Fundamental Research in Geographic Information and Analysis


University of California,
Santa Barbara

State University of New York
at Buffalo

University of Maine

Funded by the National Science Foundation

CD produced with support from Environmental Systems Research Institute, Inc.

Copyright © 1988–1997, Regents, University of California
Intelligent Assistants for Filling Critical Gaps in GIS

A Research Program

April 1992

David Lanter

NCGIA
Department of Geography
University of California at Santa Barbara
Santa Barbara CA 93106-4060

National Center for Geographic Information and Analysis
Report 92-4
Contents

OVERVIEW

1. GEOGRAPHIC INFORMATION SYSTEMS
   GIS Functions

2. CRITICAL GAPS IN CURRENT GIS
   Spatial Data Lineage Tracking
   Difficulty of Use
   Update Handling
   Accuracy and Error Propagation
   Quality Control
   Data Format Translation
   Database Optimization

3. RESEARCH AGENDA FOR FILLING THE GAPS
   User is in command - Intelligent Assistant not used
   Intelligent Assistant as monitor
   Intelligent Assistant as Supervisor
   Assistant in Command - User is Supervisor
   Assistant in Command - User is monitor
   Assistant in Command - User not consulted

4. EMERGING TECHNOLOGIES
   Skills-based Behavior
   Rule-based Behavior
   Knowledge-based Behavior

APPENDIX: RULE-BASED SYSTEMS

BIBLIOGRAPHY
OVERVIEW

This research report, *Intelligent Assistants for Filling Critical Gaps In GIS*, was sponsored by Southern California Edison Company (SCE), Geographic Designs Inc., and the NCGIA to identify a research program that provides:

- An analysis of critical gaps in current geographic information systems (GIS) that impede their use for spatial decision support.
- A research agenda adapting expert system and other related technologies to fill these gaps.
- Discussion of emerging technologies for supporting spatial decision makers.

Geographic Information System (GIS) technology provides powerful software tools for analyzing maps, charts, and other spatial data. It is differentiated from traditional Data Base Management Systems (DBMS) and Computer Aided Design (CAD) systems. This differentiation is a result of its unique ability to manipulate existing spatial data and "synthesize" new spatial entities that are explicit representations of spatial relations previously only implicit in the database. Chapter 1 gives an overview of GIS technology.

GISs equipped with powerful tools for spatial analysis have been available for two decades, yet they still lack ways of recording the most basic data quality information and viewing the most fundamental data dependencies and lineage relationships between maps stored in their databases. This is because GISs are poor in capabilities for handling information about spatial analysis and the new information that results from it. This results in communication and coordination problems between GIS staff members and subsequently a significant drop in productivity. Chapter 2 details a series of critical gaps resulting from these problems.

Mr. Jack Kawashirna of SCE believed that intelligent computer systems could play a number of roles required to fill the gaps resulting in lost productivity. In his vision he saw increased levels of automation leading to higher GIS productivity resulting from a sharing of responsibility between human operators and Intelligent Assistant computer systems. The research agenda presented in Chapter 3 focuses on applying artificial intelligence techniques of expert and related systems to create Intelligent Assistant systems to support the GIS user with capabilities for coping with the critical gaps.

Six principal types of support roles are proposed for Intelligent Assistants. These range from providing "no assistance" in using the GIS (the current situation), to "complete assistance" where the needs of decision makers are met automatically without intervention by a human GIS user. In order to design intelligent computer systems to play the different roles existing between the extremes it will be necessary to take advantage of emerging expert systems, knowledge acquisition, and case-based reasoning technologies. These technologies are discussed in Chapter 4.

1. GEOGRAPHIC INFORMATION SYSTEMS

A Geographic Information System (GIS) is computer software designed to collect, store, retrieve, manipulate, and display cartographic features defined as points, lines, or areas (Tomlinson et al. 1976; Marble 1984; Clarke 1986; and Dueker 1987). The GIS's unique ability to manipulate existing data and "synthesize" new spatial entities separates it from other information processing technologies (Burrough 1986; Cowen 1988). This is based on software functions for transforming and combining cartographic data acquired from maps of differing spatial coordinate and projection systems. Once the geographic features are placed in common geographic coordinate and projection systems they can be organized in the GIS database.

GIS data is typically organized in one of three ways: (1) Least Common Geographical Unit (LCGU), (2) objects, and (3) layers. Least Common Geographical Units, also known as Integrated Terrain Units (ITUs) combine and integrate all spatial data into a single spatial data record containing all feature and attribute classes (Chrisman 1975). Object-oriented organizations integrate individual geographic features and their attributes into semantically defined objects with inheritable properties (Kjerne and Dueker 1986; Charlwood et al. 1987; Herring 1987; Gahegan and Roberts 1988; Egenhofer and Frank 1989).

LCGU and object-oriented approaches represent a very small minority of systems'. However, they are finding favor among their users. Layer-based systems represent the most widely used and understood GIS. This is because the organization of data features into thematic layers closely corresponds with the way geographic data has been handled for centuries by cartographers. Layer-based GIS databases allow the users to visualize the cartographic database as a series of registered map separations. Spatial relationships between relevant geographic entities can be analyzed, in such systems, with the help of neighborhood and overlay functions (Chrisman and Niemann 1985; Kjerne and Dueker 1986; Aronson 1987; Bracken and Webster 1989). Examples of layer-based GIS include Environmental Science Research Institute's vector-based ARC/INFO, Tomlin's raster-based Map Analysis Package (Tomlin 1980), and Intera Tydac's quadtree based SPANS.
In any of the above organizational approaches, geographic features have characteristics that exist in three dimensions: (1) the spatial dimension captures the location of the phenomena, (2) the aspatial dimension captures descriptive attributes about the phenomena, and (3) the temporal dimension captures the time and duration of the phenomena (Berry 1964).

**The Spatial Dimension**

Vector and tessellation models offer two alternative successful approaches for storing and processing spatial data in GIS (Peucker and Chrisman 1975; Peuquet 1984; Maffini 1987). Vector approaches represent graphical aspects of geographic features as points, lines, and polygons with strings of coordinate X,Y pairs. The evolution of vector data models has witnessed an increase in the amount of spatial information explicitly represented in the point, line, and polygon data structures. This evolution has moved from a simple spaghetti data structure, where no information about the entities is encoded in the data structure, to the entity-by-entity, point dictionary, and chain-code structures encoding single polygons without regard for neighboring spaces. To address the concern for neighborhoods the topological vector data structure was introduced with its evolution from GBF/DIME, to POLVRT structures.

In contrast to vector data models, tessellation models organize point, line, and polygonal entities within a mosaic of geometric tiles. Each tile representing a homogenous region in space characterized by the absence or presence of some geographical entity. The geometry of the tile may be regular squares, triangles, or hexagons which may be nested in hierarchies subdividing large non-homogeneous areas (e.g. quad-trees). Of the three, square (raster and grid cell) tessellations are the most popular spatial data organization in non-vector GIS.

Peuquet (1984) discusses the strengths and weaknesses of the vector and raster models. She concludes that "neither type of data model is intrinsically a better representation of space. The representational and algorithmic advantages of each are data and application dependent, even though both theoretically have the capability to accommodate any type of data or procedure."

**Aspatial Dimension**

Spatial data are linked with their aspatial attributes to describe relevant characteristics of geographic features. Separating aspatial attributes from the graphic representation permits retrieval and display of geographic features meeting a combination of spatial and thematic query criteria. Aspatial characteristics such as type, material, names, etc. can be stored in either the hierarchical, network, or relational data models.

The hierarchical model organizes data in a tree structure. It is made of links connecting nodes in branches. Each node represents an entity and consists of collections of descriptive attributes. The oldest data base systems (DBMSs) are based on hierarchical data organization. Examples include, IBM's IMS, Intel's System 2000, and Informatic's Mark IV. Hierarchical systems are built on rigid tree structures that are not easily reorganized and promote redundant encoding and storage of attributes describing features stored in different nodes within different branches of the tree. They do, however, provide for rapid partitioning of data sets and retrieval of stored data.

The network data model, in contrast, organizes attribute data as nodes interconnected by a network of directed labelled links. Examples of commercial network DBMS's include Cullinane's IDMS, Cincom's TOTAL, Honeywell's IDS/II, UNIVAC's 1100, and DEC’s DBMS-20. Network-based systems are more powerful than hierarchical systems in modelling real world data relationships without redundancy. This power, however, is balanced by navigation and update complexity resulting in a detailed understanding of the complex web of nodes and pointers as a prerequisite of data retrieval.

The relational data model uses tables to organize attributes of entities and relationships between geographic features. The use of tables to organize feature attributes makes the relational data model easy to understand focusing attention on the database's information content. The hierarchical and network models, in contrast, force users to pay attention to the physical storage of data in order to access its information content. The relational model is also preferable in that it supports database normalization to remove redundant information. In addition, the relational model is based on a solid mathematical theory of relations not present for the other data models.

Power associated with ease of understanding, database normalization, and mathematical basis is offset by the high cost in computational overhead that threatens to slow down the processing of large databases. Most commercial GIS's however use a relational DBMS to provide their users with flexible organizational and query capabilities. Examples of relational DBMSs used with GIS are Henco's INFO, Relational Technologies' INGRES, and Oracle Corporation's ORACLE.
Temporal Dimension

Langran (1988) suggests that one goal of GIS research should be focused on enabling the analysis of change in spatial information over time. A temporal GIS, therefore, would store information about the changing nature of features as they occur outside the GIS database in the "real world". This information is important to producers and consumers of map products requiring currency and historical accountability (Hunter 1988; Vrana 1989). Langran and Chrisman (1988) point out that a temporal data structure should retain the elements of temporal topology, that is they should provide the means of navigating from one state or version of a feature to another through time.

Armstrong’s (1988) conceptual model creates a time topology that is based on the use of time/date stamps to tag each feature in the GIS database with temporal information. Tagging each feature in this way will facilitate tracking how the feature changes across many dates. With a time attribute it is possible to retrieve temporal information about a given feature, or retrieve particular features associated with a specific time. As far back as 1977, Basoglu and Morrison (1977) demonstrated how a system of feature-based time stamping was implemented. Their work demonstrated how various time slices could be selected to view the temporally changing configurations of historical county boundaries.

