatial operators. Two spaces are defined: the working space, used to build a part of the query, and the query space, which contains user validated query formulations.

Merging Crog and Cigales is a good idea, since the first lacks "area" operations and the former does not manage "path" manipulations. Cigales is the base for the new improved language. Unfortunately, ambiguities appeared in the merging process. A graphical query is said to be ambiguous whenever several interpretations could be performed for the same expression. Three types of ambiguities were identified by the authors:

- visual semantics: the semantics of each operator is clearly defined, but the composition of operators may lead to misunderstanding.
- level of abstraction: Crog is oriented toward a logical representation (e.g., an edge does not refer to any physical representation) while Cigales is oriented toward a symbolic graphical representation (i.e., all operators refer to graphical manipulations).
- query labels: there is not an unique way to express the notion of order on a set of objects. Object identity specification is also problematic when composite predicates are expressed.

Two principles can be adopted to resolve these ambiguities. First, a user solution can be asked. This solution increases the needs of communication with the user, and tend to be boring. Second, default semantics can be defined for each case of ambiguity. This increases the complexity of the resolution model, but simplifies user's actions. The second principle was adopted in [CM91].

The independence between operators is defined to solve visual semantics ambiguities. There is no link between operators, unless this link has been explicitly specified. The second type of ambiguities (level of abstraction ambiguities) are derived from the common basic operations between the two languages: intersection and inclusion. For each application of these operations the system assumes a default semantics following the types of the involved objects. For instance: *an intersection between two lines is a network oriented query*. Query label ambiguities were resolved by the introduction of a selective union mode, allowing the decomposition of an object into several atomic parts. The selective union operator, however, is not yet implemented.

The graphical user interface for the unified language has four zones: in the middle zone are the working space and the query space; the upper zone provides buttons for save, help and end functions, in the right side; in the left side there is a selection button for selection of objects on the query space, and two menus: working and request. The first offers validation, undo, and clear functions in the working space while the latter provides the same functions applyed in the query space. The right zone contains two menus and two icons. The icons represent the basic objects: line and area. The utilitarians menu has options such as zoom, specific display, and operator semantic modification, while the aggregate menu contains functions such as min and max. The left zone supplies the graphical operators (inclusion, intersection, adjacency, path and euclidian distance).

The authors recognize that the management of complex queries is a major drawback of the proposed visual language. Complex queries with numerous basic objects are not expressible, and the extension necessary to accommodate such queries involves cartographic generalization problems.

4.1.6 Svensson and Huang [PZ91]

The spatial query language presented in [PZ91] is one of the few existing propositions of extension of a database language different from SQL. The SAL language was designed to support analysis of data stored in a statistical database system called Cantor. The motivation for using a SAL extension as a spatial query language is that previous languages have concentrated on representation, manipulation and management of spatial data, but their analytical capability has been limited.

The authors reclassify commonly used spatial operations according to the procedures followed in a stepwise analysis:

Selection and transformation : retrieve data from the database and transform them into an appropriate form (for example, scale and orientation).

Reclassification : assign new property values to spatial objects based on their initial property values (geometry or location, for instance).

Measurement : compute metric properties, ranging from simple (e.g., arc and line lengths, point-to-point distances) to complex operations (such as travel time, optimal path, and accumulated cost).

Neighborhood : extract spatial characteristics of object locations (such as slope and aspect) and interpolate information that is missing between objects.

Overlay : find spatial relationships of objects and create new classes of objects based on combination of their non-spatial properties and spatial relationships.

Statistics : compute statistical features of different classes of spatial objects over a specified area (e.g., percentage, distribution, and frequency).

Most of these operations can be performed by existing GIS, though the complexity of the process is usually very high. The query language approach is expected to reduce considerably the number of concepts needed in future spatial analysis systems, simplifying the user's tasks.

Extensions are needed in the following aspects of the SAL language to support spatial analysis: type structure; formal argument types; definition of explicit ordering in sets; definition of initial values and type conversion for set-valued objects; modules, encapsulation and persistence; recursive view definition; and language extensibility. [PZ91] presents such extensions.

The extended language, named Geo-SAL, allows spatial data types to be used in columns of relations in the same way as ordinary data types. The internal representation of spatial objects will be accessible to the user, although it will in general not be necessary to know it in order to formulate queries.

Besides point, line and polygons, the language supports partitions of polygonals called tesselations. The elements of a tesselation can be regular or irregular polygons. The raster type is a regular tesselation of a rectangular area, and it is used to represent spatial data in which the geometry of spatial objects has not been established, such as unclassified remote sensing images. The inheritance hierarchy of Geo-SAL includes the following classes: point, vector, line, polygon, square, pointset, lineset, polygonset, tesselation, regulartesselation, and raster.

Five classes of spatial operators are defined over these spatial data types:

1. Unary operators extracting geometrical data: coordinates of a point; start/end point of a line; nodal points of a line; n-esim node of a line; segments of a line; n-esim segment of a line; length of a line; closed line which is the outer boundary of a polygon; total area of a polygon; polygon that connects the ending points of a line; and direction of a vector.

2. Object transformation operators: rotate a line or polygon; create a polygonal region extended from a source spatial object; translate a source object by adding a vector V to its points; transform two rasters representing a source distribution and a scalar velocity into a raster consisting of travel times; and transform a digital elevation raster into a raster of logical values indicating visibility of the corresponding position from a horizontal position P and elevation H.

3. Binary operators computing geometrical relationships: minimum euclidian distance; arc distance along a line; and distance between two object in a given direction.

4. Binary operators testing topological relationships: disjoint; meets; equals; contains; covers; overlaps; and intersects.

5. Object operators building new objects from existing ones: set operators (union, difference and intersection) which compute new spatial objects for homogeneous arguments; and the cut operator, which operates with heterogeneous arguments producing new spatial objects if specific topological relationships hold between the objects (e.g., if the covers relationship holds between a line and a point, cut produces a set of two lines consisting of the two parts of the input line on each side of the point).

Composition of operators compute further geometric data. For instance, applying the length operator on the result of the operation *boundary(P)* computes the perimeter of polygon P. Three complete examples of analytic queries using these operators are presented in [PZ91]. One can note from these examples that the syntax for expressing complex queries is really declarative and concise. Unfortunately, it is also very complex and distant from the conventional user mental model. End-users of GIS are not expected to understand the proposed syntax.

4.1.7 Egenhofer [Ege92]

In [Ege92], Egenhofer provides several arguments showing the inappropriateness of SQL as a framework for high level GIS query languages. It is an interesting point of view, if we take in account that the author proposed different SQL extensions as a solution for this problem (see sections 4.1.1 and 4.1.10). Egenhofer notes that almost every spatial extension of SQL extends the domains of the relational calculus with spatial data types. However, this is useless unless the pertinent operations and relationships are also defined. The problem is that there is no comprehensive set of formal definitions for spatial relationships; therefore, the semantics of the relationships differ considerably among the various spatial query languages.

Another problem with SQL extensions is how to define the graphical display. It is necessary to specify which parts of the results of a query are going to be displayed graphically, and for these parts, one needs to describe how to display the data (i.e. which colors, patterns, symbols, etc). The syntax modifications for including such definitions would have a substantial impact on the normalization of the language.

The graphical display also originates problems for selection by pointing and for reusing results. The result of each SQL query is a stand-alone instruction without any reference to the previously asked queries or their results. In GIS, queries frequently refer to previous results, that is, queries are incremental. Moreover, SQL has no provision for input other than typed characters, while the input via mouse, referencing displayed objects, is a fundamental feature of GIS user interfaces.

