

# A Comparison Of Two Direct-Manipulation Gis User Interfaces For Map Overlay\*

JAMES R. RICHARDS and MAX J. EGENHOFER

National Center for Geographic Information and Analysis and Department of Spatial Information Science and Engineering, Department of Computer Science, University of Maine, Orono, ME 04469-5711, U.S.A.

## Abstract

We have designed two different direct-manipulation user interfaces for Geographic Information Systems that are based on the map-overlay metaphor. In both interfaces, users explore geographic information by moving themes of spatial data onto a *viewing platform*, on which they can examine them in an integrated fashion. The first user interface separates the content of a theme and its graphical presentation into two different parts, referred to as the *data cube* and the *template*. Both parts are manipulated independently enabling users to view the same data in different ways by replacing one template with another. The second user interface combines the content of a theme and its presentation into a single representation, called a *layer*. For both user interfaces, iconic languages are introduced that resemble the manipulations on the Macintosh desktop. To compare the two visualizations of the user interfaces, we developed a quantitative model which measures the conceptual and operational complexities of a direct-manipulation user interface. This new method is based on the cognitive walkthrough analysis and considers (1) the number of concepts a user has to learn before performing a particular operation in the user interface; (2) the number of individual actions the user has to perform when executing a particular operation; and (3) the number of errors a user might run into when executing an operation. Applied to a set of most common tasks undertaken when manipulating geographic themes, the comparison reveals that the layer interface is less complex than the cube-and-template interface in all three categories.

## 1 Introduction

The design of user interfaces for Geographic Information Systems (GISs) has become a prominent research topics over the last few years (Kuhn and Egenhofer, 1991, Mark and Gould, 1991, Mark and Frank, 1992, Medyckyj-Scott and Hearnshaw, 1993). For a long time, GIS user interfaces were limited to typed command languages requesting query results in graphical or alphanumeric form. Such GIS user interfaces have been addressed as tailored query and manipulation languages (Tomlin, 1990) or as generic spatial query languages, primarily SQL extensions (Frank, 1982, Ingram and Phillips, 1987, Herring *et al.*, 1988, Egenhofer, 1991). Only recently, through the increasing influence of the successful Macintosh user interface, has the application of advanced considerations about human-computer interaction for GISs been discussed (Gould and McGranaghan, 1990, Kuhn and Frank, 1991, Kuhn, 1992, Kuhn, 1993).

This paper focuses on direct-manipulation user interfaces of the kind of a Macintosh desktop, which invites users to perform manipulations intuitively. We base this user-interface design on the well-known concept of *map overlay* and “bring maps onto the desktop.” Map overlay refers to the manual process of drawing a separate, single-factor transparent sheet for each theme, placing several of such sheets over one another on a light table, and viewing the resulting integrated spatial information that would not have been visible on a single sheet. The separation of geographic data into themes—frequently also referred to in the literature as *layers* or *coverages*—has been used in GIS since its early days. It is considered to be one of the major conceptualizations of geographic data and has influenced the design of many GISs. For instance, Odyssey and its successors base their implementation model on this conceptual model of organizing geographic data (Chan and White, 1987). Map overlay appeals to users through its coherent set of analytical operations upon themes. It represents a closed system as any overlay

operation acts upon one or several layers, and results always in a layer. Tomlin's (1990) MAP algebra describes this framework with a comprehensive collection of such overlay operations.

Traditionally, overlay-based GISs have had command line interfaces requiring the user to learn a cryptic language and extensive vocabulary in order to operate the system. More advanced implementations of a GIS based on Tomlin's MAP algebra offer a menu-based user interface (Pazner *et al.*, 1989) or a visualization of the commands through icons for the objects involved *and* the operations upon them (Kirby and Pazner, 1990). Similarly, a hypertext-based GIS user interface translates commands into buttons (Linsey and Raper, 1993). While these advanced interaction techniques release the users from remembering and typing the commands, they still preserve the initial command structure of the underlying language and users have to think in terms of constructing "sentences."

In a previous paper (Egenhofer and Richards, 1993a), we introduced a direct-manipulation user interface for map overlay. With that interface, users interact with the GIS by directly manipulating thematic data, represented as *data cubes*, and their graphical presentations, called *templates*, by stacking them vertically onto a *viewing platform*. The separation of a thematic layer into data and graphical presentation offers great flexibility in viewing geographic information in different formats. On the screen, data cubes and templates are represented as icons, which users manipulate directly with a mouse and for which they receive immediate feedback about successful or unsuccessful operations.

This paper reports on the lessons learnt from the design, prototyping, and subsequent feedback from users about this interface. As a consequence, we propose an alternative user interface that uses the successful features of the first interface, while overcoming some of the problems that users reported after observing a mockup and a prototype of the cube-and-template interface. This revised user interface uses a single *layer* that provides a horizontal integration of the data cube and its corresponding template. Within a layer, users may individually change the content as well as the display parameters. To assess whether the second user interface promises to be a better design, the two user interfaces are compared for complexity and analyzed for ease of use. Based on existing methodologies in the field of human-computer interaction for assessing user interfaces, we developed a comprehensive, but simple analytical method to compare the complexity of different user interfaces designed in the same user-interface style and for the same task. Using this evaluation method, the layer interface was found to be less complex than the cube-and-template interface.

The remainder of this paper is structured as follows: After an introduction of the cube-and-template interface in Section 2, a discussion of its pros and cons follows in Section 3. Section 4 presents the layer interface as an alternative to the cube-and-template interface. Section 5 introduces a novel method to assess the complexity of direct-manipulation user interfaces, which is used in Section 6 to compare the cube-and-template interface with the layer interface. The paper concludes in Section 7 with a summary of the major findings and a discussion of future work.

## **2 Cube-and-template interface**

The first of the two user interfaces that we will compare is based on the concept of separating geographic information into two components: one for database retrieval and another for graphic presentation. Such a separation has been advocated for both conceptual and operational reasons (Frank, 1992). Its three principal objects are: (1) the data cube, (2) the template, and (3) the viewing platform (Egenhofer and Richards, 1993a).

## 2.1 Data Cubes

A *data cube*, or briefly *cube*, represents the spatial and attribute data of a single theme. Simple examples of themes are roads or vegetation, with more complex examples including land parcels assessed at over \$100,000 or ski areas within 50 miles of a major airport. Each cube has an inherent spatial location, orientation, and extent. It is important to understand that a cube represents only data, without any implied knowledge about their graphic presentation.

The concept of a cube was chosen for three reasons:

- Cubes naturally convey the multi-dimensionality of the data that they represent: Two for the spatial extent and a third for some kind of attribute information (Frank, 1992).
- Cubes are very basic to human understanding of spatial relationships and to graphic communication. Most of us as children experience space while playing with “building blocks.” The prototypical operation associated with such blocks is stacking them. Data cubes correspond to such blocks. They are a very natural representation for an object to be manipulated spatially and provide a strong clue to those operations that can perform on spatial data.
- The cube structure frees the naive user from needing to know how data is stored and retrieved from the database. Much like the object-oriented concepts of information hiding and data encapsulation (Khoshafian and Abnous, 1990), GIS users see only a “black box.” They know about the contents of that box, but do not have to understand the internal representation.

The primary operation upon cubes is to combine them. Cubes covering the same geographic extent can be stacked on top of one another, much like the transparent sheets that are put on top of one another in map overlay, forming a stack of cubes. Conversely, individual cubes may be removed from a stack.

## 2.2 Templates

A *template* is a rule set that describes how to render the data in a cube. For example, a graphical state-boundary template may specify that boundaries in the associated cubes are to be drawn as dashed yellow lines, 0.6 mm thick. Cubes may then be combined with templates to apply the display specifications to their content. A template is mandatory in order to view a cube. Since a cube is merely raw data, it cannot be explored without an appropriate template. *Stacks of templates* are concise representations of combinations of templates. Unlike the cubes in a stack of cubes, the templates in a stack of templates are ordered and this stacking order, from bottom to top, determines the sequence in which the data is rendered.

## 2.3 Viewing Platform

The viewing platform represents the “light table” in the source domain of the map-overlay metaphor. On the geographer’s desktop, it is the object onto which cubes and templates are stacked in order to become visible. It is associated with a window in which the data may be displayed. Users place cubes and templates onto the platform if they want to view the contents of the cubes. Likewise, users may remove cubes or templates from the platform to erase them from the current view.