This work and that of Armstrong and Langran and Chrisman illustrate how the GIS can be designed to support spatio-temporal queries. A GIS designed to analyze historically changing geographic states could provide information necessary to detecting patterns, trends, and cycles in spatio-temporal data. This could help explain underlying processes and serve as a basis for projecting the future.

Once the geographic data is organized and stored in the GIS database, analytic spatial neighborhood operators can be applied to analyze the nature of proximity, connectivity, and adjacency relationships between stored features. Spatial statistics can be applied to describe spatial variation in data in a more refined way than typically done in traditional statistical treatments that assume a uniform distribution over space (Anselin and Rey 1991). In addition, algebraic and boolean operators can be applied to interrelate both the original and derived data layers. Berry (1991) refers to the resulting applications as 'map-ematics'. Goodchild refers to this kind of data processing as spatial analysis (Goodchild 1987). Regardless of name, a GIS operator is applied to make explicit point, line, or polygon feature entities that encode spatial relations previously implicit in the spatial, aspatial thematic, and temporal dimensions of information stored within the GIS's database. For example, a buffer operation may be applied to locations of power poles to determine areas which have ready access. This newly derived data could be overlaid with a boolean 'AND' on a rural road layer. The result would be a database representation of all power poles within easy access to rural roads.

GIS Functions

Taxonomies of GIS functions are developed in a variety of contexts. Functions are organized and classified in textbooks, systems that employ menu-based user interfaces, system manuals, and reviews of existing GIS functionalities and comparisons of systems. In an optimal sense, a taxonomy of GIS function should incorporate a wide range of system functionality (i.e. a high level of scope), define this functionality with sufficient detail to make the taxonomy applicable in an operational context (i.e. high level of detail), organize functions into meaningful and logical high-order categories (i.e. high level of structure), and avoid contradictions and ambiguities (i.e., high level of internal consistency).

The taxonomy presented here attempts to organize these functions conceptually -independent of any existing system. The intent here is to provide an extensible, consistent and logical framework that allows for the incorporation of functions as they are implemented within a particular system. This also facilitates an evolving definition of GIS functionality as new functions are developed and as old functions are modified. The taxonomy avoids separating functions according to data structure. The vector-raster distinction is relevant at the level of function implementation, since different algorithms are required to implement the same function in vector and raster contexts. From a conceptual point of view, however, this distinction is less important since data structure is not important in arriving at a conceptual understanding of the purpose or meaning of a particular function, nor of the resulting cartographic data.

GIS functions can be placed in three groups: input, manipulation, and output. The taxonomy outlined below is structured using the three groups and heavily emphasizes the manipulation functions.

Input Functions

Input functions are divided into three types: functions that compile data, functions that restructure data, and functions that edit data. Compile functions encode geographical data in a form that is usable by the system. These functions include digitizing, scanning, entry of aspatial attributes, feature labeling, encoding of survey documents, building topology, and edgematching of map tiles. Restructure functions modify the structure of an imported data base so that it can be used by the system. Restructuring may
entails simple reformatting of data, conversion between variants of the same data structure (e.g. Morton sequencing to run-length encoding), or conversion between different data structures (e.g. raster to vector). Edit functions detect and resolve topological, spatial, and aspatial inconsistencies and focus on removing errors before they enter the database. These functions include sliver polygon removal, line snapping, attribute editing, and the ability to insert geometric features.

**Manipulation functions**

Manipulate functions are divided into four groups: select, describe, transform, and derive.

**Select** functions use Boolean logic to return features based on a set of selection criteria. No new data are created, nor are existing data transformed in any way. Features may be selected by spatial attributes or by aspatial (thematic) attributes. Selection of features based on their spatial attributes can be further divided into selection based on spatial association, feature inclusion, and address range. Selection by feature association returns features based on topological notions of contiguity or connectivity. For example, one might select all polygons which border a specified polygon. Selection of features based on spatial inclusion returns features that are contained within a user-defined or selected feature. For example, the user may define a polygonal window in order to return all points contained within it, or the user may select a point which returns a polygon. Features may also be selected by address range. Selection of features based on address range returns features located at or between user-specified street addresses which are transformed into map coordinates by the system. Selection by aspatial attributes uses Boolean logic to return features with desired thematic and/or temporal attributes such as class, name, or code. For instance, one might select all polygons with a land use attribute value of "agriculture", or lines that are stream with an order attribute of 2.

**Describe** functions compute indices describing the geometric properties of features. These functions compute indices of shape, size, or angle. Indices of shape are numeric measurements describing the shapes of features. Indices of shape include: convexity, fractal dimensionality, polygon integrity and line crenularity. Size indices describe the relative magnitude of features. These size indexing functions may be divided into functions which measure length, area, and volume. Length indexing functions compute the lengths of features. Types of length measurement include straight-line distance, curved-line distance, polygon perimeter, great circle distance and network distance (e.g. shortest distance). Area indexing functions compute the area of polygon features on a flat or curved surface. Volume indexing functions compute the volume under a surface or part of a surface. Angle indexing functions describe properties of features that are measured with angles, such as azimuth, aspect and bearing.

**Transform** functions modify the spatial or aspatial attributes of features. These functions are further divided into sub-categories "arithmetic" and "categorize" which modify aspatial attributes of features and "geometric" which modifies spatial attributes of features. The arithmetic transformations can be divided into those involving one map and those involving two maps. Arithmetic transformations involving one map may apply a constant to modify an attribute. For example, a new layer may be created by multiplying the old layer by a scalar. Other operations include adding, subtracting, and dividing a layer by this constant. They may modify an attribute by applying a trigonometric, logarithmic, or exponential function. In addition to these unary operations, there are functions which apply an arithmetic operation to combine two or more attributes and re-express them as a single attribute. These multi-attribute arithmetic transformations include arithmetic overlay. Arithmetic overlay can be used to apply such operators as add, subtract, multiply, or divided to create algebraic equations where each map is treated as a variable. The result is a new layer with the derived value.

**Categorize** functions modify the aspatial attributes of features by reclassifying them into new categories or groups. Categorize transformations may be divided into those which operate on single map layers or attributes and those which operate on multiple layers or attributes. Classify functions map quantitative attributes into nominal categories. Examples of classify functions are classification of remotely sensed data, choropleth mapping and stream ordering. Reclassify functions map nominal categories into other nominal categories or numerical categories expressing weights. This includes aggregation of classes, for example; aggregation of landuse use classes "residential" and "industrial" into a class called "urban".

Geometric transformations modify the spatial attributes of features. These geometric transformations can be broken into three categories. **Projection functions** change distortion patterns resulting from placing a spherical world onto a flat surface based on accepted formulae. This type of geometric transformation has great certainty, and is reversible. A few examples of projection change include UTM to latitude/longitude, Lambert conformal to State plane, and Albers to Mercator. A linear transformation applies a general linear transformation of the plane. These adjust the spatial dimension of the feature contents of an entire map and include changing scale, rotation, translation. A non-linear transformation applies a non-linear transformation for distortion removal. This is often referred to as "rubber sheeting". Non-linear transformations are the least certain type of geometric transformation. They are user-defined, and they are not applied equally to each part of the map.

**Derive** functions extract new features based on inferences about existing features. Derive geometric elements extract standard geometric elements from spatial data. For example, polygons can be reduced to centroid or polygon skeletons or minimum
bounding rectangles. Thiessen polygons delineating areas closest to particular centroids can be derived from point data. Delaunay triangles can be derived from Thiessen polygons. Generalize functions simplify features by abstracting their salient spatial characteristics. There are three general categories of generalize functions. A line simplification function simplifies a linear feature by reducing ("weeding") the number of points used to devine its shape. This is useful as a data reduction technique. A line smoothing function fits a Bezier, spline, or some other curve function to remove minor forms from a line. Complex generalization functions perform generalization involving one or more of the following: a change in feature type, aggregation of polygons (including dropping the line between adjacent polygons with the same aspatial and temporal attributes), and conflict detection and resolution in feature locations (usually involving displacement).

Buffer functions extract features that spatially encompass an existing feature based on some application-specific criteria. Buffers may be square or curved, may surround point, line, or polygon (interior or exterior features).

Overlays are functions that extract new features by applying boolean logic. Boolean logic includes, and, or, xor, and not operators and their combinations. This logic can be applied on cartographic data type pairs including: points in polygons, points on lines, points on points, lines on points, polygons on polygons, etc. Surface derivation involves the extraction of 3 dimensional features. Interpolation functions interpolate between representations of surfaces. Resulting surfaces may be represented by lattices, random points, grids, contours, TINs, etc. A filtering function applies a neighborhood function to a surface. High-pass and Low-pass filters are commonly used in image processing of remotely sensed data to accentuate or smooth sharp boundaries. Other filters are used to determine slope and aspect of a section of a map, locate drainage features, or identify visibility regions.

Output Functions

Output functions are the collection of functions which, in effect, remove a map or other information from the GIS database for use in decision making or other systems. Graphic design functions are used to symbolize and annotate geographical data to facilitate map composition and cartographic communication. Visualize functions graphically represent geographical data using hardcopy output devices such as a graphics terminal. Restructure functions change the structure of the database for export to another system. Restructuring may entail simple reformatting of data, conversion between variants of the same data structure such as Morton sequencing to run-length encoding, or conversion between different data structures as in raster to vector conversion. Summarize functions compute indices summarizing the spatial and aspatial properties of features for the purpose of report generation. These indices may be descriptive (including both counts and averages), multivariate or inferential statistics. They may also be indices describing spatial pattern. For example, points may be clustered, spaced evenly, or spaced randomly.

2. CRITICAL GAPS IN CURRENT GIS

Managing a large dynamic spatial analytic database can lead to great frustration. After two decades of development, Geographic Information Systems still lack ways of recording the most basic data quality information or viewing the most fundamental data dependencies and lineage relationships between maps stored in the database. This is because GISs are poor in tools for handling information about spatial analysis and information that results from it. One of the main ways of recording information about what's in a database is through severely abbreviated database file names on the computer operating system. Often the result is GIS terminals and workstations covered in Post-It notes as users attempt to communicate to themselves and each other what various data layers represent. Brooks (1975) attributes such problems of communication between staff members as responsible for significant reductions in productivity. The more people involved in updating and deriving new GIS data, the more time and effort that must be spent communicating about the contents of the database. Three-user communications involve three channels of communication, whereas, four-user communications involve six channels:
As the number of people effecting change to the database increases the number of communication channels increase too, but at a faster rate. The more users sharing the database, the more time is required for each user to communicate with the rest. This results in less time available for update control, content management, as well as design, implementation, and distribution of the results of GIS applications. Unfortunately, most GIS databases are too large and expensive a resource to be turned over to one user to use in a vacuum. The alternative is sharing the care and use of the database among many users while simultaneously providing them with ways to reduce time spent communicating. The saved time is then freed for updating the database, managing its contents and designing, implementing, and debugging GIS applications.