The expressive power of SQL imposes more difficulties to its adaptation to GIS. Though SQL is relationally complete, it does not support:

- metadata queries: e.g., to show the structure of the data that is currently displayed;
- object identity: SQL is inherently value based, but a GIS object needs value independent identification because it may change its characteristics over the time, remaining, nevertheless, the same object;
- knowledge queries: involving, for instance, constraints about a given geographic entity;
- qualitative answers: SQL provides quantitative results about the values of tuples, but not results based on analysis of relationships;
- integrated retrieval and display: this refers to the graphical display problems discussed above: reuse of queries and definition of graphical presentations.

The author concludes that before syntax extensions are proposed, the semantics of the operations on the spatial data types must be defined formally. He also notes that object-oriented SQL may contribute with solutions for some of the presented problems, but definitive solutions demand the conception of new high level GIS user interfaces. These new interfaces are necessarily based on cognitive mental models applied to spatial data. In section 5 we discuss such models.

4.1.8 Boursier and Mainguenaud [BM92]

Boursier and Mainguenaud discuss, in [BM92], three different approaches for spatial query languages: extended SQL, visual languages and "hypermaps". The discussion is based on seven classes of queries that a true GIS should be able to process: non-spatial, topological, based on spatial relationships, negation, disjunction, aggregation and deduction. The proposed typology for queries is muddled, since topological and spatial relationships are not disjoint; nevertheless, the paper presents some important comments on spatial query languages.

First, it is noted that predicates have a weak expressive power if compared to operators. The more complete extensions of SQL provide spatial operators in the *select* clause, and not only predicates in the *where* clause. In such languages, a query is defined either by a combination of SQL statements ("separate approach") or by allowing an SQL statement or an operator to be an operand of an operator ("global approach").

The main advantages of SQL extensions are: SQL is the *de facto* standard database query language; it can handle alphanumerical queries in an optimized way; its coupling with classical programming languages has been studied. The major drawbacks include the weak expressive power of the relational model, the optimization of queries involving alphanumerical and geometric data, and the difficulty of expressing network-oriented queries (e.g., those involving constraints on nodes and edges of a graph).

A visual query can be defined either specifying elementary operations step by step ("imperative approach") or stating the properties to be verified by the final result ("declarative approach"). According to [BM92], in both manners the evaluation of a query is a two stage process: first convert the drawing into a formal expression and second convert this expression into DBMS understandable orders. We could envisage, however, a direct conversion of the visual query to the formal language of the DBMS as a one stage process.

Two main advantages are seen in visual languages. First, they are more natural for the end-user (i.e., they are closer to the user mental model). As we already mentioned, spatial concepts are better understood through graphical representations. The second advantage is that visual languages allow easy combination of operations and reuse of results for formulation of further queries. There are, however, major drawbacks. [BM92] cite the lack of normalization and the complexity for formulation of non-trivial queries (specially involving negation). The authors also analyze the use of hypermedia as a GIS query language. We will omit this discussion because hypermedia can not be considered a query language, since all possible queries must be defined *a priori*. Hypermedia systems are more suitable for the development of browsing tools.

4.1.9 Arikawa, Kawakita and Kambayashi [AKK94]

[AKK94] introduces *dynamic maps*, a new style of maps composed by queries and visualization methods. The dynamic feature of these maps is obtained through: (a) immediate reflection of database updates, (b) free selection of desired data by generic queries and (c) adaptation of the appearance of the map to changes on user's purposes, to results of operations, and to the limitation of display devices.

The first two points are already satisfied by conventional database interfaces. The difficulty of selecting a proper visualization method motivates dynamic maps to offer to the user various visual interfaces.

The principles of dynamic maps, although not yet formalized, consider views on databases and on interactive graphics as key aspects to provide the three characteristics cited above. Database views are virtual databases defined by queries to real databases while computer graphics views are defined by visualization procedures to create visual objects on display screens. Query languages are well-known tools for selection information from databases, providing the framework for database views. However, query languages are not appropriate for controlling the presentation of data on displays. The paper proposes a method for this task.

To create a map of their interests, the users have (1) to assign importance levels to classes of data selected by their queries, and (2) to select visualization methods for the selected data according to the importance levels and to the limitation of displays. The proposed method establishes a tradeoff point between the user's requirement of data and the limitation of the display, based on constraint solvers and relaxation procedures.

One important problem involved is to decide the limitation of a collection of display objects in a window. The idea is to offer the user graphical user interfaces for observing the results of visualizing objects and for adjusting their importance levels. This process is also influenced by cartographic restrictions, such as text labels must not overlap. The author's propose a display complexity function to help the selection of visualization methods. The basic idea is not that users define their visualization methods; rather, they should be able to choose freely their intended visualization method from some algorithms prepared in the systems.

There are two distinct types of objects involved on dynamic maps: conceptual objects and display objects. The former are database entities retrieved by a query (or view). The latter are visualizations of the formers. Multiple display objects can refer to the same conceptual object, in different visualization methods. Changes on conceptual objects are propagated to the corresponding display objects.

A visual layer is defined by a pair: database view and visualization method. The latter specifies properties of display objects (pattern, color, behavior). A dynamic map is composed of multiple visual layers. Since the database view component of each visual layer is a collection of conceptual objects, the dynamic map is automatically updated when the database changes. The method for constructing dynamic maps also defines common conditions that are applied to all database views. For instance, the time and area can be defined by the user, and all queries are referred to these common conditions.

4.1.10 Egenhofer [Ege94]

Egenhofer points out some requirements for spatial query languages that are not covered by conventional systems: a "spatial" abstract data type with corresponding operations and relationships; graphical display of results; combination of one query result with previous results; display of contextual (not explicitly requested) information; control mechanisms to check the contents of a drawing; selection by pointing for formulation of new queries; varying graphical presentation of spatial objects; descriptive legend; labeling of objects; and scale changing.

In the context of spatial data handling, a query language has to provide solutions not only to the retrieval of data, but also to the definition of its presentation. [Ege94] separates GIS queries in three categories: queries exclusively about spatial properties; queries about nonspatial properties; and queries that combine spatial and nonspatial properties. SQL is a suitable language for the second category of queries, and the paper suggests an extension of this language for dealing with the other two categories of queries.

The premise of the design of this SQL extension was to retain the concepts of the host language. The following concepts of standard SQL were specifically regarded: the structure of the *select-from-where* clause stays untouched; every query result is a relation; predicates in the *where* clause are formulated upon attributes. The proposed language does not intend to improve the drawbacks of SQL as a spatial language (see section 4.1.7 for a discussion of these problems).

The primary goal of the interaction with graphical renderings in a GIS is to make dynamic changes rather than to build static products. The integration of a full display description into the query language would make each user query unnecessarily complex. Three types of instructions are therefore distinguished: 1) the actual user query, specifying the retrieval of the set of data to be displayed; 2) Additional queries, called display queries, necessary to separate query results into more detailed sets, each to be displayed in an individual format; and 3) the actual display description specifying how to render the resulting data. The author proposes the separation of these instructions in two languages: a retrieval language and a display language. The user sets the display environment through the latter language and asks queries with the former. Query and display instructions are combined in a nonprocedural way: users describe the display style and formulate the query, and the system finds the most effective way for integrating and executing the queries.

The spatial SQL proposed provides a higher-level abstraction of spatial data and extends the relational domains with zero, one, two and three dimensional spatial objects. These domains are generalized to a dimension-independent domain called *spatial*. Unary spatial operations access a spatial property of a tuple, and it can be considered a function upon a spatial attribute. Examples of unary operations are: dimension, boundary, interior, and arithmetic operations like length, area, volume, and perimeter. Binary spatial operations calculate a value among two tuples os spatial relations. Two binary operations are allowed: distance (integer) and direction (angle). Besides, aggregate functions, such as minimum and average, can be formulated upon distances and directions.

Binary spatial relationships conform with traditional binary relationships and result in a boolean value. Hence, they can be immediately applied as predicates in the *where* clause of

SQL. Binary topological relationships are based upon the set intersections of the boundaries and interiors of the two targets. For instance: equal, disjoint, meet, overlap, inside/contains and covers/covered by.