## 2.4 A Direct-Manipulation Visualization

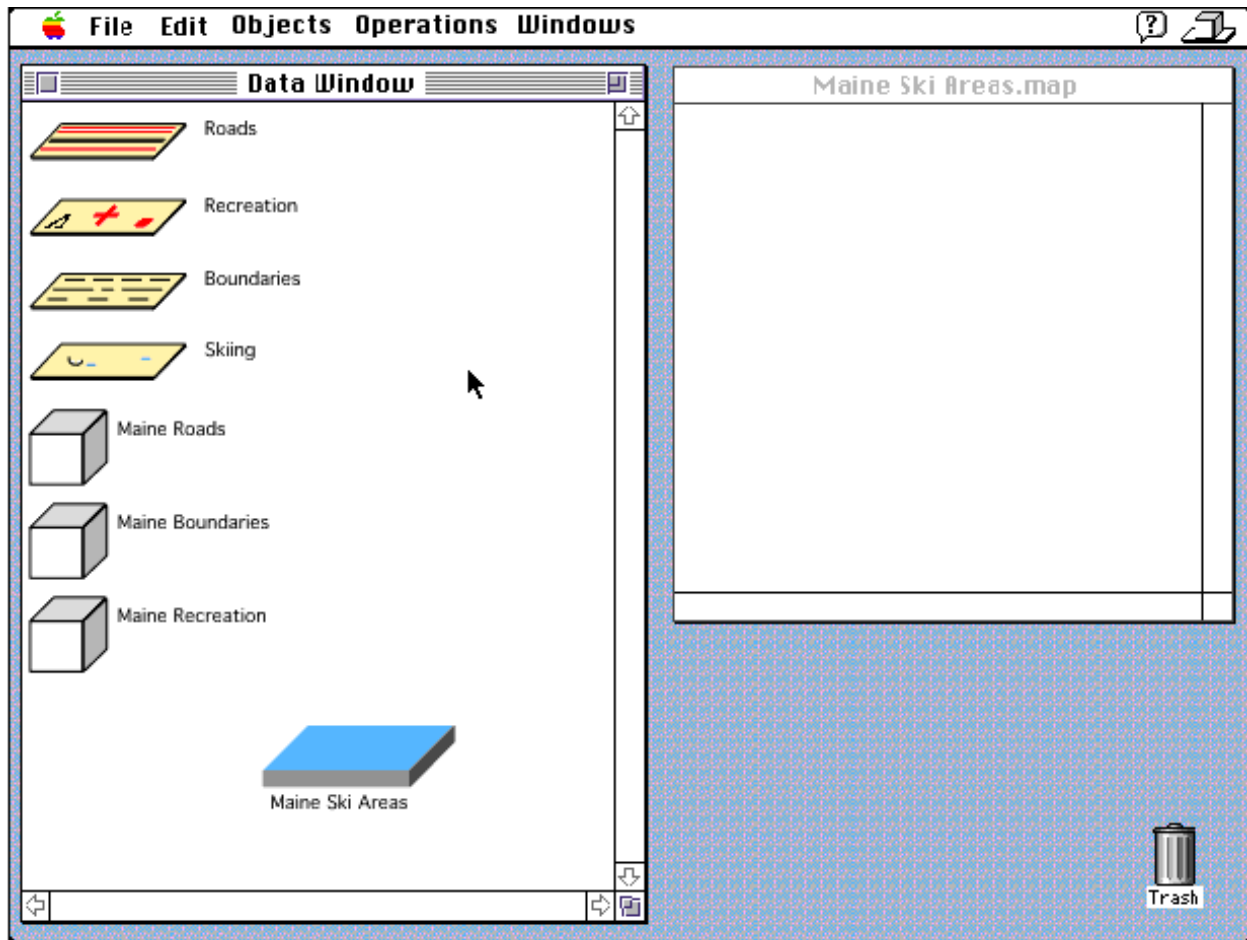
The three concepts of cubes, templates, and platforms have been implemented in a direct-manipulation language. The environment, called the *geographer’s desktop* (Egenhofer and Richards, 1993b), resembles conceptually and visually the Macintosh desktop (Apple, 1987). On the screen, cubes and templates are represented as icons, which users manipulate directly with a

mouse by moving them onto a platform icon. Typical direct-manipulation techniques such as selection and dragging apply to all icons.

The icons are dynamic since they change their appearances when they are in different states of the manipulation to provide feedback to the users. Such feedback allows users to determine immediately whether or not an operation is producing the desired result. For example, while placing a cube on top of a template, the cube's faces reflect the different states during the overlay, i.e., whether the cube is viewable because it is placed on the platform together with a template such that the template can render the cube; or whether is unviewable because it is either inactive (not placed on the platform) or on the platform, but without a corresponding template. A viewable cube displays a color icon on its face to indicate which view is currently available for the user to explore, while an unviewable cube icon reflects no information about the cube's display status. As meaningful combinations of cubes and templates are stacked on the viewing platform, feedback is provided in two ways: (1) the results of operations are shown in the appropriate viewing windows, and (2) visual clues are given by the icons themselves. For a more detailed discussion of feedback when manipulating cubes and templates, the reader is referred to Egenhofer and Richards (1993a).

## 2.5 Guided Tour

The following guided tour leads the reader through a small sequence of screen snapshots that reflect a scenario in which a user wants to find information about Maine's ski areas. At the beginning of the session the user opens an object window, the "Data Window," in which the user directly manipulates cubes and templates. In the object window, the user creates a viewing platform named "Maine Ski Areas," with which a viewing window is associated where geographical rendering takes place. The viewing platform has the platform's name plus the suffix ".map" as its title. The user loads some cubes and templates into the data window: Maine Roads and Maine Boundaries are included to provide the necessary contextual information for a geographic frame of reference; and Maine Recreation contains information about the ski areas in Maine that the user is interested in. The four templates—Roads, Boundaries, Recreation, and Skiing—specify how to render cubes in the view window (Figure 1).



**Figure 1:** The cube-and-template interface with three cubes (Maine Roads, Maine Boundaries, and Maine Recreation), three templates (Roads, Boundaries, and Recreation), and a viewing platform (Maine Ski Areas).

The user wishes to see the geographic locations of ski areas in the state of Maine, and moves first three graphic templates—Roads, Boundaries, and Recreation—and then two of the cubes, Maine Boundaries and Maine Recreation, onto the viewing platform. Upon completion of the operations, the user immediately receives feedback that it was successful, as is indicated by the colored graphic icons on the face of the two cubes and the rendering of the data in the map window as specified by the templates (Figure 2). Satisfied with the success of the system thus far, the user decides to add the remaining cube (Maine Roads) to the platform. She selects this object and drags it into a position where the platform can attract it and stack it neatly on top of itself. As the object is being dragged, the platform highlights when the cursor enters the zone directly above the platform, thus informing the user that releasing the mouse at this time will result in the desired action. When the mouse is released the data in Maine Roads are added to the map view.

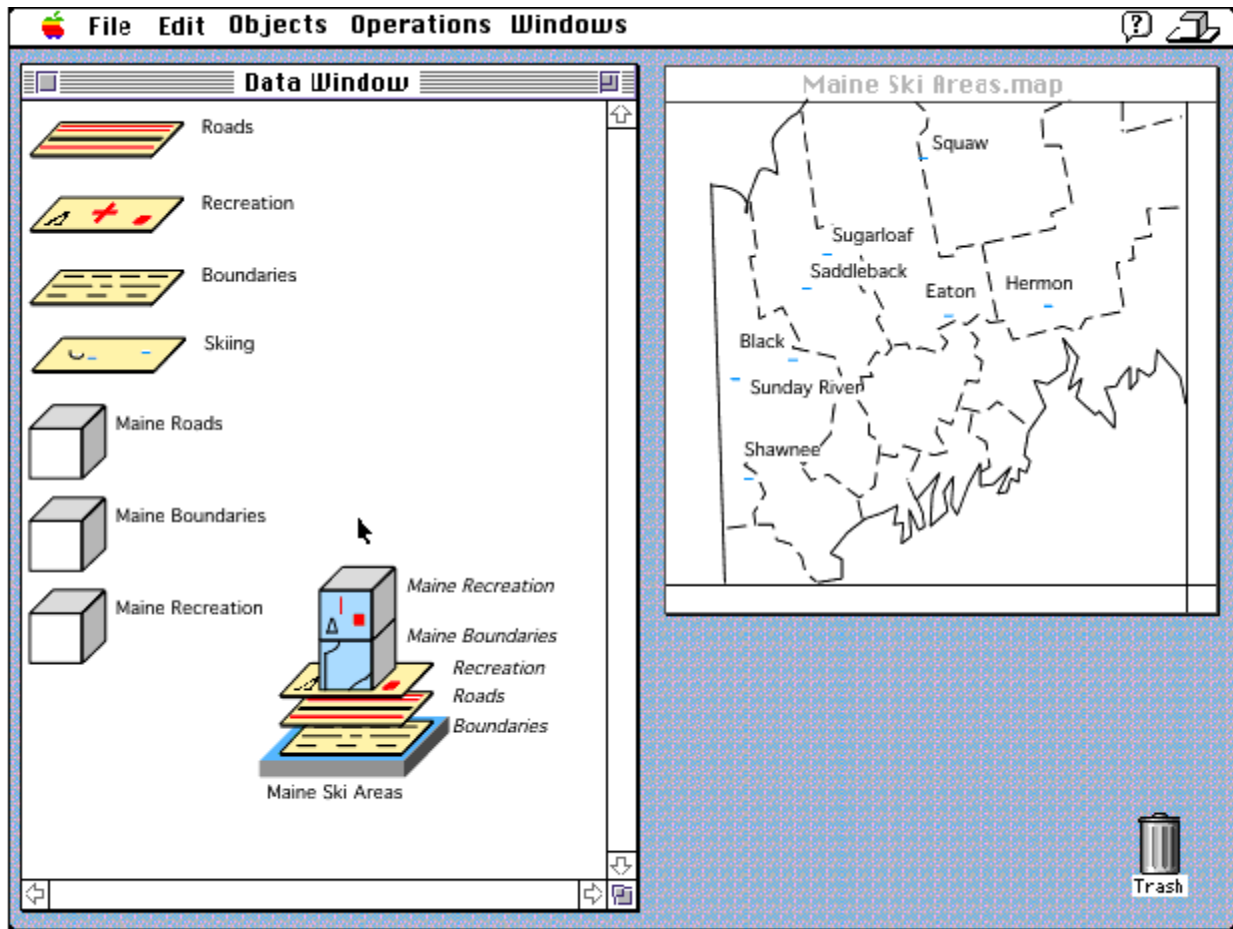
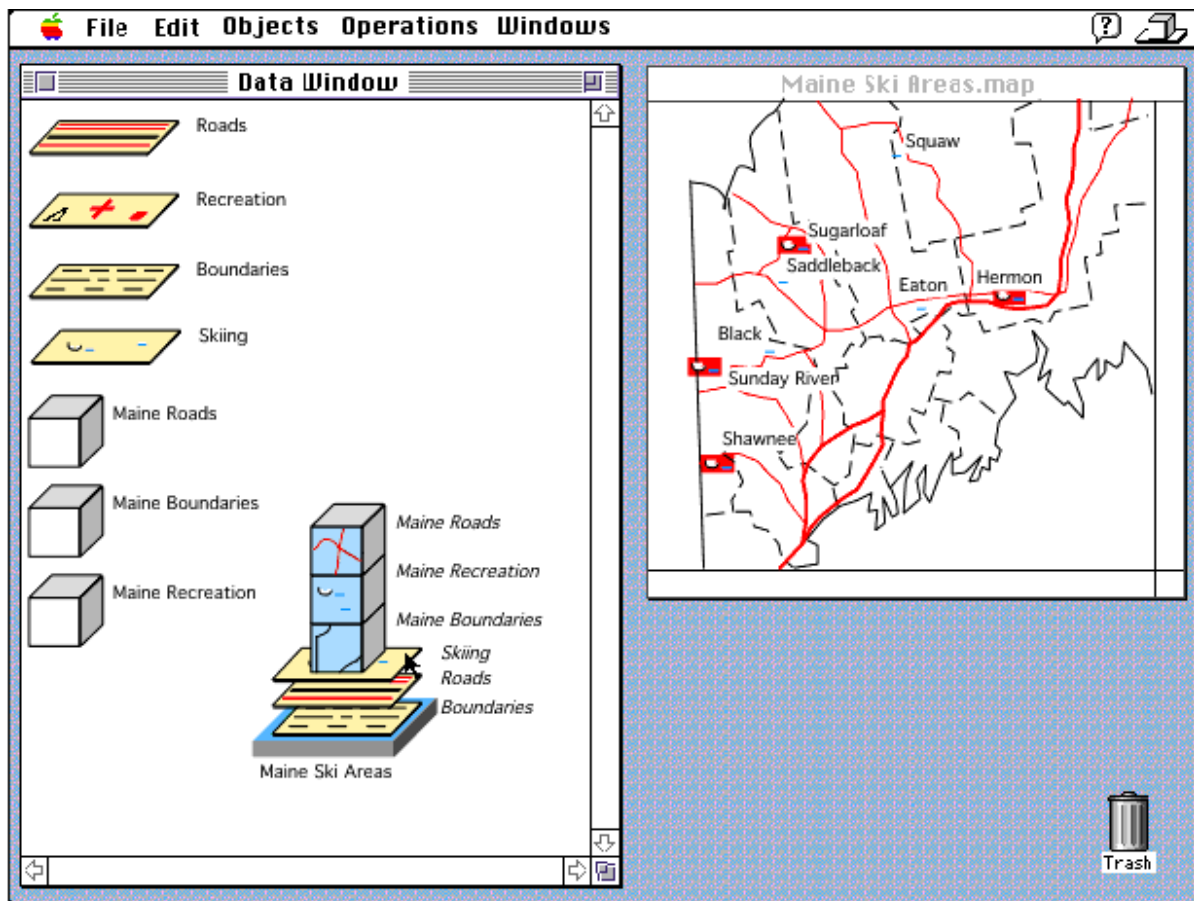