Babich (1986) discusses the problems associated with large software development teams and makes the point that group productivity loss may not be as much of a communication problem as it is a coordination problem. That is, more time is lost as the result of redundant efforts and mistakes made as the result of users failing to coordinate with one another.

**Spatial Data Lineage Tracking**

Transforming existing data into new data is a double-edged sword. The resulting database may have more information than the old one. The meaning of the new information, however, is exogenous and not found in the data (Tobler 1979). To address the problem of interpreting the meaning of derived spatial data the National Committee for Digital Cartographic Data Standards (NCDCDS) has proposed that exchanged cartographic data be accompanied by a ‘lineage’ statement within a formal quality report (Morrison 1988). The committee’s definition of lineage is information describing source materials and transformations used to derive final digital cartographic data files (Chrisman 1983a; Morrison 1988). The lineage report is intended to serve as a communication mechanism between data producers and users. It is a “truth in labeling” approach to cartographic data quality reporting (Moellering 1987). The producer’s responsibility is to label the GIS derived data product with information concerning its source materials and processing history. The user’s responsibility is to interpret such information and determine the data’s fitness for use (Chrisman 1983; Grady 1988).

A number of automated systems including history files, version control systems, map librarians, and polygon attributes are examined and none have been found capable of fulfilling the informational requirements of a lineage report. A system that would meet such requirements would detail the characteristics of cartographic sources, the topological relationships between source, intermediate and product layers, and the transformations applied to sources to derive GIS applications output products.

**Temporal Topology**

Langran and Chrisman differentiate between two temporal axes relating to cartographic change (Langran and Chrisman 1988). ‘World time’ tracks changes to geographic features occurring on the surface of the Earth. Tagging each feature with a time-stamp marks time on this axis and facilitates tracking it as it changes over many dates (Basoglu and Morrison 1977; Armstrong 1988; Langran 1988; Langran, 1989). The world time axis is important to producers of map products having strict requirements for currency and historical accountability (Hunter 1988; Vrana 1989).

'Database time’ differs from ‘world time’, according to Langran and Chrisman, in that its concern is for tracking entry of spatial entities into a geographic database (Langran and Chrisman 1988). This begins, they argue, when the first data is entered into the database and ends on the last data entry. Langran and Chrisman’s, definition of database time, however, cuts short the length of the database time axis. From the view of a GIS, database time extends beyond the phase of data collection and storage and continues through phases of retrieval, manipulation, and analysis. It marks not only the changing nature of geographic entities as they are stored within a database, but also the derivation of new ones as they are derived by geographic information processing. Lengthening the 'database time' axis will make it useful for structuring transformations recorded within lineage documentation of a GIS application.

Langran and Chrisman suggest that topologies are useful not only for organizing spatial data, but for structuring the cartographic time axes to track changing cartographic features in a geographic database (Langran and Chrisman 1988). They instruct,

> ..just as the spatial topological data structure would provide a means of navigating from an object to its neighbor in space, the corresponding temporal data structure would provide a means of navigating from a state or a version to its neighbor in time. (p. 7)

**Cartographic Model**

Tomlin's cartographic model (Tomlin 1990) is useful for focusing attention on such topologies taking place within GIS applications. The cartographic model’s diagrammatic specification explicitly records the flow of maps through transformations applied throughout the course of geographic information processing (Tomlin and Berry 1979; Tomlin 1983). The model records the sequence
of transformations thematic layers undergo from source materials through intermediate processes to final products. It is a special form of data flow diagram depicting layers as nodes and transformations as labeled directed links. The resulting network representation specifies an application's processes and data interfaces between them (Martin and McClure 1985). Its graphical representation expresses the morphology of a GIS application.

Recording processing steps applied to geographic data in a cartographic model provides a powerful mechanism for documenting input/output relationships inherent in a GIS application. Examining the cartographic model reveals the entire history of transformations applied to source materials throughout the course of a GIS application. It captures two of three important aspects of lineage information: input/output relationships and transformations. This information when combined with descriptions of source materials is lineage documentation.

The GIS Application's Structure

Tobler suggests, "a geographic structure is a transformation of geographic data, a theory, or model of reality" (Tobler 1979, 105). Lineage information is such a geographic structure. It documents source data, transformations, and input/output specifications illustrating the derivation of cartographic products within a GIS application. This structure is a major factor in the communication of the meaning of the derived map.

Analysis of a derived map's lineage provides understanding of the geographic reality it represents. It also makes obvious when a GIS application is based on flawed logic. This is important. If the theory encoded in a GIS application is not clearly understood, then a correct interpretation of the resulting map is not possible. Recording this information along with documenting sources has critical relevance to decision makers using information derived within GIS applications.

By representing structure, lineage documentation transforms the GIS application into material reality. It expresses the nature of source data, input/output relationships, and transformations. Documentation of sources and sequences of transformations expressed in terms of commands and command modifier values has the potential of simultaneously reflecting and reinforcing the world view and spatial thought of the institution applying them to analyze digital representations of the world. As a result, lineage-documentation provides the material necessary to study the institutional context of GIS-based decision making.

Automating Lineage Information

Lineage documentation is important to interpreting the nature and quality of GIS derived map products. However, it is used rarely if at all. Vonderohe and Chrisman (1989) report that of five interrelated spatial data quality components: lineage, positional accuracy, attribute accuracy, logical completeness and currency, lineage is the only one that is "not testable" in the course of spatial data processing (Vonderohe and Chrisman 1985). This is because no technique for automatically creating and manipulating lineage information has been suggested in the geographic literature. The sections that follow review existing aspatial and spatial methods for tracking lineage in various information systems.

Lineage in Aspatial Systems

Systems for documenting and tracking changes to data files within an information system are common in computer environments. These methods range from simple history files, log files and audit files to more complicated schemes of tracking updates to programmed source code and text documents.

• History Files. The most basic form of documenting changes to a data file is through use of a "history file". This capability is often part of a computer's operating system and provides a way for the system administrator to keep track of commands given to the system by users. An examination of a history file will yield commands used and subsequent changes made to various files by users. An example of such a history can be found in the UNIX operating system (McGilton and Morgan 1983).

• Audit and Log Files. Variations on the history approach include the 'audit' and 'log' files. These systems provide detailed listings of user commands and system output. Audit or 'watch' files typically record the entry and system responses for each given command. In addition, log files record date, elapsed time, and amount of system resources used during execution of each command. Examples of such facilities can be found in the ERDAS and ARC/INFO systems.

Each of the history, audit, and log files has serious shortcomings related to lineage documentation. They do not document the nature of data sources input into the GIS database. Nor do they provide any connectivity to specific data layers within the database. As a result, many pages of output or CRT screens of data must be reviewed to determine what transformations were made to specific maps by user entered commands. While they are able to capture commands as typed by users, these facilities are unable to explicitly represent topological input/output relationships between sources and derived spatial data products. Due to the dynamic nature of
spatial information systems such manual approaches to determining all the source data and transformations resulting in a given map are inadequate.

- **Version Control Systems.** Version Control Systems track revisions made to documents and program source code. They are software engineering tools often supplied along with hardware by computer manufacturers such as Control Data Corporation, IBM, SUN, Univac, and others. One example is the Source Control and Configuration System (SCCS) originally part of the Berkeley UNIX operating system. SCCS is a library of operating system functions enabling users to maintain a file of revisions to source code documents created in the course of software development (Allman 1984).

  The SCCS stores the initial document and file of changes (‘deltas’) to that document associated with each revision. The delta is the set of differences between each revision and the previous revision. Each delta is accompanied by a time and date-stamp and an optional user comment. Using this file of deltas, a user can start with an initial document and move forward in time recreating any subsequent version. Alternatively, other version control systems store the complete current versions of a document together with a series of deltas containing the history of revisions made over time. This allows the user to move backward in time to recreate any previous version. Polytron’s PVCS is an example of this latter type (Kinzer and Kinnaird 1987).

  Version control systems provide a mechanisms for both documenting sources and explicitly representing input/output relationships. Their major limitation results from their inability to create meaningful deltas from spatial data. In addition, they lack an automated capability for documenting the transformations applied to input layers used to derive a product. Users of version control systems tracking changes to aspatial data could cope with this limitation by manually keying in text to describe each transformation and its associated parameters.

**Lineage in GIS**

Users taking advantage of a growing range of source materials and automated map derivation tools feel the need for lineage information. GIS users participating in even small projects are quick to realize their ability to create new maps exceeds their ability to track the processes used to create them (Moore 1983). To address the problem of lineage in GIS, Krogulecki and Parks have proposed a system of standard forms to manually document digital map creation (Krogulecki and Parks 1988). Information concerning source digitized materials are entered by GIS users into a log book along with specific transformations applied in the course of file creation. This information can then be typed into a computer file and exchanged as a file header field of free text with an associated layer (Guptill 1987; Cooper 1989). One problem with the manual approach to lineage reporting is that it requires a conscientious effort on the part of all users to fill out forms at each step of a spatial application. A means for automatically creating lineage documentation and associating it with a derived GIS data product is clearly preferable.

A review of various commercial geographic information system manuals (from ESRI, ERDAS, Geobased, IBM, Intergraph, Tydac) and telephone interviews with technical staff of other GIS vendors and systems integrators (Anderson Consulting, Deltasystems, GeoVision, Intergraph, Kork, McDonald Douglas, Software-AG, Synercom, USA-CERL, and Wild Heerbrug) reveals many systems have capabilities for maintaining history, log, and audit files, but all lack an automated lineage tracing capability. Some vendors argue that their systems maintained critical data management information within "map librarians", or as "polygon overlay attributes".

- **Map Librarians.** In general, map librarians are a series of tools for partitioning and organizing a geographic database by location and theme (Aronson and Morehouse 1984). They provide a spatial and thematic framework for organizing a cartographic database. This framework is a basis for establishing checking-in and checking-out procedures. Such procedures are useful for controlling the process of data archival. This ability to control a data archive facilitates implementing standards of data content consistency (ESRI 1989).