Spatial attributes are used equivalently to conventional attributes, either in the *select* clause as a projection, or as a predicate in the *where* clause. The user can define several geometries for a single object, for instance, for representation at different scales and levels of detail. The *pick* qualifier allows users to formulate queries with reference to spatial objects visible on a screen. It is incorporated into the language as a predicate and can qualify each spatial attribute in the *where*— clause. Ambiguities are reduced since the user points to a location and specifies in the query which kind of object to select.

In the display language, a display environment handles the information about how to display query results. During query processing this information is integrated with the user query so that the query result is rendered according to the display description. Unless the user changes the environment through the display language commands, it continues to produce a map with the same style with each query. The types of graphic specification are: display mode; graphical presentation (hardware dependent colors, patterns and symbols); scale and window (delimiting the area to be displayed); context (portions of spatial relations to be merged with the queries results); and content (a single query with which the drawing currently visible would be produced).

In order to combine several query results in a single drawing, the selection of the appropriate display mode is necessary. The display modes allow: to display alphanumeric data in tabular form; to start a new drawing; to add the result onto an existing drawing; to erase the result from a drawing; to determine the common objects on the display and in the query result; to emphasize the result in the drawing.

4.2 Remarks on GIS Query Languages

Many features of GIS query languages were adapted from picture query languages. [CF81] reviews such languages and introduces an example-based pictorial query language for an integrated relational system binded to an image analysis system.

Cooperative environments for querying geographic databases are a recent development in GIS query languages. [CHF93] discusses cooperativeness of GIS query languages, and suggests desired features of a GIS interface:

- facilities for visualization of the geometry of objects;
- graphical formulation of queries based on previously displayed results;
- combination of previous queries and interpretation of the presented results;
- different presentation formats (color, symbology, patterns);
- definition of visual context for queries;
- scale changing facilities.

Object-oriented concepts have greatly improved spatial SQL extensions. [Lor91], for instance, presents a language that supports complex object data, abstract data types, methods, object identifiers, and inverse relationships, maintaining, however, the compatibility with the relational model. The advantages of such textual interfaces include the support to input of complex actions, not easily expressed with graphics, and facilities such as command scripts, command histories and logging. Their major drawback is the semantic distance from the user mental model. Textual languages do not allow natural expression of spatial concepts.

The main advantages of visual languages are: they are more natural to the end-user (i.e., they are closer to the user mental model); and they allow easy combination of operations and reuse of results for formulation of further queries [CM91]. The drawbacks include the lack of normalization and the complexity for formulation of non-trivial queries, specially involving negation [BM92].

Although visual interfaces are gaining importance, current GIS query languages use a hybrid approach of textual formulation and visual presentation of queries results. The expressive power of visual languages must be increased before pure visual GIS query languages can be implemented.

5 Spatial Cognition and Human Factors in GIS

The design of user interfaces for software systems relies on basic assumptions about their potential users. The set of assumptions about the users' behavior and their manner of thinking is represented through the *user mental model* (*user model*, for short). Research on human factors in GIS focuses on constructing a GIS user model. Based on this model, it defines the interface representation model and the external language which maps users concepts into that intermediate model.

Problems for the specification of the user model range from the definition of basic spatial concepts, such as "region" [Gut92] or "neighborhood" [Gol92], to the proposal of adequate metaphors [Kuh91] for representing these concepts.

In this text we will remain as close as possible to the computer science view of the question. It must be noted, however, that the research in this area is inherently interdisciplinary, involving several areas such as psychology, computer science and ergonometry. The reader interested on other views of this area applied to GIS will find a nice introduction in [MSH93].

In the remainder of this section we present some results showing the diversity of subjects studied in this research area, ending with a brief comment on the difficulties and the problems that remain unsolved.

5.1 Review of Works on Human Factors in GIS Interfaces

At a first glance, the results presented in this section seem to have little in common. Some of them treat directly the subject of user interface design; others give little attention to the user interface, concentrating on spatial cognition. In spite of these diverging emphases, we see a fundamental objective in all of them: the understanding of the human perception of the space. This is the very first step to build a GIS user mental model, which, as we already mentioned, should guide the design of the user interface for these systems.

5.1.1 Kuhn [Kuh91]

[Kuh91] distinguishes three aspects of a GIS user interface: displays, maps and views. Displays are graphical presentations of the geographical space in the screen; maps are static representations of this space; and views are visual fields containing human perceptions of a given situation.

A metaphor is a (mathematical) mapping from a source domain to a target domain, structuring the latter. The choice of the metaphor in GIS defines: what concepts the user has to deal with; how the work is divided between system and user; and what kind of communication is adopted. The invariant in the mapping represented by a metaphor is the reasoning pattern based on image-schemas (e.g., container, path, link, near-far, part-whole, center-periphery). These image-schemas are appropriate for GIS interfaces since they are inherently spatial, and in particular, topological.

Kuhn shows that the *display are maps* metaphor is not efficient for organizing GIS operations, and proposes the *displays are views* metaphor. This new metaphor explores features of human visualization in questions of resolution and scale changing. Cognitive science has already stated that humans perceive, conceptualize and deal with the world in multiple levels of detail, and this is the proposed approach for the GIS user interface.

The current GIS metaphor (displays are maps) has some major drawbacks: maps may not be understood well enough to serve as a source domain; they do not provide directives for operations beyond the display one; they tend to hide uncertainty in the data; they are a two-dimensional representation of a three- or four-dimensional (considering the time dimension) reality.

The storage of data in GIS is not based on the map metaphor anymore, and Kuhn suggests that the user interface should follow the same path. It is important to remember, however, that the current metaphor has some strengths: it inherits from cartography the objective of graphic excellence, with useful conventions and symbolisms. The problem is that it also inherits the drawbacks, such as handling multiple resolutions through series of scales and difficulties for controlling cartographic generalization.

The "displays are views" metaphor presents solutions for these problems, through anology with the human visualization system, which involves categorization of what is seen and is dependent on the spatial point of view of the observer. The basic image-schema involved in the view is the "container" schema: the visual field is a limited space with an interior and an exterior. An object is either in or out the visual field. There is also a center of attention in the view and a surrounding region (center-periphery image-schema). Finally the visual field is structured by an interaction of the part-whole with the near-far schema, that is, we perceive objects as configurations of parts, forming wholes, and getting certain parts into view implies moving nearer or farther. This connection is the essence of scale changes and zooming operations.

Kuhn concludes that adopting the displays are views metaphor implies in several changes: the user is involved on a dynamic process of viewing rather than observing static maps;

the user's point of view has a key role in defining display contents; the notion of zooming goes beyond magnification by relating different concepts to different scales. The display are maps metaphor is adequate for static (paper) map production, while the display are views metaphor allows to relax some map constraints, such as those on minimal dimensions and separations, which make automatic map generalization a hard problem.

5.1.2 Couclelis [Cou92]

[Cou92] analyses the two main views of the geographical space: object and field. Couclelis argues that none of them is the most apropriate, since human cognition use both views at different scales and for different purposes.

The *object* view is based on points, lines and polygons representing geographic entities of the real world. Those geometric objects have the properties of real world objects: they are discrete and have independent existence; they have (relatively) permanent identities; they have attributes and shapes; and they can be manipulated (counted, moved, colored, rotated, for instance). Most points, lines and polygons that exist in the real world originate from human artifacts of two main categories: engineering works and administrative or property boundaries.