Figure 2: The cub-and-template interface after two cubes and three templates have been put onto the viewing platform.

In this scenario, the user is interested in snowboarding; however, there is no graphic indication in the map window that differentiates ski areas with half-pipes from those without. The user will change this situation by replacing the Recreation template with the Skiing template, which was set up to distinguish between two types of ski areas, those with half-pipes and those without. As soon as the new template is placed on the viewing platform, the system redraws the map view to reflect the new symbolization (Figure 3).



**Figure 3:** The cube-and-template interface after changing the symbology to distinguish different types of skiing areas by replacing the Recreation template with the Skiing template.

### 3 Assessment

User feedback from presentations of the cube-and-template interface yielded primarily positive comments about the design. Feedback was collected informally during demonstrations and discussions with potential users of this user interface. The audiences ranged from experienced GIS users, researchers, and developers; over GIS-illiterate, but geographically educated individuals; to complete novices who were familiar with neither GIS nor the map-overlay metaphor. Approximately 250 individuals participated in providing feedback.

Most observers reported that they were impressed with the ease and adaptability of direct manipulation and map overlay within the realm of exploring geographic information. In a number of instances, the users found the interaction with the system so natural that they questioned why no one had done it before. The flexibility of separating the database selection from the visualization parameters was considered very appealing and powerful. Many users envisioned how easy it would be, for example, to compile just one USGS 1:24,000 symbol set as an aggregate of templates (Egenhofer and Richards, 1993a) and then apply it to a number of cubes at different spatial extents for comparison. Likewise, the rich set of semantic and operational feedback was generally considered as very helpful; however, one observer pointed out that a cube without a corresponding template—shown as an empty cube face—could be also interpreted as a cube without any data.

With the flexibility, however, comes an increased conceptual difficulty, especially for naive and inexperienced users. Many comments addressed the fact that it was not immediately apparent

how the cubes and templates related to one another, and that particularly the *vertical* separation of two related objects seemed somewhat unnatural. Sometimes, questions were raised about the fact that the stacking order of templates was meaningful, while stacks of cubes are considered unordered sets. This was considered an inconsistency in the interface design.

## 4 Layer interface

To overcome the anticipated problems with the cube-and-template interface, a second user interface was designed for the same problem. Such a step is common in the iterative development of user interfaces, when user interface designers consider user testing or other evaluation methods to improve a design (Nielsen, 1993). Improvements may be bug fixes in the interaction or reconceptualizations of the interface. Since the user feedback about the cube-and-template interface did not indicate major “bugs,” the next design phase concentrates on overcoming the anticipated problems from the separation into separate icons for data and presentation. The goal of this second phase is to find an interface that retains the ideas and functionality of separating the components of geographic information, but connects them conceptually and operationally closer into a cohesive interface object.

Similar to the cube-and-template interface, our second interface is also based on map overlay; however, unlike the cube-and-template interface, it has only two principal objects—the *layer* and the *viewing platform*—with which the user interacts. The platform keeps its initial functionality, while the new concept of a layer accommodates interaction with both database retrieval and graphic presentation. Therefore, data and their presentation are more intimately linked.

### 4.1 Layers

The concept of a *layer* is borrowed from the map-overlay metaphor, where physical maps exist as map layers. Generally, physical maps have a legend and an area for map drawing. The *legend* allows the map reader to interpret the various map symbols, and the *map drawing* contains the graphic rendering of the map as defined by that symbolization. On the geographer’s desktop, the legend corresponds to the symbolization component and the drawing corresponds to the database selection component as rendered by the symbolization rules. These concepts are brought together to form the *layer object* at the user interface.

### 4.2 Viewing Platform

The viewing platform retains its primary use as the location where map overlay occurs; however, its functionality is extended with the selection and definition of default *symbol sets* for layers placed onto the platform. A symbol set contains the display specification for a series of themes. Such symbol sets can be thought of as the union of the legends associated with several layers. Examples of such symbol sets are the standardized cartographic symbols for USGS 1:24,000 map sheets or for Rand McNally road maps. Unless a layer’s legend specifies explicitly to overwrite the platform’s current symbol set, the content of the layer will be displayed according to the symbol set associated with the platform.

### 4.3 A Visualization in a Direct-Manipulation Environment

The integration of cube and template into a layer has consequences for the operations at the user interface, because a single direct-manipulation operation on the geographer’s desktop will activate data *and* their graphical presentation. On the other hand, a layer needs two separate operations to change its components—one for choosing a different content, the other for selecting a different presentation. Recall that with the cube-and-template interface, it was necessary to apply two direct-manipulation operations, one to the cube and another to the corresponding template. Also when modifying content and presentation two operations were

assigned to two different objects, the cube and the template, and therefore, no ambiguities would arise with respect to what to change, database selection or presentation.

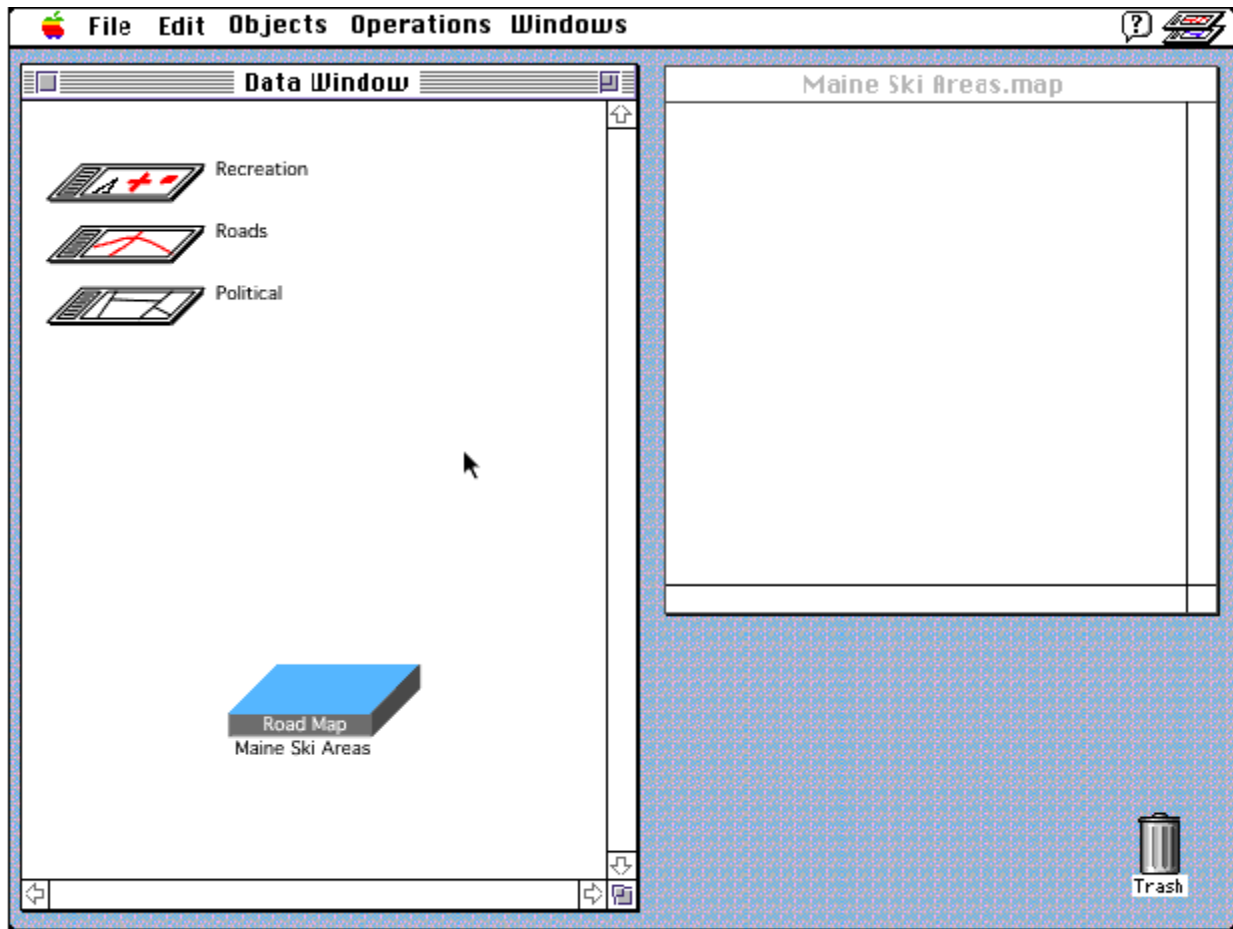
To provide the same functionality for a single object, a layer is visualized as an icon that consists of two halves, (1) the legend for the symbolization of the layer and (2) the map for the database selection (Figure 4). Any direct-manipulation operation that changes the location of the layer applies to both halves. On the other hand, the user can interact separately with each icon half when double clicking on the corresponding part. Double clicking on either the legend for symbolization or the map for database selection engages the user in a dialog with those parameters. While this partitioning of an icon is a deviation from the Macintosh interface guidelines (Apple, 1987) it is an interaction technique that can be found in other applications when an object is an aggregate of two parts and it has two principal operations that are executed on each part, and not the composite object.