  Map librarians, however, are not intended to record the lineage of derived maps. While data sources may be stored as layers within a map library, the sources are not documented with attributes concerning date (in world time), scale, projection, agency, and accuracy. In addition, librarians are not equipped to record the dynamic nature of database structure with temporal topology as new layers are created, nor the history of transformations applied when various layers are used to produce a product.

- **Polygon Overlay Attributes.** Of the various families of spatial analytic functions polygon overlay is one of the most powerful. Polygon overlay functions are set theoretic operations. Such operations involve the superimposition of spatially registered thematic map layers. The result is a composite representation exhaustively specifying the spatial interaction of thematically differentiated geographic features contained within input layers.

  Systems supporting spatially topological polygon overlay often handle spatial and aspatial data separately. In topological polygon overlay the newly derived geographic features inherit their aspatial attributes from input features. The aspatial attributes may
be organized as attribute tables transferred from sources and merged as annotation of the derived features. Often the new attributes maintain a symbolic connection to their parental’ input layers. In these cases database time topology is maintained as attributes of the layer created by the overlay.

Examination of the attributes inherited by a new feature reveals a lack of documentation concerning the type of overlay (e.g. union or intersection) and command modifiers (e.g. tolerances) used to derive the resulting map. Thus, critical information for determining the lineage of a product derived by topological overlay is missing. In addition to neglecting documentation of transformations, such systems often lack annotation concerning the nature of imported or digitized source layers. Such systems also often lack documenting imported or digitized source layers with lineage attributes. Another problem concerns the fact that many GIS applications are not based solely on map overlay. GIS applications often use other types of transformations (e.g. buffer generation, aspatial attribute selection, contiguity or connectivity determination). In contrast, to those derived by polygon overlay, features and layers resulting from these operations are not connected to their inputs. Therefore, systems supporting topological polygon overlay often lack documentation concerning data sources, transformations and associated parameters. While they do document inputs related to features derived by polygon overlay, they do not document input/output relationships between a number of other important GIS operators.

In summary, lineage information has been defined as consisting of three components. These components are: source data description, transformation documentation, and input/output specifications. The combination of the three is a complete description of the structure of a GIS application. As such, lineage information expresses the logic resulting in derived data. It is a model useful for communicating the theory embodied in a GIS application and the meaning of its derived data product.

A number of automated techniques were examined and found to be ill-equipped for creating or maintaining such critical lineage information. History lists, log and audit files document transformations and parameters lack source description and input/output specifications. Map librarians lack all three components of lineage information: source description, transformation documentation, and input/output specification. Polygon overlay attributes maintain input/output specifications but neglect source description and transformation documentation. While such systems do document the use of data sources used in deriving layers by topological polygon overlay, they do so for a limited number of transformations.

The only systems found to have the potential to include the three informational components of lineage are manual ones. Combining source descriptions with the application detailing of a cartographic model is a useful method of specifying a lineage report for a GIS application’s product. Automating this information will insure systematic documentation of the lineage qualities of derived GIS data products.

**Difficulty of Use**

The quality of the user interface has a great bearing on the utility of a geographic information system. The user interface, however, has not been a strong point of GIS (Cowen and Love 1988; Egenhofer and Frank 1988). To increase the efficiency of GIS the user interface must provide a conceptual model of what is happening to the database (Collins et al. 1983). It must be easy to learn, appear natural, and independent of implementation complexities such as data structures and algorithms (Egenhofer and Frank 1988). In order to do this, the user interface of the GIS should present itself to its user as a system, rather than as various collections of data (Driver and Liles 1983).

Deficits in traditional GIS user interface design results from a focus on how to best represent the software functionality rather than on how to meet the expectations of the user. This results from an orientation of GIS creators that treat user interface design as an engineering problem. Engineering typically attempts to eliminate the subjective factors, but it is exactly the subjective factors that are critical to the usability of any information system. To create a successful user interface the designer must realize that users do not necessarily use algorithms, data structures, networks, functions or subroutines, even though as technical professionals this is typically the domain in which they work. Instead, system users press buttons, choose options, type, make selections from menus, give commands and manipulate controls. In other words, interfaces used by users to control software functions are illusions that hide the underlying architecture of the technology prominent in the programmer’s view and repackage it as something understandable and usable by analysts and decision makers. Some of the most successful user interfaces are complete illusions which on the surface bear no resemblance to the data processing happening inside the machine. Of course, these illusions require their own code. Sometimes more than 60% of the code in a complex software system is dedicated to the user interface. This stands in sharp contrast to the 35% dedicated to the user interface in early GISs (Nicholson 1983).

Why do users need illusions? One of the major problems for end-users is that the things computers let them do are abstract. Even the terminology employed: booting up, paths, directory trees, relational keys, login on and off, function keys, scroll-bars, CONTROL-ALT-ESC key combinations, etc. is often unfamiliar and grounded in abstraction. In everyday usage of controls, such as light switches, door handles and shower fittings, there is usually a fairly obvious correspondence between the appearance of a control
and its function. If one is not sure which light switch on a panel controls which light will be illuminated, one can generally try them until the right one is found. John Carroll (1984) at IIBM calls this learning through exploration, a functionality that significantly increases the learnability of a system. In computer software, this element of physicality is often lost. Almost everything relating to the internal workings of a computer is hidden and largely divorced from anything the user understands.

To make systems truly usable software illusions are built on top of the underlying functionality. These illusions make abstract things appear concrete and give users the impression that they are controlling real objects. For example, the three dimensional buttons and animated pulldown menus of Graphical User Interface (GUIs) have utility which extends far beyond mere cosmetic drapery. They serve to restore an element of physicality and concreteness that promotes understanding and a feeling of being in control of computer software.

Users have mental models about the tasks they accomplish with a system, and the way the system lets them accomplish those tasks. These models are defined by the user's prior experience, existing knowledge, and preconceptions about tasks. To make sense a task and the way it is accomplished must correspond to the user's existing knowledge. In addition, for new material to be understood, interpreted and integrated with this existing knowledge it must be carefully introduced (Norman 1988). If new material is poorly introduced, users attempt to integrate it with existing knowledge and invent irrational explanations for the behavior of the system. For example, my students often blame themselves when the GIS packages they are working with crash as the result of software bugs. On other occasions, they can be found blaming the GIS for problems they inadvertently created. This behavior is typical of many GIS users. They tend to makeup their minds about the function of their systems based on concocted explanations for problems they do not understand. One of the jobs of the user interface designer is to make sure the conclusions they reach are the correct ones.

Each GIS user possesses a conceptual model of the software system he or she interacts with. This model is shaped and influenced by internal and external factors. The internal factors relate to the user's goals, expectations, intentions, experiences, preconceptions, past knowledge and explanations. The external factors relate to the system's interface as it initially presents itself to the user and in revealing additional dimensions as it is used. System interfaces designed to match the user's existing conceptual models may not require much documentation. GISs are typically designed without attention to the user. They require extensive documentation and training to change the user's conceptual models to fit the system. GIS support is a euphemism for helping users adapt to difficult aspects of the system. All of these: thickness of user manuals (in inches), training time (in days or weeks), and support (in numbers of telephone calls, e-mail, and fax messages) are indications of a failure of the GIS user interface to map onto the user's conceptual model.

**Update Handling**

Users may spend weeks editing and updating a particular source map that is widely used in GIS applications resulting in map products used by others. However, users updating the sources may not be aware of their other uses. As a result decisions often are based on out-of-date products. A key question is: how can those editing or updating a source map make sure that the "new source" data is used by all affected decision makers. Another key question is: how can decision makers make sure the maps they use are up-to-date.

GIS specialists must continue to toil without conceptual aids to create derived spatial analytic products for decision makers. As a result, GIS users spend too much time struggling to understand: (1) exactly what data they have at hand, and (2) what state of progress in the application does a particular GIS data layer represent? There is no way to know which source materials were used to create a derived data product nor which operations or parameters were chosen in the derivation of the maps that are being used. Often the only way to record information about what is in the database is through abbreviated file names and file creation dates within the computer operating system. But there is no way to be certain of which source materials were used to derive a particular layer from file creation dates nor layer file names. As a result it is impossible to track the data dependencies between data layers stored in the GIS database.

The result can be confusion. GIS database users trip over each other's data as they search through system directories. They do not know which data layers need to be regenerated due to a source update. When users are able to determine which layers are affected, they have trouble knowing the precise method required to update them. The resulting confusion diverts attention from design, implementation, and evaluation of GIS applications used to meet critical decision making requirements. One symptom of this confusion is the repeated use and publication of out-of-date maps, resulting in poor decision making and end-users losing faith in GIS derived data quality and eventual GIS disuse.

**Accuracy and Error Propagation**

Geographic information systems provide users with convenient and consistent mechanisms for applying automated transformation functions to manipulate and analyze spatial data. These capabilities expand the role and increase the value of spatial
databases used in a variety of decision-making contexts. Such systems, however, often lack capabilities for establishing the accuracy and validity of products derived to support decisions. That is, GIS provides a means of deriving new information without simultaneously providing a mechanism for establishing its reliability. The literature detailing GIS applications shows there is a lack of concern for error in spatial databases and its propagation through sequences of data transformation functions. In such applications, input data quality is often not ascertained, functions are applied to these data without regard for the accuracy of derived products, and these products are presented without an associated estimate of their reliability or an indication of the types of error they may contain.

Such omissions do not imply that errors are of such low magnitude that they can simply be ignored. Rather, they reflect the lack of a standard framework for modeling how error is propagated through sequences of data transformation functions. Paradoxically, an enormous volume of research has been carried out on the question of spatial database accuracy and the errors introduced by various types of data transformation (Goodchild and Gopal 1989; Veregin 1989a). Numerous indices have been developed to measure spatial and aspatial dimensions of error in databases, and methods have been proposed for modeling the ways in which data transformation functions modify and introduce error. Much of this research, however, has been carried out in isolation from the broader context of error propagation modeling in real-time throughout the course of spatial analytic applications processing. There is a lack of a methodology for specifying the interactions among these various error indices and models of error propagation. That is, there is no accepted paradigm for modeling error propagation that explicitly recognizes the interdependence between basic concepts of spatial database accuracy and formal methods of error propagation in an actual system.