The object view of the geographic space focuses on the spatial (topological, projective, metric) relations that may hold among objects. It misses, however, important human geography relations that characterize territorial behavior. These relationships lack some of the basic properties of objects: they are defined by social relations rather than by intrinsic object properties; their changing is determined by social transformations, not by physical movements; they do not partition the space, although they may share it; they are context- and place-specific. Thus, the object view forces the geographic world into a uniform mold of geometric objects

The *field* view takes an opposite approach: it is based on maximal ignorance of the nature of things in the real world. Any geographical phenomenom is represented by an array of pixels. Grouping of pixels in particular configurations, or sharing particular attributes, can be identified with a specific "feature" on the Earth's surface. This nature of features is in contrast with the strong individuality of objects in the object view. Nevertheless, the geographical world is understood by means of these two opposite views. Two basic factors influence geographic cognition: scale and purpose (or human intentionality). Each combination of these factors tend to a different (object or field) perspective of the geographical reality.

5.1.3 Guttenberg [Gut92]

In [Gut92], Guttenberg discusses the division of space (and time) into regions. He starts by recalling a previous definition of the term "region": *an entity for purposes of though, created be the selection of certain factors that are relevant to an area of interest and by disregard of all features that are considered irrelevant*. According to this definitions, regions are behavioral phenomena rather than natural phenomena.

The author suggests that human behavior originates four types of regions:

1. Referential regions: regionalization takes the form of reference to (description and analysis of) natural and cultural variations in space. Example: french speak areas.

2. Optative regions: originated by social visions or aspirations. Example: perfect political division of a country.

3. Appraisive regions: regionalization that characterizes the quality of life. Example: pollution areas.

4. Prescriptive regions: the division of space is intended to take actions in the face of perceived social and environmental ills. Example: wilderness areas.

The conclusion from this typology of regions is that the term may refer to fixed, invariant geographic entities or to dynamic, movable socio-economic systems. Therefore, regions are mental constructions which appear and disappear, imitating the adaptive behavior of humans. Besides, rather than beeing sterile objects, regions always generate related regions. Ambiguity is inherent to the term: it refers to both an absolute space and to a spatially referenced (social, economical, cultural, geographical) system. This ambiguity must be foreseen in the design of a GIS user interface system.

5.1.4 Lindholm and Sarjakoski [LS92]

Lindholm and Sarjakoski define language as a set of signs (vocabulary) and a set of rules (grammar) governing the use of the signs. In GIS, the user interface acts as an intermediary language between the formal, abstract database language and the informal, empiric user language. The vocabulary of database language is the set of stored data items, and the grammar is the conceptual schema coded as a data dictionary. The user interface must provide an input language to transmit user's commands to the DBMS, and an output language for presenting the database contents to the user. In [LS92], the authors focus on maps as the output language. They note, however, that although the most common way to express georeferenced data is through maps, other languages (e.g., multimedia presentations) can be used.

A communication system in GIS may be seen as sets of informations coded in some language, and the process of translation between these languages. The input and output languages act as common ground for the user and the DBMS, whose message sending and receiving methods are not otherwise compatible. Figure 14 present this communication system.

The study of human perception and action is mostly of concern of ergonomics. The design of query process is related to database theory. The process of mapping information from the database to the GIS output language has traditionally been the subject of cartography. In cartographic communication, three main sources of error can be detected: incorrect encoding (erroneous use of map symbols); incorrect decoding (the reader does not understand the symbology); and language mismatch (the cartographic language is unsuitable for expressing the desired phenomenon). Moreover, cartographic communication is unidirectional. In a GIS environment, the user is allowed to change the contents or the looks of the output, and the communication becomes circular (figure 14).

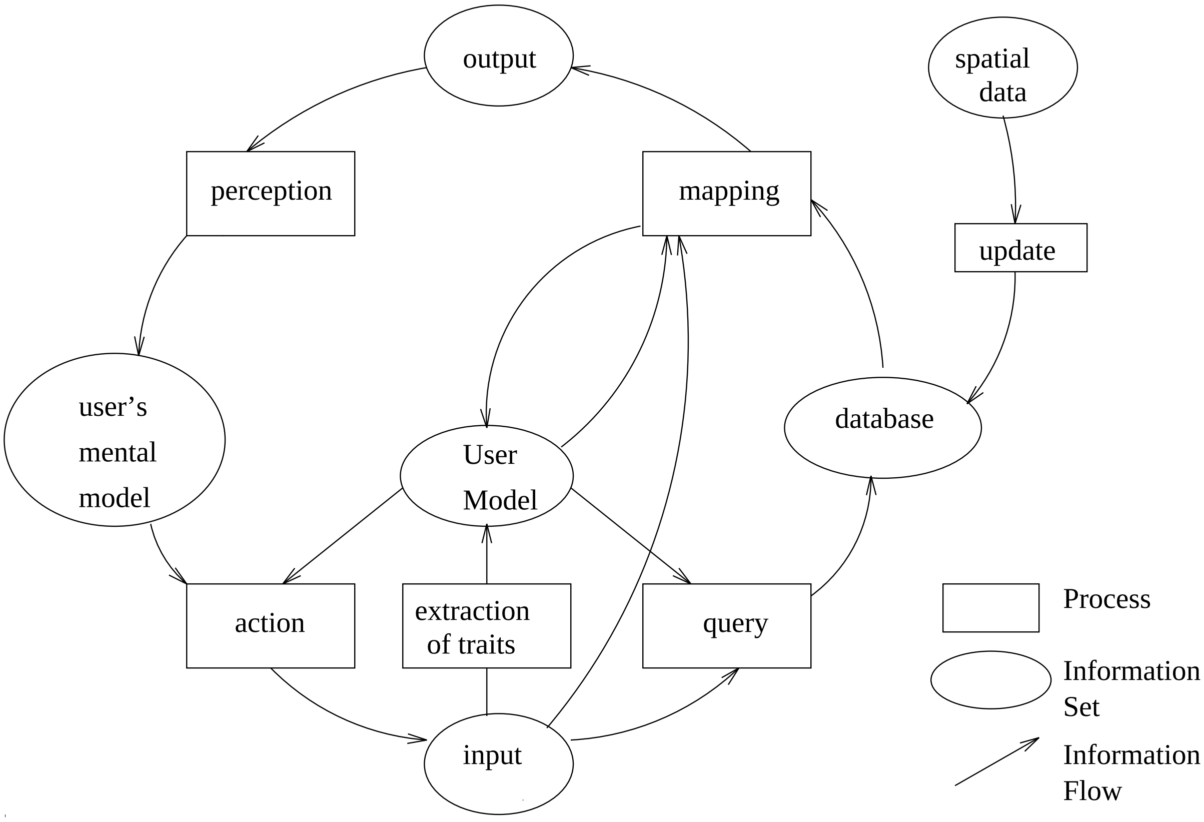

Figure 14: Communication system in GIS, adapted from [LS92]

The communication process may be decomposed in three aspects: syntatic, semantic and pragmatic. The syntatic information of a message is defined in relation to the frequency distribution of all possible messages: the least frequent item has the highest information content. The semantic content of a message is defined as how many different states of the universe distinguishable in the language it excludes logically. Pragmatic information is associated with psychologic questions, such as value (or usefulness) of the message and sign recognition and interpretation.

The main interest in user model research is to develop systems that can adapt themselves to different users at run time. The most important function of user models is to predict the user's actions and preferences and to change functionality accordingly. Therefore, user models aim on controlling pragmatic informations. Building a user model is useful for economy of interaction, for improving user acceptability and efficiency of use of the target system. Nevertheless, it has its weak points too: people may find it unpleasant to be controlled by a computer; the user model may be too mechanistic and discard information that is important for the user; it may present a risk to privacy.

One usual way of constructing the user model is through the observation of user behavior during a work session, and its association to one or more *stereotypes*, which are attribute-value pairs describing characteristics of the user (e.g., age and experience). Some attributes may trigger the activation of a stereotype, which can be organized in a hierarchy, with the canonical user on top. The stereotype of a particular user can be built by combining and modifying different stereotypes in the hierarchy.