**Figure 4:** The icon for a Roads layer. The left half of the icon represents the symbolization, while the right half represents the database selection.

#### 4.4 Guided Tour

The following guided tour through the layer visualization is structured similarly as the tour through the cube-and-template visualization. The first part—opening an object window and creating a viewing platform—remains the same; however, *in lieu* of templates and cubes, there are layers on the desktop. In this example, three layers have been loaded into the object window: *Recreation* contains information about recreational facilities, while *Roads* and *Political* contain contextual information for geographic reference (Figure 5).



**Figure 5:** The layer interface with three layers (Recreation, Roads, and Political) and the viewing platform (Maine Ski Areas), on which the symbol set Road Map has been selected as the default symbolization.

In order to examine the information in the layers, the user selects the three layers and drags them onto the viewing platform. Immediately, they are drawn in the map window (Figure 6).

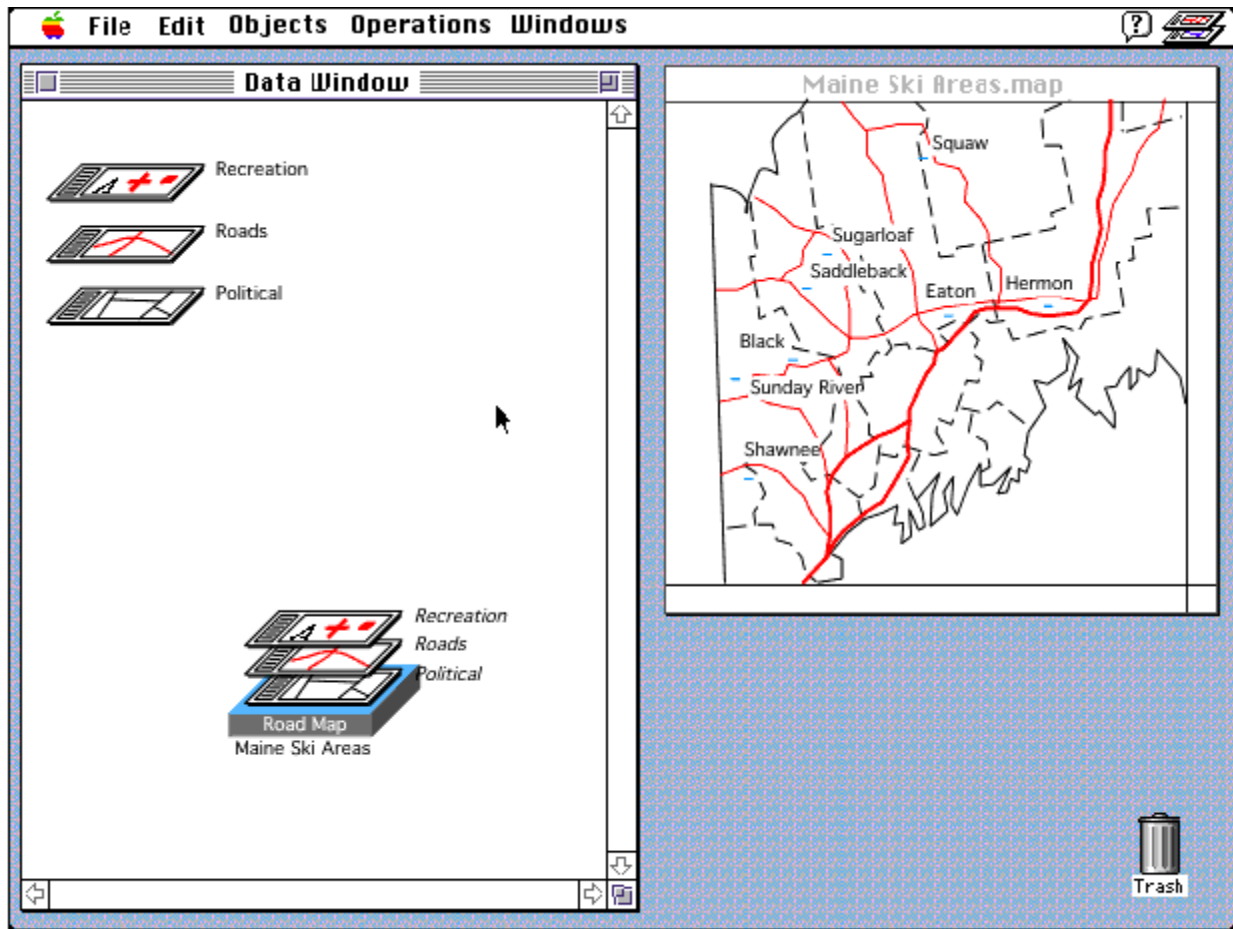
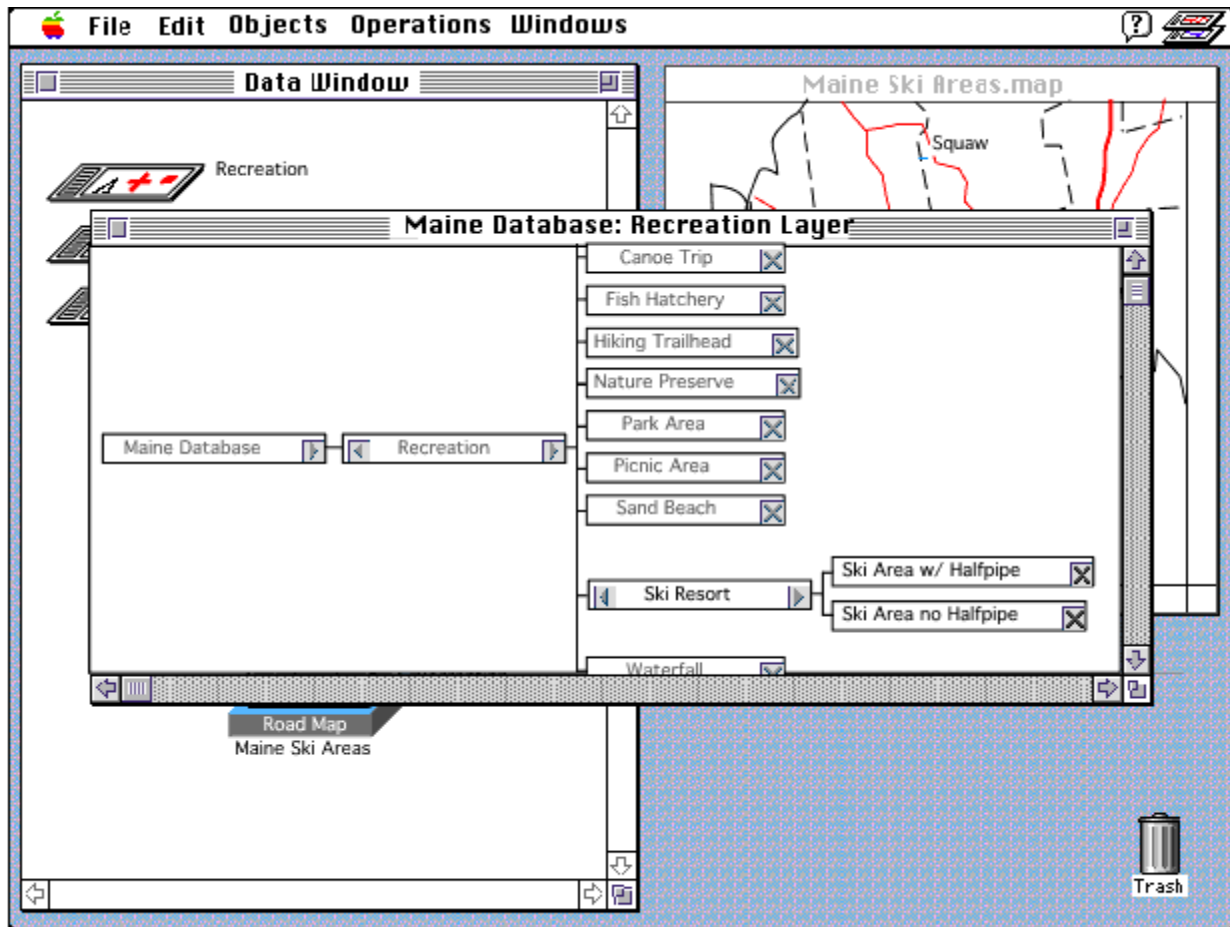


Figure 6: The layer interface after three layers have been put onto the viewing platform.