Quality Control

A GIS database can be thought of as an abstract representation of a portion of the 'real world'. The database's 'integrity' is a property reflecting the extent to which this representation is an accurate model of that portion it represents. Database integrity can be negatively affected by the entry of errors in data. These errors can occur in a number of ways, each of which must be identified and prevented in order to successfully protect the integrity of the database. The many possible sources of database corruption include hardware, software, and operator causes. After decades of refinement and tuning, the role of hardware and software in database corruption has been dwarfed by that played by the GIS operator. This evidenced by the call by GIS protagonists for:

- self-checking mechanism to detect and potentially correct data errors (Peuquet 1983),
- the use of integrity constraints which help to guarantee long term usability of data by detecting clearly erroneous data before they are stored (Frank 1984),
- a system to assist a user in overlaying and combining data sets by indicating likely sources of error (Robinson and Jackson 1985),
- an ‘idiot-proof’ system that discourages dangerous practices (Rhind 1988), a negative system to discourage dangerous practices (Muller, 1989),
- a system consciously designed to avoid or minimize misuse (Beard 1989).

Quality control procedures must be in place as users work with their data to insure the integrity of derived data. 'Semantic integrity' results when the data in the database complies with constraints derived from knowledge about what is not permitted in that portion of the real world which is modeled by the data in the database. The maintenance of semantic integrity in GIS involves barring the insertion into the database of derived features made explicit by prohibited applications of particular GIS functions to specific source materials. Thus, semantic integrity is maintained by procedures that 1) detect the improper use of GIS functions and data, 2) inform the user about the error, and 3) bar the entry of the resulting derived data into the GIS database.

A basic requirement of a system for preserving the semantic integrity of the GIS's database is the screening of user entered commands to insure that they do not violate any semantic integrity constraints. For example, automatic semantic integrity constraint based screening would detect a user attempting to overlay two data layers of incompatible map projections. The attempted operation would be barred from execution, the database would be protected, and the user would be informed of their error.

Data Format Translation

Data format translation often takes place as data is moved from one system to another. It is necessary to track the change in data structure as the data is manipulated to prepare it for analysis within another GIS, another subsystem of the same GIS, or an external data processing system. Such changes to the data structure include the mapping between raster and vector, image and raster, spaghetti and topological, gridded elevation and triangular irregular polygon (TIN) elevation, and changes in format of associated aspatial attribute tables.
Nyerges (1991) suggests that this type of research will serve as the basis for coupling external systems. This research can focus on how to manage external mathematical and statistical models within so called model management systems (Geoffrion 1987). Jankowski (1987) applied the idea of model management to water quality modeling in a stream. Solution of this problem will result in more statistical and mathematical models. Full integration with the GIS will result in more model-based GIS than data-based GIS (Birkin et al. 1990).

**Database Optimization**

Standard GIS macro applications provide GIS users with a mechanism for automatically manipulating the spatial database to derive map products. After being used as intermediate steps toward a final product, however, most data derived during spatial analyses become obsolete. They are often left in the database causing confusion as to what they are, while pushing storage devices to their limits. One way to deal with these problems is to keep spatial database contents to a minimum. Since cartographic products can be created by GIS application programs when needed, it makes sense to minimize database size by removing a subset of derived data. In order to do this in an optimum fashion it is necessary to explore techniques for trading spatial database space for savings in processing time and to apply them in a way that more efficiently utilizes available GIS resources.

Techniques for trading processing time for disk space are common in the computer science literature. Dynamic programming is an approach to algorithm design that breaks a problem into a set of smaller and simpler subproblems. Combining the solutions to such subproblems forms solutions to larger and larger parts of the problem until finally the original problem is answered. When subproblems overlap, this 'bottomup' technique saves time wasted recomputing duplicate solutions by storing and reusing subproblem results (Brassard and Bratley 1988). Similar techniques for trading data space for computation time are widely used in the implementation of recursive functional programming languages (Bird 1980). These techniques are useful since recursive functions often are repeatedly called with the same parameter values resulting in the calculation of redundant results. Therefore, it is more efficient to store and reuse these values rather than recomputing them each time. Such is the case in recursive algorithms that calculate Fibonacci numbers, solutions to the Towers of Hanoi puzzle, and binomial coefficients (Cohen 1983).

Michie (1968) proposed that efficiency could be gained in recursive algorithms by the use of 'rote' and 'rule' methods. Rote methods use tables to store the value of computed parameters (Frieman, Wise and Wand 1976). As each parameter is used, the recursive function first references the table for its value. If the table lacks the value a rule method is invoked. Rule methods are procedures for calculating missing values. Once calculated, the missing information is placed in the table for subsequent use by rote methods. As the recursive function terminates a large table has been built containing every value computed during execution of its algorithm. "Small table" methods offer a more efficient alternative to storing every computed value (Hilden 1976, Bird 1977, Cohen 1983). These approaches gain efficiency by preprocessing the recursive function to determine which set of values are used repeatedly and which are used only once. Reused values are computed and stored in a small table from which they are retrieved when needed again. Those values used once are not stored, but computed as necessary.

Storing reused values efficiently trades costs of storage space for savings in computation time. Beyond dynamic programming and recursive languages, this technique has been adapted for use in deductive database programs (Naughton and Ramakrishnan 1990). KBGIS-II is an experimental deductive geographic program, that trades data space for savings in processing time. KBGIS-II’s ‘learning’ capability is similar to table-based rote methods described above (Smith et al. 1987). Including rote learning capability allows it to save processing time by storing spatial coordinates of thematic objects found in response to database queries. Once stored, these objects are available for reuse in response to subsequent queries. Like small table methods, efficiency is gained by automating a choice between the use of disk space to store retrieved geographic features or the use of Central Processor Unit time to recompute their location at the time of the user's request. KBGIS-II’s ability to decide whether to store or recompute an object's location is based on an automated decision that determines when a feature is too costly to recompute and should be saved in the spatial database. This decision is made possible by an internal model KBGIS-II keeps of its own data processing. This model encodes information concerning the storage and derivation costs of geographic objects. Unlike KBGIS-II, most GISs do not create and use models of their own data processing. As a result, they cannot directly support functions that trade database space for processing time.

**3. RESEARCH AGENDA FOR FILLING THE GAPS**

Intelligent computer systems can play various roles required in filling the gaps described in the previous chapter. In designing human-GIS interaction support tools, it is important to abstract the cognitive and operational processes of the human operator. This is necessary to the goal of building support tools which can support the GIS user.
The increase in levels of automation of GIS operation will continue to lead to higher productivity characterized by an increase in sharing of responsibility between human operators and computer systems providing intelligent assistance. This assistance will not result in total automation right away. This would require an understanding of everything there is to know about spatial analysis and the application of GIS to support decision making, an understanding that is not presently available. As a result, decision support tasks will incrementally migrate from humans to machines. Task migration will happen as knowledge acquired by humans is transferred to machines. This sharing of tasks between humans operators and intelligent assistants can be viewed as a continuum. At one end, the human operator is in complete control of the GIS without an intelligent assistant system. At the other end, the intelligent assistant system is in complete control without human operator assistance. The gradations between the two extremes are characterized by tradeoffs in shared levels of responsibility. Boy (1991) introduces a theoretical model, based on the work of Sheridan (1984) that plots changes in performance of a human-machine system as levels of autonomy and knowledge of the system change:
Effect of levels of autonomy on performance

Six principal types of support roles can be considered when considering the productivity implications of intelligent assistants for aiding GIS users. These roles can provide a range of assistance viewed along a continuum from "doing nothing" to "doing everything". When the intelligent assistant is "doing nothing" all responsibility for attaining the goal of GIS application rests with the GIS user. When it is "doing everything" the intelligent assistant assumes "complete responsibility" for application goal attainment leaving the GIS operator completely out of the loop.

The roles are as follows:

1. User in command - Intelligent Assistant not used.
2. User in command - Intelligent Assistant as monitor.
3. User in command - Intelligent Assistant as supervisor.
4. User and Intelligent Assistant share command.
5. Intelligent Assistant in command - user as monitor.
6. Intelligent Assistant in command - user not consulted.

User is in command - Intelligent Assistant not used

In this case, the Intelligent assistant does not play a part. The three entities that do are: the decision maker, the GIS user, and the GIS:

In this situation the GIS user is given a goal in the form of an information requirements statement from a decision maker. The user first commands the GIS to give a listing of the contents of its database. If any hardcopy or electronic documentation exists concerning the contents of the spatial database it is consulted. This information is evaluated with respect to the informational requirement received from the decision maker. If the information exists in the database, the user extracts it and delivers it to the decision maker.

If the information does not exist, the user determines if the goal can be attained by newly deriving the required information from the contents of the existing database. He/she does this by combining what is known about the information in the GIS database with knowledge about the functional capabilities of the GIS to determine if new spatial information can be derived to meet the goal of the decision maker. If the user decides that it is possible to derive the required information then he/she begins the process of submitting a sequence of commands to the GIS. This process is a "manual" one in which each command is sequentially typed in to the GIS. As each command is processed the user typically evaluates the outcome, and continues until the goal map is achieved.
The GIS operates under complete control of the user, responding to the sequence of commands issued by the user. The user in this case is unaided by automated support tools. This situation characterizes the contemporary GIS operating environment. That is, the user relies on documentation of the GIS database, formal training in the functionality of GIS, experience with the specific GIS, user and technical manuals, telephone support services provided by the GIS vendor, and knowledgeable colleagues for help in applying GIS functions to derive new spatial information for use in decision making.

After two decades of GIS use, not much has changed with respect to the support of the GIS user. Therefore, the research agenda proposed here is to apply artificial intelligence techniques of expert and related systems to create intelligent assistant systems to support the GIS user with capabilities for coping with the critical gaps:

- **Spatial Data Lineage Tracking,**
- **Difficulty of Use,**
- **Accuracy and Error Propagation,**
- **Quality Control,**
- **Data Format Conversion,** and
- **Database Optimization.**

**Intelligent Assistant as monitor**

As a monitor, the Intelligent Assistant’s role is to examine the User-GIS interaction and extract lineage information, making it available to the user as it is needed:

As in the case where Intelligent Assistant is not used, the GIS user is given a goal from a decision maker. This goal is delivered in the form of an information requirement specifying a request for the derivation of spatial information. The user evaluates the informational requirement received from the decision maker to determine if the needed information already exists in the spatial database. The user does this by consulting the Intelligent Assistant for lineage information necessary for cartographic source assessment. In addition to lineage information the Intelligent Assistant provides the user with additional information concerning cartographic type and accuracy measures reflecting the contents and quality of the contents of layers stored within the spatial database. This information (lineage, cartographic type, and accuracy indices) serves the user as he/she determines the meaning and utility of the database’s contents.