Three pragmatic factors should be considered on the definition of a GIS user mental model: the information the user wants, to restrict the universe of possible states of the real world; the information the user already has, to deduce the context the user is work in; and the user's ability to infer deeper information from a given piece of surface information. These factors determines which data is presented to the user, and help in the definition of stereotypes.

[LS92] concludes that GIS metaphors should be associated to user stereotypes, since it is difficult to find one metaphor to suit all the different tasks in GIS. Each metaphor presupposes a certain stereotypical user who is familiar with the concepts and operations of the real world phenomenon from which the metaphor is derived. In an ideal interface, each user would be able to communicate with the system in his "native" language, that is, with the concepts most familiar to him.

5.1.5 Gould [Gou93]

Gould's approach to GIS user interface is based on three main impediments to GIS use. First, there is a training problem: today's GIS are difficult to learn. Often the major part of a training session must be devoted to mastering the command language and the structure of the system. Second, there is a task mapping difficulty: current GIS are designed around suites of commands comprising classes of similar data manipulations. The user has to map tasks to the commands available. Both novice and expert users have problems with this mapping. The former are not able to deal with large number of commands while the latter are handicapped by expressiveness limitations of today's languages. Finally, there is a customization impediment: only rudimentary customization is allowed in existing systems. This is a serious problem since most GIS users in the near future will be neither experts nor programmers; besides, the addition of customization increases the training overhead.

A solution for these impediments can be reached only if the user interface design is completed before writing the GIS software. Without this primary focus, developers may be providing solutions to the wrong problems. A key aspect of the user interface design is user requirements assessment. The goal is to discover what tasks the GIS user performs and how the interface system can simplify these tasks. To reach this objective the designer must transcend the software engineering word to deal with cognitive and management sciences.

Adopting a user-driven GIS design, Gould proposes two approaches to the interface design: the data manipulation view and the abstract world view. They are neither mutually exclusive nor sufficient in themselves, but focusing on one over the other may be best for certain GIS applications. The data manipulation view seems best for quantitative data management use of the GIS, and this is by far the major GIS use type. The abstract world view is better for in-depth scientific analysis. This view may give the scientist the opportunity to consider new solutions for key human and environmental problems which have a geographic component.

The data manipulation view is based on data manipulation tools, which have been considered the building blocks of GIS. The focus here is on the internal mechanics of the system to provide results as quickly and as accurately as possible. Data manipulation tools should exploit the capability of the current computing system, such as multiple windows (scales and views), graphics (colors, shapes), multitasking, and networking. This view is widely accepted and promoted by major GIS vendors (e.g., ARC/INFO). The toolbox metaphor involved in this view is useful to users that are also programmers, because they are already choose and combine commands and functions from standard libraries in order to perform desired operations.

Interfaces based on the data manipulation view should provide only indirect access to internal data manipulation tools. The problem in this approach is how the user controls the operations externally. In general, external tools are used to this purpose. Optimally designed interfaces allow the user to overlook these external tools, focusing on the geographic problem-solving. In other words, users should be able to interact in terms of geographic problems.

Many external tools take advantage of visual and direct manipulation concepts towards this objective. Using these concepts reduces learning time and increases the understanding of the system due to the elimination of memorization of commands and their syntax. Gould notes that interesting possibilities also remain for linguistic (textual) tools. In this case there are two main design approaches: top-down, extending formal query languages, and bottom-up departing from concepts of human languages and building spatial subsets of these natural languages.

The abstract world view of GIS user interface is a direct connection of the user to a virtual existence “inside” the computer. This subject has only recently drawn attention from GIS researchers, mostly in the area of metaphors. The emphasis here is more in the realm of cognitive modeling than of data modeling. The goal is to match concepts of the user interface to those which people use naturally. The main difference between data modeling and cognitive modeling is that the latter does not pre-define the user’s mental model or world view, but rather attempts to discover and exploit it.

An abstract world is best viewed as a “world metaphor”, super-ordinate to organizing metaphors. The desktop metaphor is a well-known example of such an organizing metaphor for the business world. In analogy to this, it is suggested that an organizing metaphor for GIS might be a geographic analyst’s workbench, and that the world metaphor may simply be geography. The advantage of this metaphor is that it provides context, i.e., it constraints the possible operations within GIS to those within certain scale. The assumption is that the user needs to focus on manageable chunks of the Earth’s surface.

The primary aid for designing a user interface based on abstract worlds is metaphor. The central question is the view of the world that is to be offered to the GIS user. Furthermore, it is necessary to establish the level of abstraction at which the view is represented. Each abstraction provides a certain “sense of place” within the GIS, in relation to the user’s real world. This sense of place is what user views of the spatial database should support. Gould notes that some users want and expect a two-dimensional map, but others distinctly may not. Design the user interface based on the abstract world, rather than on data manipulation tools, provides the opportunity to meet the user’s true needs.

5.1.6 Egenhofer and Herring [EH93]

[EH93] investigates the relationship between query languages and user interfaces in GIS, under the point of view of users' requirements. The idea is to classify query languages based on the kind of tasks users perform, in opposition to the traditional taxonomy based on implementation aspects (e.g., formal and natural languages, menu based, graphical).

The basic assumption is that a GIS query language is basically a database query language. The authors note that much of the work on spatial query languages has been influenced by trends in the database community. In this context, the problems for interface design stem from differences between the conceptual models human employ when they think about a particular task and the data model of a DBMS. Query languages are delivered with the system, instead of being designed with the user's conceptual model in mind. This problem is not serious in relational systems employed on business, since the table metaphor matches the user model of manipulating business data. In GIS, however, this model is not efficient.

The design of a language directed to the user conceptual model has to deal with two conflicting requirements: the language should be sufficiently user-oriented to allow unambiguous interpretations of data manipulations; on the other hand, it should be sufficiently concise and powerful for the user to access and manipulate the data to the full extent that the data model will allow. If users are allowed to violate the intended use of the data, they will often incorporate hidden meanings into the data; if they are restricted to the database model, much of the functionality specific to a particular GIS application may be lost. This problem is increased if data manipulation includes updates. In [EH93] the discussion is restricted to retrieval of data.

Users specify the information that should be retrieved from the database through queries. A query contains clues indicating the particular data of interest. These clues may be some particular values that the objects of interest should have, or some constraints that the desired object should fulfill with respect to other objects (e.g., topological relationships). The user interface must isolate the user from the internal working of the database. Thus, if a query language is to operate at the surface of the application, it must be able to support, to some degree, the meanings imbued to the data by the application. Failure to control the semantic range of the query language can have severe data quality and integrity impacts. For instance, a road may be sent across a river without a bridge beneath it. To deal with real world meanings and application semantics, it is necessary for a spatial query language to support spatial concepts, i.e. users must have access to the ideas they usually use when they think about the space. These spatial concepts may range from simple concepts, such as "close" and "adjacent", to complex application-specific concepts, such as networks as collections of edges connecting pairs of nodes in two-dimensional space.

Traditional database query languages have been used for spatial databases. While these experiments showed the feasibility of modeling spatial data as relational tables, they also revealed the lack of functionality of these languages to treat spatial data at a higher level of abstraction. Fundamental differences exist between management of spatial and conventional data. For example, conventional DBMS organize information in abstract name spaces and allow users to refer to the data through their values, while spatial DBMS organize information in two-dimensional geometric spaces and allow users to exploit spatial relationships to retrieve data of interest. Furthermore, tabular concepts popular in business representations play a minor role on spatial databases, where data is specifically associated with geometry and map-like presentations. Indeed, the simplest use of spatial data requests information in a "boundary" area, that is, an "inside boundary" analysis must be done before data can be retrieved. The use of relative position, such as "adjacent" is also common is GIS queries. Since it is difficult to store all the spatial relationships, the GIS query language relies on implicit data and spatial analysis, which does not occur in conventional database languages.