Again, the user is interested in snowboarding and wants to see the difference between general ski areas with half-pipes and those without. This time, appropriate database selection must be made before changing the symbolization. By double-clicking on the map drawing half of the Recreation layer icon, a dialog appears which presents the overview of the database hierarchy, from which the user activates the selection of half-pipes (Figure 7).



**Figure 7:** Refining the database selection of the Recreation layer to distinguish between ski areas with and without halfpipes. Afterwards, the user assigns a different symbol to each kind of area so that they appear differently in the map view. This is done by interacting with the symbolization component of the graphical Recreation layer. By double-clicking on the legend half of the icon, a dialog box appears which shows the user the current symbolization of all the recreation types. The user changes the symbol for ski areas with half-pipes to include a miniature u-shaped cross section next to the mountain (Figure 8). After the dialog is dismissed, the system redraws the map view to reflect the new symbolization. Additionally, the pop-up menu on the front of the viewing platform displays the current style collection with a “++” suffix to indicate that there has been a change in the symbolization of the style collection.

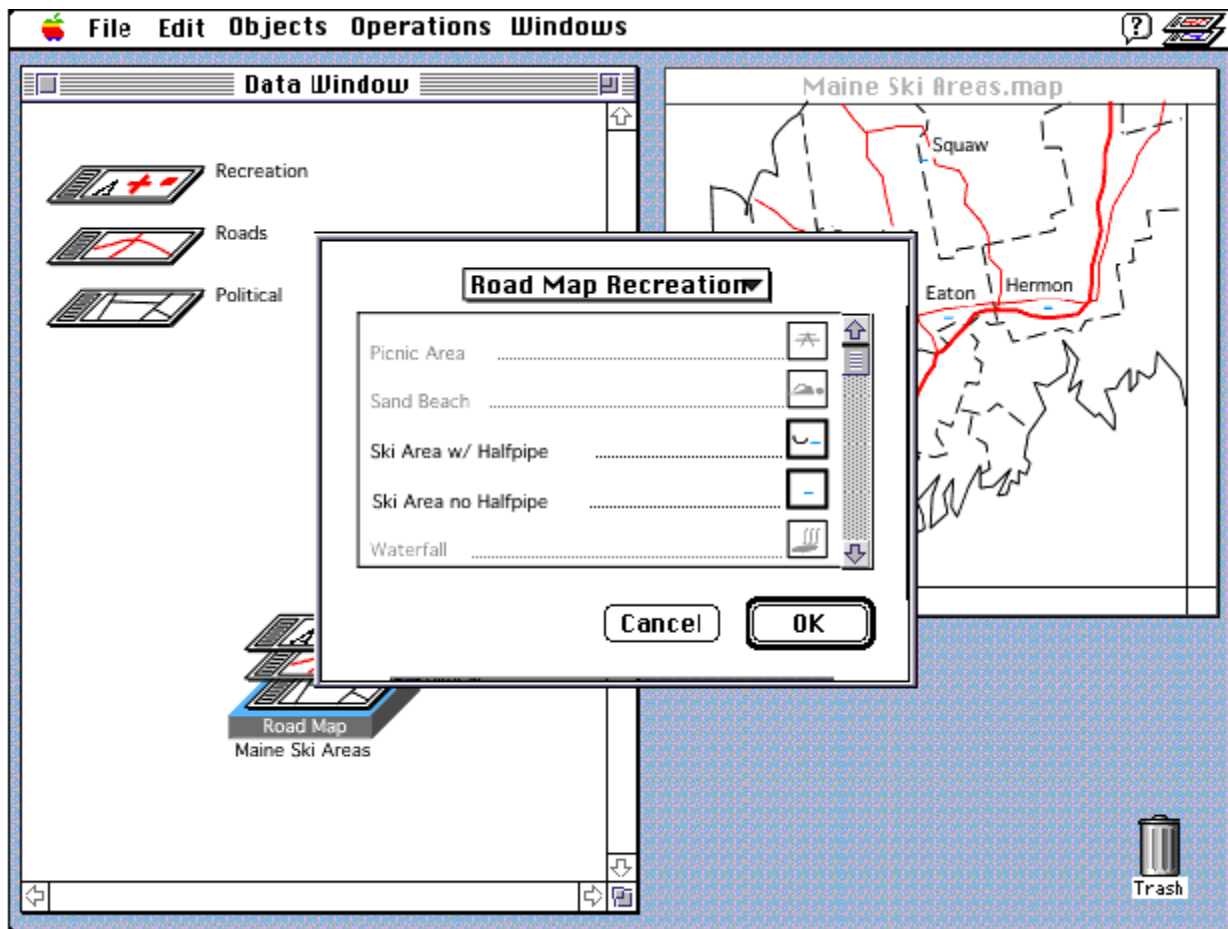


Figure 8: Changing the legend of the Recreation layer to have different symbols for ski areas with and without halfpipes.

## 5 Comparison methodology

Although the two interfaces allow their users to perform the same operations, the two designs differ considerably in the way users interact with them. An obvious example of this difference is when a user starts with an empty viewing platform and wants to visually examine one theme: with the cube-and-template interface, two icons—a cube and the corresponding template—would have to be selected, whereas with the layer interface a single icon—the layer—would have to be dragged onto the viewing platform.

In order to effectively evaluate these two visualizations of the map-overlay metaphor and to find out which of the two approaches promises a simpler interaction, it is necessary to compare them based on an analytical model. Such a model may be based on measures of user performance gained from *human-subject tests*, where all test persons execute the same task and an experimenter records such parameters as the time they needed to complete the task successfully or the number of errors they committed. The test instruments for human-subject evaluations are fairly expensive as they require, among other things, for each user interface a working implementation of the tasks to be tested. While sometimes postulated as the best evaluation method of user interfaces, human-subject tests are rarely found in the design of GIS user interfaces (Mark and Gould, 1991) and to date, no results from such tests of GIS user interfaces have been reported.

An alternative evaluation method to human-subject testing is the *a priori* analysis of the complexity of a user interface. Many such models exist that are based on the concept of *task analysis*, which examines the steps users take when completing a task. Unlike with testing human subjects, task analyses can be performed prior to the development of an expensive

prototype, while an interface is still on the drawing board. Certainly, task analysis methods do not measure actual user performance, but hypothesize what users would do and how they would perform. Frequently these analysis methods are such that they consider the best case (e.g., shortest time to perform the task) or worst-case scenarios (e.g., what are all the errors a user might commit).

This section first reviews three methods of assessing or predicting user performance, each of them has been applied in GIS user-interface design or is being discussed as viable measures to analyze GIS user interfaces. Lewis (1991) made a cognitive walkthrough analysis of some parts of Arc/Info. Haunold and Kuhn (1993) extended the Keystroke-Level Model to analyze user interfaces for digitizing maps. Finally, the use of task analysis techniques for GIS user interfaces is a central discussion point in the research plan of NCGIA's Initiative on GIS User Interfaces (Mark and Frank, 1992). We analyze the strengths and weaknesses of the three models and, using a combination of the strong points of these established techniques, develop a new methodology for comparing the visualizations of different user interfaces.

### 5.1 Task Analysis

Task analysis was originally designed to identify elements that should be included in job training. A typical example of task analysis is the description of jobs in terms of identifiable units of activities (McCormick, 1979). A task is a meaningful unit of work and consists of a set of related worker actions involving the worker's interaction with equipment, other people, the environment, and concepts. As such, task analysis is not a design tool for computer interfaces; however, it has been extended in certain instances to evaluate user interfaces. Task analysis as it relates to the design of human-computer interfaces is discussed by Phillips *et al.* (1988). Its analysis method is to subdivide a job into all of the tasks that might be performed, and to rate each task with respect to (1) frequency of performance, (2) relative importance, and (3) estimated learning difficulty. Each task is then broken down into the necessary steps for completion, and these are characterized with respect to the type of skills needed and the difficulty of learning.

### 5.2 The Keystroke-Level Model

The Keystroke-Level Model (Card *et al.*, 1980) of human-computer interaction is used to measure the time it takes for an advanced user to perform certain tasks in a particular computer system. It is based on the idea that a complete task can be broken down into a number of *unit tasks*, each with an associated time interval, which when added, yield the total time for the task. A unit task has two components, an acquisition time and an execution time. The acquisition time is the time it takes for a user to mentally formulate the task, and the execution time is the time it takes for the user to physically perform it. Unit tasks are compiled into *methods*, which represent a specific combination of unit tasks in a specific order that successfully complete the overall task.

The Keystroke-Level Model is very restricted in that it assumes an expert user, it assumes that unit tasks are routine, it requires detailed specification of methods, and it assumes an error free performance. Additionally, the model is only concerned with the time of interaction, and ignores the potential errors, learning, functionality, and recalls aspects of a given design.

### 5.3 Cognitive Walkthrough

Cognitive Walkthrough (Lewis *et al.*, 1990, Polson *et al.*, 1992) is a theory-based method for the evaluation of ease of use in user-interface designs. Strengths and weaknesses are identified by modeling required user steps for the successful completion of a given task. Each individual step of a task is then analyzed by filling out an evaluation about: (1) the goals of the task, (2) the ease of selecting the correct action, (3) the system response, and (4) the user's ability to evaluate the appropriate next action. Test subjects are then asked to simulate the steps of interaction. Deviations from the prescribed solution path are considered to be errors. Once the evaluation

forms and simulations are complete, the analyst can arrange their contents to demonstrate common errors in a typical interaction.

Cognitive Walkthrough is useful for identifying potential user and designer errors in the completion of tasks, given the strict definition of an error as a deviation from the prescribed solution path; however, it is limited in that it is designed for use in applications that do not require a great deal of formal training.

#### 5.4 Extended Cognitive Walkthrough

In order to compare the visualizations of the two (or more) user interfaces, we developed a method that is an extension of cognitive walkthrough, as well as a combination of the methods discussed above. It is based on the selection of a set of fairly high-level tasks whose goals and outcomes can be described independently of a particular implementation, such that they should be performable in either visualization. Each task can be broken down into more detailed implementation steps, which a user must perform in order to accomplish it. These steps depend on the type of the basic operations and manipulations available for each individual implementation. For example, for a direct-manipulation interface, the implementation steps may include identifying an object on the desktop, selecting from a menu, or dragging an icon. The number of individual steps necessary to perform a specific task is in part an indicator of the complexity of a user-interface visualization.