**Spatial Data Lineage Tracking,**

1) A conceptual design for a spatial data lineage tracking system has been presented in Lanter (1989, 1990, and 1991). The implementation of this lineage tracking system, however, must be tested in a production environment. Therefore applied research is
required to demonstrate that Intelligent Assistant is capable of acquiring, storing and providing the user with lineage information concerning the data in the GIS database in real time under production workloads.

2) Research effort is needed to adapt Lanter’s system to include the maintenance of lineage information documenting the derivation of thematic source materials from image processing systems used in the processing of remotely sensed data.

**Difficulty of Use**

3) As it is acquired, lineage information must be presented to the user in a way that facilitates the use of the GIS and increases productivity. A lineage-based graphical user interface design has potential for filling this gap (Lanter 1989). This interface has the potential of increasing productivity by easing the cognitive stress and effort associated with tracking spatial data lineage in production GIS settings. Therefore, empirical research should focus on the degree to which this lineage-based graphical user interface design matches the GIS user’s mental model and eases the current difficulty of GIS use as suggested by Lanter and Essinger (1991).

**Accuracy and Error Propagation,**

4) As the GIS is used by the user to derive information it must also simultaneously establish the accuracy of the new information. Providing this information will more fully support the user in determining the nature and quality of the data contained within the GIS’s spatial databases. Therefore, research is needed to provide the intelligent assistant with a means of propagating error measures to describe the accuracy of the derived cartographic data. Research should focus on creating [Lanter, Veregin 91a, 91b: Veregin, Lanter 91]:

i error indices to capture components of spatial data accuracy, and

ii. creation and testing of error propagation functions.

For a given GIS data transformation function, there will be a vector of error propagation functions (corresponding to a set of error indices) that depend on the error measurement indices to be propagated (corresponding to a row in the figure below). For a given error measurement index there will also be a vector of error propagation functions (corresponding to a column in the Figure below) depending on the GIS data transformations through which the error index is to be propagated. Automated error propagation research involves filling in the cells of the matrix with error propagation functions that matched the pair: GIS data transformation function, and error measurement index. The Intelligent Assistant must be designed in a way that uses GIS functions and error indices as keys for selecting the appropriate error propagation function to apply in automatic error propagation:
Research should also address whether or not an error propagation function for a specific index can be targeted toward an entire taxonomical group of GIS function (e.g. spatial filtering). In addition research should examine the translation between indices in various mappings such as 1:1, 1:many, and many:1.

**Intelligent Assistant as Supervisor**

In this role, Intelligent Assistant acts as a supervisor addressing the issue of automating semantic integrity constraints to protect the contents of the GIS's spatial database. It does this by observing the user's interaction with the GIS and applying rulebase semantic integrity constraints to determine whether or not to constrain the user's commands from reaching the GIS. To do this, the Intelligent Assistant requires an inference engine (see Appendix A) capable of accessing lineage information. One hypothesis of this research is that source attribute testing can serve as a basis for semantic integrity constraint checking. Lineage information processing can provide such source attributes for constraint checking. This will provide attributes of source materials tested by semantic integrity constraints obtained from applications experts. For example, the intelligent assistant:

1. detects that there is a problem
2. informs the user of the problem
3. suggests how to fix the problem

**Quality Control**

5) Research is needed to further extend the semantic integrity constraint rule processor demonstrated by Lanter (1989) within the Intelligent Assistant. This will add quality control by insuring that the user does not threaten the quality of the database by applying invalid GIS commands. For example, if the semantic integrity constraint processor detects an attempt to overlay two layers with incompatible map projections, the processor will inform the user of the problem and suggest a solution: such as reprojecting one of the layers before overlaying.

6) Semantic integrity constraints are formalized during the process of knowledge acquisition. During knowledge acquisition, semantic integrity constraints are extracted from experts and expressed in the rule form:

\[
\text{if } <\text{situations}> \text{ then } <\text{prohibit action}> <\text{make suggestion to fix situation}>. 
\]

Research should focus on the how source description attributes relate to the validity of derived data created as the result of applied GIS transformation functions. Once an improper transformation of source data is understood (e.g. arithmetic overlay of
nominal feature codes) it can be represented as a semantic integrity constraint. Research resulting in such integrity constraints will contribute to filling the matrix illustrated below:

These constraints are oriented toward detection of 'incorrect' uses, transformations and combinations of source materials. Acquisition of constraints to fill the matrix will be based on discussions with applications experts possessing an understanding of faulty uses of data and potential pitfalls resulting from data processing in the context of their domain of expertise. The result of the process of knowledge acquisition will be the specification of semantic integrity constraints to protect the integrity and quality of the GIS database. This effort should focus on associating each source with a set of attributes used by experts to insure the validity of the GIS transformations they apply. Each GIS command will be associated with a set of constraints which examine the characteristics of the source materials to determine if an illegal condition exists. In addition each constraint should be accompanied by suggestions to the user concerning alternative ways to derive the required information in a way that does not violate a constraint. Such suggestions should permit the user to both derive the required information and protect the integrity of the spatial database.

7) Research is also needed to focus on the creation of a software subsystem for knowledge acquisition that helps applications experts encode knowledge pertaining to database integrity and its maintenance in semantic integrity constraints. This sub-system should be based on an intuitive user interface that makes constraint building easy while supporting automatic detection of conflicting constraints. An example of a conflicting constraints is where one constraint is "do not use maps derived from sources with dates after 9/11/91", and another is "do not use maps derived from sources with dates on or before 9/11/91. The combination of the two rules results in a situation where no command is possible, since all source map dates are excluded from processing.

**Assistant in Command - User is Supervisor**

The support tools run the GIS in an automated mode, but the user can take control at any time.
8) After the intelligent assistant detects a violation of a semantic integrity constraint the intelligent assistant should be able to not only suggest a fix to a problem with the users command input, but also automatically apply this fix itself. Research is needed to determine the kinds of automatic fixes that the Intelligent Assistant can apply, as well as how to apply them to handle violations of semantic integrity constraints. The Intelligent Assistant will then be able to exhibit the following behavior:

1. detect that there is a violation of a semantic integrity constraint,
2. inform the user of the problem
3. suggest how to fix the problem
4. automatically fix the problem when it can

9) Research is needed to not only track lineage information but represent the temporal topology inherent in it to update the database. When users bring in updated source material it is important for them to know which products are affected by the sources. Source rippling would allow them to automatically ripple the new material via the temporal topology of lineage through intermediate derived layers all the way to the product. This would increase productivity by allowing for a central data source management authority that tracked all the uses of cartographic material.

10) Research at this level is also needed to focus on how to support the users as they move data between the GIS and external data analysis systems. This support should take the form of an intelligent subsystem for automatically changing the required data's content and format. This subsystem would consist of a collection of automatic data translators capable of recognizing the requirements of a particular destination system and altering the source GIS data content and format to meet those requirements. Translators resulting from this research will need to address both content and format conversions. Their format conversion capabilities can be organized in the following table:
Assistant in Command - User is monitor

In this role the Intelligent Assistant is in training to learn how to relate the decision maker’s goal to what it knows about the GIS and its spatial database. The Intelligent Assistant runs the GIS in an automatic mode, as long as this mode is not deactivated by the human operator.

11) The research agenda at this level of responsibility is to equip the Intelligent Assistant to become more autonomous and ease the burden of routine GIS database administration. Research should be aimed at enabling the Intelligent Assistant to maintain only needed information in a manner that optimizes the contents of the spatial database. Ibis research should focus on applying lineage
information modeling to determine the manner in which data is derived in the GIS, how it is used (i.e. source, derived, or product), and the associated costs associated with its derivation and storage, the result of which should be a sub-system capable of applying automated data reduction and data recreation routines to help maintain an optimal database configuration in ‘real-time’ as the user manipulates and updates the GIS database.

**Assistant in Command - User not consulted**

The GIS is built to operate only in response to the commands issued by the support tools which are always in control of it.

![Diagram]

It is not clear if this role is either attainable of desirable. However, in order to make this role possible it will be necessary to have success at all the preceding steps. For the Intelligent Assistant to be allowed to have complete rein over the GIS to meet decision makers goals it will first be necessary for trust to be earned. This may require the ability to encode not only intelligence, but intuition, loyalty, and a sensitivity to political climates and social values.

12) The research agenda for helping the Intelligent assistant to become completely autonomous includes:

- create a communication subsystem enabling it to directly acquire the goal from the decision maker.
- acquire the expertise necessary to successfully apply the GIS to meet a broad range of informational requirements.
- create the ability to select and apply available expertise to meet the goal of the decision maker.

### 4. EMERGING TECHNOLOGIES

Emerging technologies can be evaluated within the context of adding functionality associated with an increase in the level of autonomy and requisite intelligence to an intelligent assistant for GIS. Artificial intelligence (AI) research offers a source of relevant technologies for adding increasing amounts of intelligence to the intelligent assistant so that it can assume the roles of increasing responsibility discussed in the previous chapter. AI research often focuses on the study of intelligent behavior in humans as a basis for extending this behavior to computer systems. Intelligent behavior is abstracted to form simple conceptual models which are implemented in computer systems and tested. Testing examines the extent to which the implemented conceptual model reproduces the specific intelligent behavior under study. Successful conceptual models are iteratively extended to include more knowledge. This cycle of abstraction, modeling building and extension results in intelligent systems that employ large amounts of knowledge. The basic focus of building an intelligent assistant for the GIS operator is: 1) the knowledge required to apply GIS to support geographic decision making, 2) its representation, storage, access, and use in a system capable of intelligent behavior, and 3) testing and evaluation of its performance.
One result of AI research is a collection of techniques for building tools capable of autonomously perceiving and solving problems. Such tools have the potential of increasing human capabilities with respect to geographic information processing. Their utility is a direct result of the abilities they possess with respect to a particular activity. Such abilities can be viewed along a continuum from problem solving — requiring a high degree of attention and effort, to skilled behavior — which is automatic and relatively easy (Card, Moran and Newell 1983). This continuum can be viewed within the human performance model framework proposed by Rasmussen (1983) for analyzing human cognitive behavior. Human performance modeling breaks down task performance behavior into three levels: skills-level, rules-level, and knowledge level. These levels are ordered from:

- **Skills-based automatic behaviors** of situation-response at the skills level,
- **Rule-based systematic behaviors** applying domain specific knowledge at the rules level, to
- **Knowledge-based abstract behaviors** of plan formulation, conceptual modelling and simulation found at the knowledge level.