Egenhofer and Herring cite spatial relations and spatial concepts as necessary features for spatial query languages. Spatial relations based on topology, metrics, and order have been identified as a crucial spatial criteria in spatial query languages. Some of their semantics, however, are still unclear and lack formal definitions. The spatial concepts that underlie the spatial relations may vary depending on the user's task. For instance, "closest" may be based on euclidian distance or on shortest path. Even for the same application, different users may have different concepts. For example, "adjacent parcels" on a cadastral application may have different meanings, depending upon each local law. Within the language, it is also important that the users can specify what parts of an object to retrieve and how the query results should be presented. Moreover, the query language should support the mixing of two retrieval modes, based on the values of attributes and on the location of objects.

Four types of spatial query language implementations have been proposed: command languages, natural languages, semi-visual languages and direct-manipulation languages. The most common implementation is through command languages in which users express queries in terms of instructions that reflect the set of operations implemented in the GIS. Typically this type of language is extensible, providing inclusion of new features, but making it difficult to understand how the individual commands relate to each other. SQL dialects are another approach to command languages, but they have been shown to be cumbersome to use and to result in complex query statements.

Natural languages reduce the need for user training, but although humans can usually solve the inherent ambiguity, it is too difficult for a system. Semi-visual languages integrate an SQL-like query language into a graphical environment including windows, icons, mice, pointers, pull-down menus, and data flow diagrams of queries. While the environment remove the need to remember and type commands, it still preserve the command language structure so that users must think in terms of constructing sentences. Direct-manipulation languages describe spatial constraints graphically. It is a natural way of expressing spatial concepts, but it also presents problems. For instance, graphical presentations overdetermine spatial constraints; it is impossible to describe only a single spatial constraint in a picture, since it contains information about the objects' relative sizes, their shapes, topological relations, etc. Furthermore, they apply only to very simple spatial queries since it is very difficult to sketch disjunctions and negations.

5.1.7 Wood [Woo93]

Wood presents particularities of map interaction, in comparison to the general definition of interaction as a reciprocal action of influence comprising both mental processes (thinking) and physical actions (speech, gesture). Furthermore, conventional (or real) paper maps have different interactions from GIS (virtual) maps. The former are directly viewable and have a permanent tangible reality; the latter are subdivided into three categories: directly viewable without permanent tangible reality (e.g., map on screen); not directly viewable with permanent tangible reality (e.g., map in CD); not directly viewable and no permanent tangible reality (e.g., map in a database).

The realms of map interaction are map making and map use. The first is a cartographic activity, and it has been object of intense research for a long time. The paper focus on the second interaction, map use, which can be analyzed as a visuo-cognitive dialogue or as a set of scenarios.

The visuo-cognitive dialogue analysis considers a user interacting visually and mentally, in silent dialogue, with a single map. The steps of the process are: reading, analysis and interpretation. This process plays an important part in all map use activities.

The scenarios analysis is used in multi-map environments. Examples of scenarios with different levels of physical interaction during map use are: (a) multiple map use, where the mental dialogue may be assisted by physical interactions, such as folding a map sheet; (b) user map-making indirect, where a separate cartographer generates special maps; (c) user map-making direct, where the user alone generates computer-assisted maps; and (d) user as interactive explorer, where advanced spatial systems provide multi-media tools allowing virtual time- and space-travel. In reality, combination of these and other scenarios would emerge.

Wood notes that making and using were largely exclusive activities, but are converging with the support of GIS. This trend issues two important questions: can new user-makers take on the responsibility of this cartographic role? Do they have the correct education and training or must the system embody some cartographic expertise? These questions are discussed in [Woo93] under three points of view (map-reading process, map-use tasks, and effectivity of maps), with emphasis on the map-reading process. Indeed, a taxonomy of map-use tasks presents a considerable challenge while effectivity of maps is task dependent.

Wood proposes measurement and visualization as core categories of map-use tasks. The former may be aided by visualization but encompassed external assistance such as measuring scales and calculators. Without such support the tasks would be limited to the perceptual-cognitive procedures: visualization, navigation and interpretation.

An effective map is one which fulfills its required task. Conditions for effectiveness include map projection, scale, content, generalization and design. Conflicting user's needs also counts: novices prefer simple maps while experts benefit from rich visual fields. Guidelines for design and generalization remain largely heuristic and have resisted attempts to total automation. Aesthetics may not be a priority for map design; nevertheless, clarity and legibility are important for user acceptance.

Map-reading can be described as a form of human information processing. The system model describes this activity on each stage of operation, from stimulus to response; the processing model focuses on what happens at each stage. The system model describes the human visual information processing, initiating from the visual receptor system, composed of different types of photo-receptors, sensitive to distinct input (light intensity, wavelength). The limitations of these sensors define minimum dimensions such as lines thickness and separation between symbols. The core of the system is the short-term memory, controlling the process of information retrieval from the long-term memory.

The processing of maps attempts to track the stages of increasing awareness during map-reading and how elements of the image are detected, progressively recognized through links with the long-term memory, and combined into larger meaningful symbol groups (synthesised visual memory). The knowledge acquired from education and experience is stored in the long-term memory, playing a vital role on this processing model. This memory seems to be structured in previously established knowledge schemas, with general themes at higher levels linked to more specific elements lower down.

This organization enables to processing modes. In the bottom-up (data driven) processing, the visual system responds to primitive graphic marks on the map, and only then does meaning emerge through reference to long-term memory. In the top-down (concept driven) processing, the percipient is assumed to approach the image as a map, and thus has a ready selection of relevant hypothetical schemas to test against it. In practice, some combination of the two occurs.

The interactive map display will demand awareness of the content of available databases, besides map-reading and manipulative device skill. Mental visualization questions, such as "add the contour layer to this map", will introduce a pre-perceptual element to the map-reading process, bringing it closer to the processing and use of natural language.

5.2 Remarks on Human Factors in GIS Interfaces

The absence of an adequate user interface for GIS can be explained by the relative little knowledge of the real needs of the GIS users' community. Rhind, Openshaw and Green point out two factors that complicate most *user need* studies: the rapidity of technical change and the growth of user appreciation of what can be achieved by a computer system. In [ROG88] these authors illustrate the nature of needs of GIS users with a series of simple, generic questions:

1. What is at location...?
2. What is adjacent to...?
3. Where is... true/to be found?
4. What has changed since... and where has this occurred?
5. What spatial pattern(s) exist(s) and where are the anomalies?
6. What if...?

It should be noted that functionality is important for the user, but performance, reliability and ease of use are important as well. From the diversity of tasks and the heterogeneity of users, we can conclude that no single interface model is suitable for GIS. Therefore, GIS user interfaces should provide multiple interaction paradigms, and a variety of access paths to the same information, in order to satisfy a considerable number of users.

Although many problems remain open, researchers have already proposed adequate interaction mechanisms for supporting different users point of view. There is now a demand for integration of these tools into a single general purpose GIS user interface. Moreover, there is an evident lack of mechanisms treating temporal cognition and suggesting temporal metaphors. [Mon92] suggests cartographic animation for understanding of correlation between geographic variables. The same approach could be the startpoint of new metaphors for temporal representation in GIS. [Peu94] presents a comprehensive discussion about the importance of time in geographical systems.

6 Conclusions

GIS are a new class of software tools characterized by the manipulation of georeferenced data. An increasing number of applications is demanding GIS services and motivating large research efforts in this area. In spite of the great progress reached in the last years, there are still many problems lacking proper solutions in the GIS context. The user interface is one of these problems.

This paper surveyed some of the main contributions for understanding and solving this problem. The main difficulties for the development of user interfaces for GIS were classified in three main areas: architectural aspects, external languages, and human factors. The analysis of existing approaches in each of these areas led to the following considerations:

- Generic solutions for GIS user interfaces are not foreseen in a near future; even the existing ad hoc solutions are far from ideal.