The systematic comparison of the conceptual complexity of the two visualizations will be based on (1) the amount of previous knowledge necessary to handle a visualization, (2) the basic steps necessary to perform the task, and (3) the potential errors a user might commit during the execution of a task. For each task, the prerequisites of user understanding will be presented as *assumptions* detailing the conceptual knowledge necessary to complete that task. This will be followed by the direct-manipulation *actions* needed for completion of the task. Finally, an *error analysis* will reveal possible semantic errors that could arise during the actions.

The measures in each category will determine the assessment of the complexity of a user-interface visualization: the less assumptions necessary, the faster a user will be able to learn a visualization; the less actions involved in a task, the faster a user will execute the task; and the less the number of potential errors, the higher the likelihood a user will succeed in performing the task. If performed over a set of tasks, a particular constraint applied to the concepts: Since a user learns a particular concept with the first task and will be able to re-apply it with any other task, it is a more accurate measure to count the number of *assumption primitives*, rather than all their occurrences. If all three measures for one visualization are lower than the corresponding measures for another visualization, then the task would be considered easier to perform in the first visualization.

This model contains some simplifications such as “all concepts are equally difficult to learn,” “all actions are of equal length,” and “all errors are equally fatal.” By including weighted concepts, actions, and errors, the model could be refined to account for differences among the items in each category.

## 6 Comparison

A set of common user tasks was selected, such that performing the five tasks in a particular order emulates an almost complete scenario of user interaction. They are (1) creating a new theme by specifying the database selection and the display parameters; (2) adding a theme to the currently visible themes; (3) removing a theme; (4) applying a different set of visualization parameters; and (5) changing the database selection for a theme. For each task, the assumptions, actions, and potential errors are displayed in Tables 1-5. Each individual step is referred to by a number *It.c.n*, where *I* stands for the interface type (C = Cube, L = Layer); *t* stands for the task number; *c* stands

for the assessment category (1 = Assumptions, 2 = Actions, 3 = Potential Errors); and  $n$  refers to the number of the individual step (1, 2, ...). In the assessment of the complexity of the tasks, it is assumed that the users have a sufficient knowledge of direct-manipulation techniques on a Macintosh so that they understand, for instance, the mapping from the pointing device onto the cursor and such operations as double clicking for opening an icon or point-press-move-release for dragging an icon across the screen (Apple, 1987).

### 6.1 Task 1: Make a New Theme

The first task for comparison is that of creating a theme (Table 1). In both interfaces, the theme consists of two components: (1) the database selection and (2) the visualization parameters. In order to create a theme, the user must specify the nature of these components. In the cube-and-template interface, this is done by the creation of the two principal objects, the data cube and the template (Actions C1.2.1-4). Besides knowing how to specify database selection (Assumption C1.1.2) and symbology (Assumption C1.1.3), the user must know that both a cube and a template are necessary to represent geographic data (Assumption C1.1.1). This separation of the components of a theme into cubes and templates makes the interface more flexible (i.e., order of Actions C1.2.1/2 and C1.2.3/4 is not important), but simultaneously introduces several potential user errors that are related to a possible misunderstanding of the relationship between the two components (Errors C1.3.1-2). In the layer interface, the user specifies the nature of the two components in parallel. Uniting the two components in this way is conceptually less straining on the user, because knowledge of only two concepts is necessary (Assumptions L1.1.1-2) vs. three concepts with the cube-and-template user interface (Assumptions C1.1.1-3), and it allows for concurrent specification of visualization parameters and database selection. Note that both of these tasks occur together in Actions L1.2.1-3. Potential semantic errors are eliminated, because the match between database selection and visualization is enforced.

<i>Cube Visualization</i>	<i>Layer Visualization</i>
<p><b>C1.1 Assumptions:</b></p> <p>User understands that</p> <p>C1.1.1 both a cube and a template are needed for representing geographic data;</p> <p>C1.1.2 database selection is modeled after a hierarchy; and</p> <p>C1.1.3 visual variables make up the symbology.</p>	<p><b>L1.1 Assumptions:</b></p> <p>User understands that</p> <p>L1.1.1 database selection is modeled after a hierarchy; and</p> <p>L1.1.2 visual variables make up the symbology.</p>
<p><b>C1.2 Actions:</b></p> <p>C1.2.1 Choose menu (<b>New Cube...</b>).</p> <p>C1.2.2 Specify database selection (involves several choices and mouse clicks).</p> <p>C1.2.3 Choose menu (<b>New Template...</b>).</p> <p>C1.2.4 Specify visualization parameters (involves several choices and mouse clicks).</p>	<p><b>L1.2 Actions:</b></p> <p>L1.2.1 Choose menu (<b>New Layer...</b>).</p> <p>L1.2.2 Specify database selection (involves several choices and mouse clicks).</p> <p>L1.2.3 Specify visualization parameters (involves several choices and mouse clicks).</p>
<p><b>C1.3 Error Analysis:</b></p> <p>C1.3.1 User might not continue after creating a cube (C1.2.1-2).</p> <p>C1.3.2 User might create a template (C1.2.3) that does not match the cube.</p>	<p><b>L1.3 Error Analysis:</b></p> <p>Semantic errors should not occur.</p>

**Table 1:** Making a new theme (Task 1).

## 6.2 Task 2: Add a Theme to the Current View

The second task for comparison is that of adding a theme to a view (Table 2). In both interfaces, this is done by stacking the proper object(s) onto a platform and viewing the results in its view window. In the cube-and-template interface this is done by placing both the intended data cube (Actions C2.2.1-2) and an appropriate template for rendering that cube onto the viewing platform (Actions C2.2.3-5). The separation of a theme into cubes and templates provides for increased flexibility in choosing visualization parameters for a theme (i.e., one can have several templates for rendering the same cube in different ways), but simultaneously introduces several potential errors that can arise if the user is unsure about how to match proper templates to the correct kinds of cubes (Errors C2.3.1-2). In the layer interface, the user accomplishes this task by simply placing a layer on the intended viewing platform (Actions C2.3.1-2). The two components, visually united into a single object, make the interface conceptually more simple for the user, because only one concept needs to be known (Assumption L2.1.1) vs. two for the cube-and-template interface (Assumptions C2.1.1-2). Potential semantic errors are eliminated, because only a single object needs to be placed on the platform in order to view a theme.

<i>Cube Visualization</i>	<i>Layer Visualization</i>
<p><b>C2.1 Assumptions:</b></p> <p>User understands that</p> <p>C2.1.1 both a cube and a template are needed for representing geographic data; and</p> <p>C2.1.2 the viewing platform is where visualization of data occurs.</p>	<p><b>L2.1 Assumptions:</b></p> <p>User understands that</p> <p>L2.1.1 the viewing platform is where visualization of data occurs.</p>
<p><b>C2.2 Actions:</b></p> <p>C2.2.1 Select cube.</p> <p>C2.2.2 Drag cube onto platform.</p> <p>C2.2.3 Find matching template.</p> <p>C2.2.4 Select template.</p> <p>C2.2.5 Drag template onto platform.</p>	<p><b>L2.2 Actions:</b></p> <p>L2.2.1 Select layer.</p> <p>L2.2.2 Drag layer onto platform.</p>
<p><b>C2.3 Error Analysis:</b></p> <p>C2.3.1 User might add a template that does not match the correct cube (C2.2.3-4).</p> <p>C2.3.2 User might forget to drag a matching template onto the platform (C2.2.3-5).</p>	<p><b>L2.3 Error Analysis:</b></p> <p>Semantic errors should not occur.</p>

**Table 2:** Adding a theme to the current view (Task 2).

### 6.3 Task 3: Remove a Theme from the Current View

The third task for comparison is that of removing a theme from the current view (Table 3). In both interfaces, this is done by removing the proper object(s) from the stack on the viewing platform in order to eliminate them from appearing in its view window. In the cube-and-template interface, the user drags the intended cube away from the platform (Actions C3.2.1-2), identifies the matching template (Action C3.2.3), determines if it is still necessary for visualizing cubes still on the stack (Action C3.2.4), and drags it away if it is no longer needed (Actions C3.2.5-6). The separation of a theme into cubes and templates can cause several semantic errors in this task, i.e., only removing a template causes the contents of a cube to disappear from the view, but leaves an unnecessary cube on the stack (Errors C3.3.1-2). In the layer interface, the user accomplishes this task by simply removing the intended layer from the platform (Actions L3.2.1-2). There is no need to find matching objects for removal, because only a single object needs to be dragged away from the platform in order to remove a theme from its view.

<i>Cube Visualization</i>	<i>Layer Visualization</i>
<p><b>C3.1 Assumptions:</b></p> <p>User understands</p> <p>C3.1.1 the relationship between both components for the purpose of eliminating redundancy in the stack of objects on the viewing platform; and</p> <p>C3.1.2 that the viewing platform is where visualization of data occurs.</p>	<p><b>L3.1 Assumptions:</b></p> <p>User understands that</p> <p>L3.1.1 the viewing platform is where visualization of data occurs.</p>
<p><b>C3.2 Actions:</b></p> <p>C3.2.1 Select cube.</p> <p>C3.2.2 Drag cube away from platform.</p> <p>C3.2.3 Find matching template.</p> <p>C3.2.4 Determine if matching template is still necessary for other cubes.</p> <p>C3.2.5 Select matching template (if not needed).</p> <p>C3.2.6 Drag template away from platform (if not needed).</p>	<p><b>L3.2 Actions:</b></p> <p>L3.2.1 Select layer.</p> <p>L3.2.2 Drag layer away from platform.</p>
<p><b>C3.3 Error Analysis:</b></p> <p>C3.3.1 User might not drag away an unused template leaving an unnecessary element on the platform (C3.2.3-4).</p> <p>C3.3.2 User might mistakenly drag away a template that is still needed by other cubes on the platform (C3.2.5-6).</p>	<p><b>L3.3 Error Analysis:</b></p> <p>Semantic errors should not occur.</p>

**Table 3:** Removing a theme from the current view (Task 3).

#### 6.4 Task 4: Apply a New Symbolization to a Theme

The fourth task for comparison is that of applying a new set of symbolization parameters to a theme (Table 4). In the cube-and-template interface, there are two ways of accomplishing this task, (1) by removing a cube's matching template from the platform and replacing it with one that contains the desired visualization parameters (Actions C4.2.1-6), and (2) by accessing the visualization parameters of a cube's matching template and changing them directly (Actions C4.2.7-9). The separation of a theme into cubes and templates allows users to perform the same task in multiple ways, but can also cause semantic errors if the user misunderstands the relationship between the two components (Errors C4.3.1-3). In the layer interface, the user performs this task by accessing and changing directly the symbolization parameters associated with the intended layer. This is accomplished by double-clicking on the half of the layer icon that corresponds to the visualization (Actions L4.2.1-3). Generally, uniting the two components is

conceptually less straining, because the user interacts with only a single object; however, errors may result if the user misunderstands that the two halves of the layer icon access the two different components (Error L4.3.1).