**Skills-based Behavior**

Skill-based behavior is acquired motor and/or cognitive behavior that results in rapid stimulus response type operations "without conscious control". In humans it is the result of long and intensive training that results in behaviors that only require stimulus detection for response generation. Skilled behavior in experts can be found as a repertoire of methods built up for dealing with stereotypical situations resulting from accumulated experience in performing tasks (Card, Moran, and Newell 1980). Each method is explicitly "written and debugged" as the expert gains cognitive skills by compiling and modularizing steps found at the rules-level into higher skills-level methods (Neves and Anderson 1981). The result is the cognitive equivalent of a compiled sub-routine. Each method is paired with a representation of the specific problem it solves. This insures that it can be readily selected and invoked as the problem is recognized. Thus, skills-based problem solving involves automatic problem detection, method selection, and method application. This ability in human experts results from familiarity with problem situations and practice with method selection and application (Rosenbloom and Newell 1982; Baecker and Buxton 1987).

Skills levels processing in computer systems is realized in the form of an analog to the stimulus-response pair. The stimulus part is typically put in the form of a conditional expression usually taking the form of a test of a value of a variable. Such tests can be found at various levels within computer systems. The most primitive form of conditional expression is the "conditional branch" found in assembly language programs. Conditional branching in assembly language programs tests the condition of the contents of a set of hardware registers. Based on the results of the test a decision is made as to whether or not the flow of control of an executing program, stepping through a sequence of memory addresses, should be passed ("made to jump") to another portion of memory. When such a conditional expression is tested and found to be true ("stimulus") control is passed elsewhere in the assembly language program ("response").

The analog of conditional branching in slightly higher level programs (e.g. early versions of BASIC and FORTRAN) is the expression "IF <condition> GOTO <label>". The IF statement is applied to test the <condition> which is typically an equality or inequality comparison of the value of a variable to a constant or that of another variable. When the condition is found to be true then the GOTO command transfers control to the part of the program identified with the <label>. Thus, when the "IF <condition>" evaluates to true stimulus is found. The "GOTO <label>" transfers control to the part of the code that is the response.

A higher form of stimulus-response is the "condition-action" capability of high level programming languages. In condition-action a stimulus is found with one of many forms of the "IF <condition> THEN <action>" statement. The condition part is implemented in the C programming language as a CASE statement, in LISP as a COND statement, in PROLOG as a matching of a predicate's Left Hand Side. The action side is a set of directives to be carried out by the machine. Although the compiler implements the direct action as a conditional branch to a section of code that executes the action, the programmer is insulated from that fact. Thus, the "IF <condition> THEN <action>" allows the programmer to encode stimulus-response activity in his/her programs.

Skills-based systems are typically targeted toward a special purpose that can be solved by means of simply detecting one of a set of conditions and applying an appropriate set of actions. Such a special purpose orientation is a reactive instinctual type of behavior not normally associated with transferability and reuse in other domains. Researchers interested in creating more intelligent levels of behavior in computer programs have looked from skills-based processing to more general forms of problem solving behavior.
Rule-based Behavior

One main goal of Artificial Intelligence (AI) research has been the application of general mechanisms of intelligence to solve specific new problems (Riesbeck and Schank 1989). The realization of the goal would be when a human operator could give the computer a problem which would respond with the answer. The expert system is often thought of as the state of the art in attaining this goal. Expert systems apply rule-based approaches in solving domain specific problems. The basic tenet in expert-systems is that expertise (i.e. knowledge of an expert) can be embodied in a set of rules. This tenet is based on the pioneering work of Newell and Simon (1972). Newell and Simon created one of the first AI programs called the General Problem Solver that resulted in one of the first rule-based expert systems called DENDRAL (Buchanan et al. 1969; Feigenbaum et al., 1971).

Rule-based behavior is based on the belief that the mind consists of a reasoning mechanism that can apply a collection of general rules to solve any problem. That is, general problem solving is made possible by a reasoning mechanism that is independent of domain specific knowledge. This reasoning mechanism is applied to the rules in the same way regardless of their content. Expert system constructions focuses on domain specific rules to use within the reasoning mechanism to solve specific problems.

Expert Systems and Geographic Related Information Processing

Expert system applications in areas related to geographic information processing can be classified into five generic problem solving roles: interpretation, diagnosis and prescription, design and planning, monitoring and control, and instruction (Waterman 1985; Ortolano and Perman 1990). The first three are discussed here. The problem of interpretation refers to the evaluation of the nature of a situation from data (Kim et al. 1990). Interpretation systems can combine expert systems with traditional database management systems for intelligent data retrieval for use in subsequent analysis (Kerschberg 1986). Examples include Tanic's URBYS intelligent urban information system (Tanic 1986) and "The Intelligent Information Center" developed by CarnegieMellon University for the Electric Power Research Institute in Palo Alto, CA (EPRI project manager is Shishir Mukherjee). Interpretation systems can also be applied to analyze legal issues pertaining to land use laws including the determination if a proposed use meets with all the relevant zoning and related regulations (Waterman 1985; Harmon and King 1985).

Expert systems have found many applications in reasoning about "malfunctions" and prescribing solutions. One example of such a general purpose inference engine is EMYCIN. EMYCIN was applied to various sets of rules to reason about both medical and nonmedical domains. EMYCIN was applied to the MYCIN rule-base to reason about therapies for bacterial infections (Buchanan and Shortliffe 1984). The PUFF rule base was used to reason about pulmonary-functions (Aikins et al. 1983). The CLOT rule base was added to reason about disorders of blood coagulation (Bennett and Engelmore 1984). In the non-medicine domain, the SACON rule-base was applied to reason about structural-engineering (Bennett and Engelmore 1984).

Expert systems have found application in the identification and generation of solutions within the contexts of urban planning and infra-structure projects (Waterman 1985; Levitt and Kunz, 1985; EPRI 1985a, EPRI 1985b; Ritchie and Mahoney 1986; Maser 1986; and Destrigeneville et al. 1986). Design and planning studies have applied expert systems to characterize inactive hazardous waste sites (Law et al. 1986) and find sites appropriate for a site selection (Findikaki 1986). Williams et al. (1986) discuss a number of systems for use in park management. Once such system makes use of spatial information establishes ecological zones for planning the use of controlled fires to manage ecological habitats. HYDRO assists in calibrating a watershed hydrology model (Gaschnig, Reboh, and Reiter 1981). MaPKBS uses a rule base for choosing appropriate map projections to apply in designing maps (Jankowski and Nyerges 1989; Nyerges and Jankowski 1989). ISIS integrates an expert system, with a GIS, and an optimization model for use in site selection (Wright 1990).

Knowledge Acquisition

Much effort in AI research focuses on solving the "knowledge-acquisition bottleneck" problem described by Feigenbaum (1980). The bottleneck results from the amount of time and labor it usually takes for computer scientists to interview experts, extract knowledge, and encode it as a set of domain specific rules. This process is often time consuming and expensive. For example, knowledge acquisition for building a rule base for the cancer treatment advisor ONCOCIN took over 2 years, 60 person-hours were spent building a very small rule base for CLOT, 300 person-hours were spent building a depression treatment advisor Blue-Box, 2 years of many peoples effort were applied to develop the rules included in a telephone switching system advisor COMPASS for GTE, and over $8 million was spent for a rule base for evaluating the risk of insurance underwriting (Musen 1989).

In order to overcome this bottleneck much research has focused on providing "knowledge engineers" with tools for extracting knowledge from human experts functioning at the knowledge-level and encoding it in rules usable within standard reasoning mechanisms existing in both "home grown" and commercial expert systems. A number of systems for aiding in knowledge acquisition are described in a collection of papers edited by Boose and Gaines (1990). Boy (1991, pp. 293-319) provides an excellent short and clear overview of many knowledge acquisition systems.
The modeling of spatial knowledge is one area in which geographers are weak (Robinson and Frank 19870). "Geographers themselves", Lundberg speculates, "do not have a clear sense of what constitutes geographical expertise (Lundberg 1989)." One exception is Smith, who has recognized that the user's interaction with the geographic information system is an ideal source of knowledge. Smith has focused on the usercomputer interaction as a source of knowledge. Ms knowledge-based geographical information systems (KBGIS and KBGIS-11) learn about the location of spatial and aspatial characteristics of objects that the user focuses on (Smith and Pazner 1984; Smith et al. 1987). Another exception is Kubo's Trinity GIS (Kubo 1986). Trinity distinguishes between 'personal' and 'common' knowledge when it learns about places different users are interested in and the labels they use for referring to such places. Trinity's inferencing mechanism involves searching to see if it already knows which place the user entered label refers to; if not it models the user's application of spatial query procedures for extracting the area of interest from a cartographic database. The resulting place name and the rules for its extraction are then stored for use in responding to subsequent queries.

Example of Geolineus's Rule-based Processing

Geolineus (Lanter 1989; 1991) provides intelligent assistance with its rule-based inference engine capable of using a lineage knowledge representation as working memory in applying sets of cartographic propagation rules and semantic integrity constraints:

There are many ways of categorizing data type: spatial or aspatial; nominal, ordinal, interval, or ratio; point, line or area. Some GISs have commands for querying a layer for type information. An example of this is the ARC/INFO DESCRIBE command. The DE' SCRIBE command returns, along with other information, feature type content of a data layer. Such a command may be useful to the user, however, within its output DESCRIBE does not provide a simple response such as "layer contains line type data". Rather DESCRIBE returns a cryptic report containing several tables of information that must be examined and interpreted by the user. Such examination is important as the GIS user often must know the data type of input layers to apply a GIS transformation. This is often crucial because if a type is left out of the user's command the system will assume a particular default type for the transformation (e.g. ARC/INFO defaults to polygon type).