- Existing architectures for GIS user interfaces fail to define at least one of the following aspects: integration of the interface to the GIS; functionality and interoperability of the main modules; intermediate representation model and mapping to GIS model; division of tasks between user interface and GIS.

- Although visual interfaces are gaining importance, current GIS query languages use a hybrid approach of textual formulation and visual presentation of queries results. The expressive power of visual languages must be increased before pure visual GIS query languages can be implemented.

- Great progress has been reached in GIS users' models and human spatial cognition. The challenge now is for integration of different mental models in a coherent interface tool.

- Practically all existing purposes for GIS interfaces deal with representation of and user interaction with space. Although the spatial aspect is inherent to georeferenced data, there is another aspect of these data that is fundamental for the understanding of geographic phenomena: time. GIS user interfaces have not yet taken this into account. Thus, it is urgently necessary to provide solutions for the representation of temporal aspects in GIS. [Peu94] discusses some of the complex problems involved in this subject.

We are now applying these observations on the design of a genericũser interface for GIS. The architecture of the interface is basically that presented in [OM95]. We are currently working on the specification of a external language based on visual and direct manipulation paradigms.

References

[ABC+91] J. Antenucci, K. Brown, P. Croswell, M. Kevany, and H. Archer. *Geographic Information Systems - a guide to the technology*, chapter 3 - Applications, pages 34–64. Van Nostrand Reinhold, 1991.

[AKK94] M. Arikawa, H. Kawakita, and Y. Kambayashi. Dynamic Maps as Composite Views of Varied Geographic Database Servers. In *Proceedings of the International Conference on Applications of Databases*, 1994.

[AYA+92] D. Abel, S. Yap, R. Ackland, M. Cameron, D. Smith, and G. Walker. Environmental decision support system project: an exploration of alternative architectures for geographical information systems. *International Journal of Geographical Information Systems*, 6(3):193–204, 1992.

[BM92] P Boursier and M Mainguenaud. Spatial Query Languages: Extended SQL vs Visual Languages vs Hypermaps . In *Proc 5th International Symposium on Spatial Data Handling*, pages 249–259, 1992. Volume 1.

[Car89] J. Carter. *Fundamentals of Geographic Information Systems: A Compendium*, chapter On Defining the Geographic Information System, pages 3–8. American Society for Photogrammetry and Remote Sensing, 1989.

[CF81] N. Chang and K. Fu. Picture Query Languages for Pictorial Data-Base Systems. *Computer*, 14(11):23–33, 1981.

[CFS+94] G. Câmara, U. Freitas, R. Souza, M. Casanova, A. Hemerly, and C. Medeiros. A model to cultivate objects and manipulate fields. In *Proceedings of the 2nd ACM Workshop on Advances in GIS*, pages 30–37, Maryland, USA, December 1994.

[CHF93] M. A. Casanova, A. Hemerly, and A. L. Furtado. Cooperative Environments For Geographic Databases: A Prescriptive Analysis. In *Proceedings of the 8th Brasilian Symposium on Databases*, pages 266–279, 1993.

[CM91] D. Calcinelli and M. Maingenaud. The Management of the Ambiguities in a Graphical Query Language for GIS. In *Proc. 2nd Symposium on Spatial Database Systems*, pages 99–118. Spring Verlag Lecture Notes in Computer Science 525, 1991.

[Cou92] H. Couclelis. People Manipulate Objects (but Cultivate Fields): Beyond the Raster-Vector Debate in GIS. In *Proc International Conference on GIS - From Space to Territory: Theories and Methods of Spatial Reasoning*, Springer Verlag Lecture Notes in Computer Science 639, pages 65–77, 1992.

[EF88a] M. Egenhofer and A. Frank. Designing Object-oriented Query Languages for GIS: Human Interface Aspects. In *Proc 3rd International Symposium on Spatial Data Handling*, pages 79–98, 1988.

[EF88b] M. J. Egenhofer and A. Frank. Towards a Spatial Query Language: User Interface Considerations. In *Proceedings of the International Conference on Very Large Data Bases*, pages 124–133, Los Angeles, California, 1988.

[Ege92] M. Egenhofer. Why not SQL! *International Journal of Geographical Information Systems*, 6(2):71–86, 1992.

[Ege94] M. Egenhofer. Spatial SQL: A Query and Presentation Language. *IEEE Transactions on Knowledge and Data Engineering*, 6(1):86–95, 1994.

[EH93] M. J. Egenhofer and J. R. Herring. *Querying a Geographical Information System*, chapter 10, pages 124–135. In Medyckyj-Scott and Hearnshaw [MSH93], 1993.

[Env92] Environmental Systems Research Institute, Inc. *ARC-INFO: GIS Today and Tomorrow*, 1992.

[Goh89] P-C. Goh. A Graphic Query Language for Cartographic and Land Information Systems. *International Journal of Geographical Information Systems*, 3(3):245–256, 1989.

[Gol92] C. Gold. The meaning of neighbor. In *Proc International Conference on GIS - From Space to Territory: Theories and Methods of Spatial Reasoning*, Springer Verlag Lecture Notes in Computer Science 639, pages 220–235, 1992.

[Gou93] M. D. Gould. *Two views of the user interface*, chapter 8, pages 101–110. In Medyckyj-Scott and Hearnshaw [MSH93], 1993.

[GR93] O. Günter and W. Riekert. The design of godot: an object-oriented geographic information system. *Bulletin of the Technical Committee on Data Engineering*, 16(3):4–9, September 1993. Special Issue on Geographical Information Systems.

[Gut88] R. Guting. Geo-relational Algebra: a Model and Query Language for Geometric Database Systems. In *Proc. 1st EDBT Conference*, pages 506–527, 1988.

[Gut92] J. Guttenberg. Towards a Behavioral Theory of Regionalization. In *Proc International Conference on GIS - From Space to Territory: Theories and Methods of Spatial Reasoning*, Springer Verlag Lecture Notes in Computer Science 639, pages 110–121, 1992.

[Kuh91] W. Kuhn. Are Displays Maps or Views? In *Auto-Carto 10*, pages 588–598, 1991.

[Lor91] R. Lorie. The Use of a Complex Object Language in Geographic Data Management. In *Proc. 2nd Symposium Spatial Database Systems*, pages 319–337. Springer Verlag Lecture Notes in Computer Science 525, 1991.

[LR93] J. P. Lagrange and A. Ruas. Etat de l'art en generalization. Technical report, IGN/DT/SR/COGIT, April 1993.

[LS92] M. Lindholm and T. Sarjakoski. User Models and Information Theory in the Design of a Query interface for GIS. In *Proc. International Conference on GIS - From Space to Territory: Theories and Methods of Spatial Reasoning*, Springer Verlag Lecture Notes in Computer Science 639, pages 328–347, 1992.

[MGR91] D. J. Maguire, M. F. Goodchild, and D. W. Rhind, editors. *Geographical Information Systems - volume II - Applications.* John Wiley and Sons, 1991.

[MM91] J. Mamou and C. B. Medeiros. Interactive Manipulation of Object-Oriented Views. In *Proc. of the International Conference on Data Engineering*, pages 60–69, Kobe, Japan, April 1991.

[Mon92] M. Monmonier. Time and Motion as Strategic Variables in the Analysis and Communication of Correlation . In *Proc 5th International Symposium on Spatial Data Handling*, pages 72–81, 1992. Volume 1.

[MP94] C. B. Medeiros and F. Pires. Databases For GIS. *ACM SIGMOD Record*, 1994.

[MSH93] D. Medyckyj-Scott and H. M. Hearnshaw, editors. *Human Factors In Geographical Information System.* Belhaven Press, 1993.

[OA93a] J. L. Oliveira and R. O. Anido. Browsing and Querying in Object Oriented Databases. In *Proc. 2nd International Conference on Information and Knowledge Management*, pages 364–373, Washington, DC, USA, November 1993.