<i>Cube Visualization</i>	<i>Layer Visualization</i>
<p><b>C4.1 Assumptions:</b></p> <p>User understands</p> <p>C4.1.1 the relationship between both components for the purpose of applying an alternate visualization to the cube or accessing the current visualization by double-clicking; and</p> <p>C4.1.2 that the viewing platform is where visualization of data occurs.</p>	<p><b>L4.1 Assumptions:</b></p> <p>User understands</p> <p>L4.1.1 the separation of data into components which are accessed by double-clicking on the two halves of the layer icon; and</p> <p>L4.1.1 that the viewing platform is where visualization of data occurs.</p>
<p><b>C4.2 Actions:</b></p> <p>C4.2.1 Find matching template.</p> <p>C4.2.2 Select template.</p> <p>C4.2.3 Remove template from platform.</p> <p>C4.2.4 Find (or make) matching template (with desired visualization parameters).</p> <p>C4.2.5 Select template.</p> <p>C4.2.6 Drag template onto platform.</p> <p>C4.2.7 Select template.</p> <p>C4.2.8 Access visualization parameters (by double-clicking).</p> <p>C4.2.9 Choose new visualization (involves several choices and mouse clicks).</p>	<p><b>L4.2 Actions:</b></p> <p>L4.2.1 Select layer.</p> <p>L4.2.2 Access visualization parameters (by double-clicking on correct half of the layer icon).</p> <p>L4.2.3 Choose new visualization (involves several choices and mouse clicks).</p>
<p><b>C4.3 Error Analysis:</b></p> <p>C4.3.1 User might select a template from the platform that does not match the cube, thus erasing a different cube from the view (C4.2.1-2).</p> <p>C4.3.2 User might not add a matching template to the platform (C4.2.4-6).</p> <p>C4.3.3 User might select an incorrect template from the platform for editing, thus changing the visualization of the wrong cube while doing nothing for the cube needing change (C4.2.7-9).</p>	<p><b>L4.3 Error Analysis:</b></p> <p>L4.3.1 User might double-click on the wrong half of the layer icon, thus accessing the database selection mechanism of the theme (L4.2.2).</p>

**Table 4:** Applying a new symbolization to a theme (Task 4).

6.5 Task 5: Apply a New Database Selection to a Theme

The fifth and final task for comparison is that of applying a different database selection to a theme (Table 5). In the cube-and-template interface, there are two ways of accomplishing this task, (1) by removing a cube from the platform and replacing it with one that contains the desired database selection (Actions C5.2.1-4), and (2) by accessing the database selection of a cube and changing it directly (Actions C5.2.5-7). The separation of a theme into cubes and templates allows the user to perform this task in multiple ways, and gives users the ability to easily compare different database selections with the same visualization parameters. In the layer interface, the user performs this task by accessing and changing directly the database selection associated with the intended layer. This is accomplished by double-clicking on the half of the layer icon that corresponds to the database selection (Actions L5.2.1-3). Generally, uniting the two components is conceptually less straining, because the user interacts with only a single object; however, errors may result if the user misunderstands that the two halves of the layer icon access the two different components (Error L5.3.1).

<i>Cube Visualization</i>	<i>Layer Visualization</i>
<p><b>C5.1 Assumptions:</b></p> <p>User understands</p> <p>C5.1.1 the relationship between both components for the purpose of choosing an alternate database selection cube or accessing the current database selection by double-clicking; and</p> <p>C5.1.2 that the viewing platform is where visualization of data occurs.</p>	<p><b>L5.1 Assumptions:</b></p> <p>User understands that</p> <p>L5.1.1 the separation of data into components which are accessed by double-clicking on the two halves of the layer icon; and</p> <p>L5.1.2 the viewing platform is where visualization of data occurs.</p>
<p><b>C5.2 Actions:</b></p> <p>C5.2.1 Select cube.      C5.2.5 Select cube.</p> <p>C5.2.2 Drag cube away from platform.      C5.2.6 Access database selection (by double-clicking).</p> <p>C5.2.3 Find (or make) cube (with desired database selection).      C5.2.7 Choose new database selection (involves several choices and mouse clicks).</p> <p>C5.2.4 Drag cube onto platform.</p>	<p><b>L5.2 Actions:</b></p> <p>L5.2.1 Select layer.</p> <p>L5.2.2 Access database selection (by double-clicking on correct half of the layer icon).</p> <p>L5.2.3 Choose new database selection (involves several choices and mouse clicks).</p>
<p><b>C5.3 Error Analysis:</b></p> <p>C5.3.1 User might forget to drag a new cube onto the platform (C5.2.3-4).</p>	<p><b>L5.3 Error Analysis:</b></p> <p>L5.3.1 User might double-click on the wrong half of the layer icon, thus accessing the visualization parameters of the theme (L5.2.2).</p>

**Table 5:** Applying a new database selection to a theme (Task 5).

## 6.6 Assessment

Evaluating the five task comparisons involves counting the number of assumptions, actions, and potential errors in each visualization. In the cube-and-template visualization, there are a total of 11 points during the interaction where an assumption of some previous knowledge is made, while in the layer visualization there are 8 such points. Some of these assumptions reoccur throughout the five tasks in each visualization. Counting only the non-redundant assumption primitives, one finds 7 for the cube-and-template visualization, whereas the layer visualization has 4.

In terms of necessary user actions, the minimum number required to complete all five tasks in the cube-and-template visualization is 21, whereas the maximum number of actions required is 25. The number of actions that can be taken varies because some of the tasks can be done in multiple ways, each requiring a different number of steps a user has to go through. The number of actions required to complete all five tasks in the Layer visualization is 13.

Counting potential errors provides the greatest discrepancy between the two visualizations. In the cube-and-template visualization there is the potential for a semantic error to occur between 7 and 9 times, depending on which execution was selected for tasks with multiple solutions. In the layer visualization a semantic error is likely to occur at only 2 points during interaction. The potential errors in the cube-and-template interface, while all different, can be thought of as belonging to 3 basic groups: continuation errors, matching errors, and redundancy errors. Continuation errors occur when a user does not add both the necessary cube and template to the viewing platform; matching errors occur when a user adds a template incompatible with the desired cube to the viewing platform; and redundancy errors occur when there are unnecessary objects on the viewing platform. The 2 errors that can result in the layer visualization are also of the same basic type, both resulting from users choosing the wrong half of a layer icon when interacting with one of its components.

The figures resulting from this analysis are summarized in Table 6. They indicate that the five tasks selected to compare the two visualizations are simpler to perform in the layer visualization than in the cube-and-template visualization, as the performance of all three criteria—number of assumptions, necessary actions, and potential errors—is better.

	Cube-and-template visualization	Layer visualization
Assumptions	7 (11)	4 (8)
Actions	21-25	13
Potential Errors	3 (7-9)	1 (2)

**Table 6:** Summary the results of the assessment.

## 7 Conclusions

### 7.1 Summary

This paper presented and compared two visualizations of a user interface based on the map-overlay metaphor. The current state of GIS user interface design was discussed, and the geographer's desktop was introduced as a framework for visualizing a user interface to geographic information. Map-overlay was selected as the metaphor for interaction in the design, and it was suggested that many non-GIS experts who are familiar with its concepts will be able to learn and use such a direct-manipulation user interface easily. Separating database selection and presentation into two interaction components within the framework of map-overlay was shown to be a valid extension of that metaphor which allows for a more flexible, exploratory

interaction. The interface was then given two visualizations, the *cube-and-template visualization* and the *layer visualization*, and these were compared in terms of five important tasks that can be completed in either interface visualization. In each of the five tasks, the interaction with the *Layer Visualization* is significantly less complex, because of the visual (and horizontal) linkage of the database selection and presentation components. The reduced complexity is apparent in the comparison because there are generally less assumptions, fewer necessary steps, and fewer possible errors in the *Layer Visualization* than in the *Cube Visualization*.

## 7.2 Discussion

One might argue that the comparison result is not surprising, favoring the layer user interface over the cube-and-template user interface. The issue of discussion is that one can mix and match cubes and templates, therefore applying a particular symbolization to different data by simply exchanging the cubes. Likewise, applying a different (predefined) symbology may be achieved by replacing one template by another. While the layer interface does not have the same *explicit* operation for exchanging templates, it allows users to perform the corresponding operation through the selection of the symbol set and the definition of user-defined symbol sets through the extended viewing platform. As such, both user interfaces have the same expressive power as they allow users to perform the same operations—in different ways, though.

This raises an important issue in the comparison of different user interfaces, namely whether the two user interfaces allow users to perform the same operations, or whether one can do more in one user interface than in another. If this was the case, there is the danger of comparing apples with oranges, and the user interface that comes out of the evaluation as the more complex one might actually be more complex due to added functionality. Means to assess the comparative expressive power of a user interface may include the writing of an algebra for each user interface, identifying whether or not the same operations can be performed in both user interfaces.