Cartographic type propagation rules were acquired in a systematic interactive manner. The result is a set of rules applied to automatically infer the cartographic type of derived data. This information is stored along with the lineage meta-data and is consulted by the user during database query, map building, and selection of GIS functions. This information is stored in the form of a "truth
Along one axis of the table is a listing of GIS functions. Along the other axis are the cartographic types being modeled within the GIS. These are repeated twice along the top of the matrix illustrated below. The first set of point, line, polygon, and field (implemented discretely in a raster data structure) relate to input cartographic types, the second to output types:

The buffer function calculates a distance zone around a chosen feature. It is only applicable to point, line, and polygon features. Regardless of which of the three input types are passed to the buffer function, the output will be polygon:
Knowledge-based Behavior

Knowledge-based behavior is oriented toward moving from an initial problem state to a final goal state. Knowledge-based behavior is required in unfamiliar situations where systematic rule-based procedures and automatic skills do not apply. In this situation: 1) the goal is stated in terms of the state of the environment and a requirement to fulfill, and 2) a useful plan is developed and selected based on its effect with respect to the goal. Plans can be developed as the result of trial and error or from an understanding of the interactions between the functional properties of the environment and the use of a conceptual model to predict the effects of the plan considered (Rasmussen and Goodstein 1988). The conceptual model provides a basis for simulating the results of implementing a plan within the environment. Knowledge-based behavior results from the ability to generate a set of plans, model each plan's application, evaluate the outcome, and choose between alternative plans (Georgeff 1987; Hendler et al. 1990; Allen 1990).

Most systems capable of knowledge-level behavior generate a plan in response to goals expressed in terms of a problem to be solved. Wilensky (1983) suggests that a machine capable of autonomous knowledge-based planning requires an ability to infer which of many goals is most pertinent to the situation. In order to do this it must be able to identify its goals, determine which ones are relevant, and infer which one is paramount when goals compete. For example,

"...suppose a robot planner was given the job of maintaining a nuclear reactor. In addition to sustaining the generation of power, the robot may also be in charge of keeping the floors clean, preventing meltdowns, cleaning up dangerous spills, while maintaining itself at the same time. However, most of the time most of these goals are inoperative. For example, the robot need not be concerned with cleaning up a spill until one occurs. Thus it is desirable to build a planner that can recognize one of these situations when it occurs and infer the goal it should have at the time. The robot should infer it has the goal of cleaning up a spill when one occurs, of defending itself and the plant if they are attacked by terrorists, and of replenishing its resources if they are low." (Wilensky 1983, p. 13)

Wilensky proposes four components of a knowledge level planner:

1. The Goal Detector
2. The Plan Proposer
3. The Projector
4. The Executor

The Goal Detector determines whether or not the planner has a goal. It determines if a situation relevant to planning exists. Relevant situations include environmental changes, existence of complementary or competing goals, or problems in the structure of a plan under construction.

The Plan Proposer is responsible for finding stored plans that are relevant to the current goal(s). The stored plans may be directly used in stereotypical situations, edited to fit the current situation, or combined to handle complex situations.

The Projector tests the plans by applying them within the context of a conceptual model of the problem domain. Projecting the impact of the executed plan on the environment requires a simulator capable of taking as input a conceptual model and a plan. The result is identification of plan "bugs" that result in undesirable effects on the future. Future problems created as the result of a plan may be amenable to additional planning. These problems are therefore passed to the Goal Detector to formulate sub-goals for improving the situation. If the plan does not achieve the goal, the plan proposer is called to try again.
The Executor tries to implement the sequence of tasks within the plan. As it does so it keeps track of the interactions between the task execution and changes in the environment to monitor its progress. At any time in the execution of a planned sequence of tasks the Goal Detector may be invoked by a problem (sub-goal) needing attention.

Riesbeck and Schank (1989) suggest that planning may not be the result of as much thought as imagined in many scientific treatments. Rather than basing plan creation on scientific principles, most people search their memories for the best plan that fits the current situation. These memories may be based on personal experience or those learned from others. Much of human reasoning, Riesbeck and Schank argue, results from matching a problem to a similar situation and adapting and applying a solution that was successful in the past experience. This is called "Case-Based Reasoning". Case-based reasoning assumes that people reason from their own relevant experiences ("cases"), or from experiences learned from others. Knowledge is thus the collection of cases an individual has or has learned about.

Since the number of cases an individual has is vast, it is important to store indexes that label them individually for retrieval. General principles expressed as rules and patterns can be extracted from cases for used to label the cases so that in new situations relevant cases can be retrieved. These principles can also be used to determine which cases are relevant to the current situation and how to adapt parts of cases that do not apply. Riesbeck and Schank make the point that general principles alone are too abstract and as a result lose a large amount of very useful knowledge. Knowledge-based behavior may not involve a great deal of understanding, rather, it involves being able to retrieve the right information at the right time. They claim "the most difficult part of thinking is the creation of labels upon processing an input that will allow us to retrieve something we already knew that was labelled with those labels so that we can produce it as output (Riesbeck and Schank 1989, pg. 9)."

Case-based reasoning research is a potential source of techniques for creating high levels of autonomy in GIS Intelligent Assistants. Research based on the concepts discussed above has resulted in a number of automated knowledge-based level reasoning systems, (although at this time none have been reported to have any direct application to GIS). Much theoretical work is required to take advantage of and adapt the advances offered by the conceptual frameworks found in a number of interesting systems. For example, NOAH (Sacerdoti (1977) is capable of detecting a situation consisting of a configuration of blocks and formulating and reasoning about alternative plans for stacking them. PANDORA (Wilensky 1983) implements the four components of a knowledge-level planner discussed above to solve common-sense problems and understand natural language. Additional examples focusing on case-based reasoning can be found in Reisbeck and Schank (1989).

These include: IPP (Lebowitz 1980), a system that reads and understand stories about terrorist activities. CYRUS (Kolodner 1984) understands and reasons about the diplomatic travels of Cyrus Vance. JUDGE (Bain 1986) uses heuristics to determine sentences for legal cases. As new cases are read JUDGE updates its heuristics to insure an overall consistency in sentencing. CHEF (Hammon 1989) generates new Chinese stir fry and souffle recipes by adapting old ones. It does this in a way that satisfies several goals simultaneously. If CHEF makes a recipe that is considered unsatisfactory by a human user, CHEF learns from its mistake by updating its case library to avoid making the same mistake in similar situations. COACH (Collins 1987) creates new football plays by adapting and improving old ones. MEDIATOR (Simpson 1985) and PERSUADER (Sycara 1987) both propose compromises in the domain of dispute resolution. HYPO (Rissland and Ashely 1986) generates arguments for either the defense or prosecution in legal patent law claim violations. PLEXIS (Altermann 1986) adapts old plans to new situations and has been used to generate plans for navigating in various subway systems in the U.S., and CASEY (Koton 1988) takes a set of symptoms and diagnoses possible reasons for heart failure.

APPENDIX: RULE-BASED SYSTEMS

Rule-based systems apply general inferencing mechanisms to knowledge expressed as production rules. The goal is to apply knowledge in a way such that an expert level of performance can be reproduced. The advantages of production rule knowledge representation are:

- modularity
- uniformity
- naturalness

Modularity makes it possible to structure the knowledge of the domain being studied as a series of distinct production rules collected together for application within a particular domain under consideration. Knowledge expressed as production rules takes the form of a set of stimulus-response pairs. Each pair has a left hand side ("IF" side), and a right - hand side (THEN side). The declarations that human experts often explain themselves is consistent with the "IF <condition> THEN <action>" stimulus-response format of production rules. This form of representation is best suited to domains where expertise is (Boy 1991):

- diffuse,
Expert systems consist of three parts:

1) working memory consists of a dynamically updated set of facts,
2) a rule base consisting of rules gleaned from domain experts, and
3) a rule processor consisting of an inference engine.

The inference engine applies the rules to working memory in one of two ways. Forward chaining is a deductive mechanism where the inference engine steps through the rules base examining each rule in turn. The inference engine attempts to match the condition on the rule’s left hand side with a set of facts in the initial rule base. If the condition does not match the inference engine sets the rule aside and examines the next one. If the condition is matched by a set of facts in working memory then the right hand action side of the rule is ‘fired’. Firing of a rule typically results in new facts being added to working memory. Once the inference engine finishes trying to match the left hand side of all the rules in the rule base it begins a second cycle. As rules within the cycle are matched and fired, new facts are placed in working memory. After repeatedly cycling through the rules the inference engine finally "settles down" and ceases operation when a cycle completes without firing a rule.

Backward chaining uses a different inference engine that is goal driven. In backward chaining an initial goal is expressed. The inference engine examines a rule by looking at its right hand side attempting to match the conclusion of the rule with the expressed goal. Each rule is examined in turn. If a rule’s right hand action (or conclusion) side matches the goal, the inference engine then seeks to see if the rule’s left hand condition side matches the facts in working memory. If any of the rules considered by the inference engine have a conclusion side that is the goal and a condition side that is matched by facts in working memory then the goal is verified by the system.

If the condition on the rule’s left hand side does not match the facts then the facts needed to match the left hand side of the rule are made the current goal (referred to as "subgoals"). Each of the subgoals examined in turn. If rules are found that have the subgoals on their right hand sides, and their left hand sides match the contents of working memory then the subgoals are considered verified as is the original goal. If the left hand side of any of the subgoals do not match they are recursively considered as subgoals of the previous subgoals. If the inference engine cycles through the rules and finds none whose right hand side match the subgoals then inferencing "settles down" and the original-goal is not verified.

Both goal-driven backward chaining and data-driven forward chaining inferencing are the result of a controlled composition of a sequence of acquired knowledge. Expert systems are the most successful application of artificial intelligence techniques. Before
1980 they were found in only a few specialize fields including medicine and chemistry. Since that time, many experimental expert systems have been built (Hewett 1986).

**BIBLIOGRAPHY**


EPRI 1985a. "BWR Shutdown Analyser Using Artificial Intelligence techniques." Special Report by the Electric Power Research Institute, Palo Alto, Calif


Hilden J. 1976 "Elimination of Recursive Calls Using a Small Table of 'Randomly' Selected Function Values." Nordisk Tidskrift For Informationsbehandling (BIT) 16(l). pp. 60-73


Tomlin, C.D. 1980, "The Map Analysis Package (MAP)" , Papers in Spatial Information Systems, Yale University School of Forestry and Environmental Studies

Tomlin; C.D. 1983, Digital Cartographic Modeling Techniques in Environmental Planning, Dissertation, Yale University