[OA93b] J. L. Oliveira and R. O. Anido. Integração de uma interface para navegação em diferentes sistemas de bancos de dados orientados a objetos. In *Proc. 13th Brasilian Computer Society Congress*, pages 61–75, Florianópolis, SC, Brasil, September 1993.

[Oli94] J. L. Oliveira. On the Development of User Interface Systems for Object-Oriented Databases. In *Proc. ACM Workshop on Advanced Visual Interfaces*, pages 237–239, Bari, Italy, June 1994.

[OM95] J. L. Oliveira and C. B. Medeiros. A Direct Manipulation User Interface for Querying Geographic Databases. In *Proc. 2nd International Conference on Applications of Databases*, Sao Jose, USA, December 1995.

[Ooi90] B. C. Ooi. *Efficient Query Processing in Geographic Information Systems*, chapter 2. Springer Verlag Lecture Notes in Computer Science 471, 1990.

[Peu94] D. Peuquet. It's About Time: A Conceptual Framework for the Representation of Temporal Dynamics in Geographic Information Systems. *Annals of the Association of American Geographers*, September 1994.

[PMP93] N. Pissinou, K. Makki, and E. Park. Towards the Design and Development of a New Architecture for Geographic Information Systems . In *Proc. 2nd International Conference on Information and Knowledge Management*, pages 565–573, Washington, DC, USA, November 1993.

[PZ91] P.Svensson and Z.Huang. Geo-Sal: A query Language for Spatial Data Analysis. In *Proc. 2nd Symposium Spatial Database Systems*, pages 119–140. Springer Verlag Lecture Notes in Computer Science 525, 1991.

[Rig95] P. Rigaux. *Interfaces Graphiques pour Bases de Données Spatiales: Application à la Représentation Multiple*. PhD thesis, CEDRIC - Conservatoire National des Arts et Metiers, 1995.

[ROG88] D. Rhind, S. Openshaw, and N. Green. The Analysis of Geographical Data: Data Rich, Technology Adequate, Teory Poor . In *Proc. 4th International Working Conference on Statistical and Scientific Database Management*, Springer Verlag Lecture Notes in Computer Science 339, pages 427–454, 1988.

[Voi91] A. Voisard. Towards a Toolbox for Geographic User Interfaces. In *Proc. 2nd Symposium on Spatial Database Systems*, pages 75–97. Springer Verlag Lecture Notes in Computer Science 525, 1991.

[Voi94] A. Voisard. Designing and Integrating User Interfaces of Geographic Database Applications. In *Proc. ACM Workshop on Advanced Visual Interfaces*, pages 133–142, Bari, Italy, June 1994.

[VS94] A. Voisard and H. Schweppe. A Multilayer Approach to the Open GIS Design Problem. In *Proc. 2nd ACM Workshop on Advances in GIS*, pages 23–29, Maryland, USA, December 1994.

[VvO92] T. Vijlbrief and P. van Oosterom. The GEO++ system: an extensible GIS. In *Proc. Symposium on Spatial Data Handling*, pages 40–50, August 1992.

[Woo93] M. Wood. *Interacting with maps*, chapter 9, pages 111–123. In Medyckyj-Scott and Hearnshaw [MSH93], 1993.

Relatórios Técnicos – 1995

95-01 **Paradigmas de algoritmos na solução de problemas de busca multidimensional,** *Pedro J. de Rezende, Renato Fileto*

95-02 **Adaptive enumeration of implicit surfaces with affine arithmetic,** *Luiz Henrique de Figueiredo, Jorge Stolfi*

95-03 **W3 no Ensino de Graduação?,** *Hans Liesenberg*

95-04 **A greedy method for edge-colouring odd maximum degree doubly chordal graphs,** *Celina M. H. de Figueiredo, João Meidanis, Célia Picinin de Mello*

95-05 **Protocols for Maintaining Consistency of Replicated Data,** *Ricardo Anido, N. C. Mendonça*

95-06 **Guaranteeing Full Fault Coverage for UIO-Based Methods,** *Ricardo Anido and Ana Cavalli*

95-07 **Xchart-Based Complex Dialogue Development,** *Fábio Nogueira de Lucena, Hans K.E. Liesenberg*

95-08 **A Direct Manipulation User Interface for Querying Geographic Databases,** *Juliano Lopes de Oliveira, Claudia Bauzer Medeiros*

95-09 **Bases for the Matching Lattice of Matching Covered Graphs,** *Cláudio L. Lucchesi, Marcelo H. Carvalho*

95-10 **A Highly Reconfigurable Neighborhood Image Processor based on Functional Programming,** *Neucimar J. Leite, Marcelo A. de Barros*

95-11 **Processador de Vizinhança para Filtragem Morfológica,** *Ilka Marinho Barros, Roberto de Alencar Lotufo, Neucimar Jerônimo Leite*

95-12 **Modelos Computacionais para Processamento Digital de Imagens em Arquiteturas Paralelas,** *Neucimar Jerônimo Leite*

95-13 **Modelos de Computação Paralela e Projeto de Algoritmos,** *Ronaldo Parente de Menezes e João Carlos Setubal*

95-14 **Vertex Splitting and Tension-Free Layout,** *P. Eades, C. F. X. de Mendonça N.*

95-15 **NP-Hardness Results for Tension-Free Layout,** *C. F. X. de Mendonça N., P. Eades, C. L. Lucchesi, J. Meidanis*

95-16 **Agentes Replicantes e Algoritmos de Eco,** *Marcos J. C. Euzébio*

95-17 **Anais da II Oficina Nacional em Problemas Combinatórios: Teoria, Algoritmos e Aplicações,** *Editores: Marcus Vinicius S. Poggi de Aragão, Cid Carvalho de Souza*

95-18 **Asynchronous Teams: A Multi-Algorithm Approach for Solving Combinatorial Multiobjective Optimization Problems,** *Rosiane de Freitas Rodrigues, Pedro Sérgio de Souza*

95-19 **wxWindows: Uma Introdução,** *Carlos Neves Júnior, Tallys Hoover Yunes, Fábio Nogueira de Lucena, Hans Kurt E. Liesenberg*

95-20 **John von Neumann: Suas Contribuições à Computação,** *Tomasz Kowaltowski*

95-21 **A Linear Time Algorithm for Binary Phylogeny using PQ-Trees,** *J. Meidanis and E. G. Munuera*

95-22 **Text Structure Aiming at Machine Translation,** *Horacio Saggion and Ariadne Carvalho*

95-23 **Cálculo de la Estructura de un Texto en un Sistema de Procesamiento de Lenguaje Natural,** *Horacio Saggion and Ariadne Carvalho*

95-24 **ATIFS: Um Ambiente de Testes baseado em Inje,c ao de Falhas por Software,** *Eliane Martins*

95-25 **Multiware Plataform: Some Issues About the Middleware Layer,** *Edmundo Roberto Mauro Madeira*

95-26 **WorkFlow Systems: a few definitions and a few suggestions,** *Paulo Barthelmess and Jacques Wainer*

95-27 **Workflow Modeling,** *Paulo Barthelmess and Jacques Wainer*

Relatórios Técnicos – 1996

96-01 **Construção de Interfaces Homem-Computador: Uma Proposta Revisada de Disciplina de Graduação,** *F'abio Nogueira Lucena and Hans K.E. Liesenberg*

96Abs **DCC-IMECC-UNICAMP Technical Reports 1992–1996 Abstracts,** *C. L. Lucchesi and P. J. de Rezende and J.Stolfi*

96-02 **Automatic visualization of two-dimensional cellular complexes,** *Rober Marcone Rosi and Jorge Stolfi*

Instituto de Computação
Universidade Estadual de Campinas
13081-970 – Campinas – SP
BRASIL

`reltec@dcc.unicamp.br`