The second question of interest is about complimentary methods to evaluate the difference between two user interfaces. While the proposed method may provide conclusive results if all three evaluation criteria point into the same direction, it will require more interpretation and experience if the criteria give mixed messages. In such situations, adding statistical measures of significance could offer further validity to the methodology. Finally, there remains certainly human-subject testing to confirm results obtained from counting assumptions, actions, and semantic errors. The methodology presented here may be used as the benchmark.

## 7.3 Future Work

This research project has given rise to several important questions, each of which could be a research project or thesis investigation in itself. These questions, and a suggestion for direction of research, are as follows:

*What are other kinds of feedback that can be provided within the framework of map-overlay?* There are apt to be other kinds of semantic feedback that can alert the user about possible semantic errors. For example, when two cubes or layers covering different or partially similar geographic extents are combined, repelling forces could indicate this to users.

*What are additional spatial operations that can be performed within the framework of map-overlay?* Many other operations can be performed by direct manipulation with the objects in each visualization. One example is “edge-matching,” which might be performed by lining up cubes or layers horizontally. An important question to be tackled in solving this problem relates to the differences in the semantics of placing layers side by side instead of on top of one another. It must be clear whether or not the user intends to initiate an operation or if she is merely organizing cubes on the desktop.

*How can computational overlay be included within the framework of map-overlay?* The user interface as designed and visualized thus far addresses only graphic overlay; however, one of the strengths of GIS is its capability to perform computational overlays such as intersections, buffering operations, etc. The question of whether or not to extend the map-overlay visualization to include computational overlay is open, as there may be other more natural visualizations. One viable solution is to combine the map-overlay metaphor with a visualization similar to elementary school addition, which is a source metaphor that virtually everyone is familiar with (Egenhofer and Richards, 1993b; Egenhofer and Bruns, 1994).

*How does the user interact with the database selection and presentation components?* Two previous works have presented formalizations of interaction with the two components. Volta and Egenhofer (1993) presented a formalization of interaction with the database selection component in terms of manipulating attribute information via categorical coverages; while Egenhofer (1991) has addressed the presentation component in terms of a Graphical Presentation Language (GPL) that compliments a Spatial SQL dialect. While these works cover the formalization issues involved with the two components, there is still the need for visualizing these concepts at the user interface.

*How can the map-overlay metaphor be effectively combined with other metaphors?* Map-overlay is not the only metaphor available and in use for applications in GIS and other spatial problems. For example, Jackson (1990) presented a family of metaphors for visualizing panning and zooming in geographic space. Investigation into how these and other metaphors can co-exist seamlessly in a GIS would be valuable research.

## **8 Acknowledgments**

A number of our colleagues have contributed considerably to this project. Thanks to all of them. Particularly, Andrew Frank participated in the development of the concepts for the cube-and-template user interface, Todd Rowell worked on the implementation of the prototypes, Mike Gould provided valuable feedback, and Kathleen Hornsby helped with the preparation of the manuscript.

## **9 References**

Apple (1987) *Human Interface Guidelines: The Apple Desktop Interface*. Addison-Wesley Publishing Company, Reading, MA.

Card, S., Moran, T., and Newell, A. (1980) The Keystroke-Level Model for User Performance Time with Interactive Systems. *Communications of the ACM* 23(7): 396-410.

Chan, K., and White, D. (1987) Map Algebra: An Object-Oriented Implementation. In: R. Aangeenbrug, and Y. Schiffman, editors, *Proceedings of International Geographic Information Systems (IGIS) Symposium: The Research Agenda*, pp. 127-150, Association of American Geographers, Washington, D.C.

Egenhofer, M. (1991) Extending SQL for Cartographic Display. *Cartography and Geographic Information Systems* 18(4): 230-245.

Egenhofer, M. and Bruns, T. (1994) Getting Maps onto the Desktop: GIS User Interfaces Beyond Menus and Buttons. *International IGUG NEWS*, Fall issue, pp. 8-9.

Egenhofer, M. and Richards, J. (1993a) Exploratory Access to Geographic Data Based on the Map-Overlay Metaphor. *Journal of Visual Languages and Computing* 4(2): 105-125.

Egenhofer, M. and Richards, J. (1993b) The Geographer's Desktop: A Direct-Manipulation User Interface for Map Overlay. In: R. McMaster and M. Armstrong, editors, *Autocarto 11*, Minneapolis, MN, pp. 63-71.

Frank, A. (1982) MAPQUERY—Database Query Language for Retrieval of Geometric Data and its Graphical Representation. *ACM Computer Graphics* 16(3): 199-207.

Frank, A. (1992) Beyond Query Languages for Geographic Databases: Data Cubes and Maps. In: G. Gambosi, M. Scholl, and H.-W. Six, editors, *Geographic Database Management Systems. Esprit Basic Research Series* pp. 5-17, Springer-Verlag, New York, NY.

Gould, M. and McGranaghan, M. (1990) Metaphor in Geographic Information Systems. In: K. Brassel and H. Kishimoto, editors, *Fourth International Symposium on Spatial Data Handling*, Zurich, Switzerland, pp. 433-442.

Haunold, P. and Kuhn, W. (1993) A Keystroke Level Analysis of Manual Map Digitizing. In: A. Frank and I. Campari, editors, *Spatial Information Theory, European Conference, COSIT '93. Lecture Notes in Computer Science* 716, pp. 406-420, Springer-Verlag, New York, NY.

Herring, J., Larsen, R., and Shivakumar, J. (1988) Extensions to the SQL Language to Support Spatial Analysis in a Topological Data Base. In: *GIS/LIS '88*, San Antonio, TX, pp. 741-750.

Ingram, K. and Phillips, W. (1987) Geographic Information Processing Using a SQL-Based Query Language. In: N. R. Chrisman, editor, *AUTO-CARTO 8, Eighth International Symposium on Computer-Assisted Cartography*, Baltimore, MD, pp. 326-335.

Jackson, J. (1990) Developing an Effective Human Interface for Geographic Information Systems Using Metaphors. In: *ACSM-ASPRS Annual Convention*, Denver, CO, pp. 117-125.

Khoshafian, S. and Abnous, R. (1990) *Object Orientation: Concepts, Languages, Databases, User Interfaces*. John Wiley & Sons, New York, NY.

Kirby, K. C. and Pazner, M. (1990) Graphic Map Algebra. In: K. Brassel and H. Kishimoto, editors, *Fourth International Symposium on Spatial Data Handling*, Zurich, Switzerland, pp. 413-422.

Kuhn, W. (1992) Paradigms of GIS Use. In: D. Cowen, editor, *Fifth International Symposium on Spatial Data Handling*, Charleston, SC, pp. 91-103.

Kuhn, W. (1993) Metaphors Create Theories for Users. In: A. Frank and I. Campari, editors, *Spatial Information Theory, European Conference COSIT '93, Marciana Marina, Elba Island, Italy*. 716, pp. 366-376, Springer-Verlag, New York, NY.

Kuhn, W. and Egenhofer, M. (1991) CHI '90 Workshop on Visual Interfaces to Geometry. *SIGCHI Bulletin* 23(2): 46-55.

Kuhn, W. and Frank, A. (1991) A Formalization of Metaphors and Image-Schemas in User Interfaces. In: D. Mark and A. Frank, D. Mark and A. Frank, *Cognitive and Linguistic Aspects of Geographic Space*. pp. 419-434, Kluwer Academic Publishers, Dordrecht.

Lewis, C. (1991) Cognitive Walkthrough Analysis of a GIS. In: W. Kuhn and M. Egenhofer, W. Kuhn and M. Egenhofer, *Interfaces to Geometry. NCGIA Technical Papers* 91-18, pp. National Center for Geographic Information and Analysis, Santa Barbara.

Lewis, C., Polson, P., Wharton, C., and Rieman, J. (1990) Testing a Walkthrough Methodology for Theory-Based Design of Walk-Up-and-Use Interfaces. In: *CHI '90*, Seattle, WA, pp. 235-242.

- Linsey, T. and Raper, J. (1993) HyperArc: A Task-Oriented Hypertext GIS Interface. *International Journal of Geographical Information Systems* 7(5): 435-452.
- Mark, D. and Frank, A. (1992) Research Initiative 13 Report on the Specialist Meeting: User Interfaces for Geographic Information Systems. National Center for Geographic Information and Analysis, University of California, Santa Barbara, Technical Report 92-3.
- Mark, D. and Gould, M. (1991) Interaction with Geographic Information: A Commentary. *Photogrammetric Engineering & Remote Sensing* 57(11): 1427-1430.
- McCormick, E. (1979) *Job Analysis: Methods and Applications*. American Management Association, New York.
- Medyckyj-Scott, D. and Hearnshaw, H. (1993) *Human Factors in Geographical Information Systems*. Belhaven Press, London.
- Nielsen, J. (1993) Iterative User-Interface Design. *Computer* 26 (11), 32-41.
- Pazner, M., Kirby, K. C., and Thies, N. (1989) *MAP II: Map Processor—A Geographic Information System for the Macintosh*. John Wiley & Sons, New York, NY.
- Phillips, M., Bashinski, H., Ammerman, H., and Fligg, C. (1988) A Task Analytic Approach to Dialogue Design. In: M. Helander, editor, *Handbook of Human-Computer Interaction*. pp. 835-857, Elsevier, North-Holland, Amsterdam.
- Polson, P., Lewis, C., Rieman, J., and Wharton, C. (1992) Cognitive Walkthroughs: A Method for Theory-Based Evaluation of User Interfaces. *International Journal of Man-Machine Studies* 36(5): 741-773.
- Tomlin, C. D. (1990) *Geographic Information Systems and Cartographic Modeling*. Prentice-Hall, Englewood Cliffs, NJ.
- Volta, G. and Egenhofer, M. (1993) Interaction with GIS Attribute Data Based on Categorical Coverages. In: A. Frank and I. Campari, editors, *Spatial Information Theory, European Conference COSIT '93, Marciana Marina, Elba Island, Italy. Lecture Notes in Computer Science* 716, pp. 215-233, Springer-Verlag, New York, NY.