THE ARCHITECTURE OF ARC/INFO

Scott Morehouse
Environmental Systems Research Institute
380 New York Street
Redlands, California

ABSTRACT

Arc/Info is a generalized system for processing geographic information. It is based on a relatively simple model of geographic space - the coverage - and contains an extensive set of geoprocessing tools which operate on coverages. Arc/Info is being used in a wide variety of application areas, including natural resource inventory and planning, cadastral database development and mapping, urban planning, and cartography.

The design philosophy and architecture of Arc/Info is described. This includes the spatial data model, the spatial operators, the engineering of the system as a practical software product.

INTRODUCTION

This paper provides a general overview of the philosophy and architecture of the Arc/Info geographic information system. I begin with an overview of the basic approach to GIS system design used by Arc/Info, then review briefly the geographic data model implemented in the system. Any real system is more than just a data model, so the basic geoprocessing tools associated with Arc/Info are introduced. Finally, geographic information systems are complex software systems. I discuss some of the software engineering philosophy and methods that have proved successful in creating Arc/Info.

GENERAL ARCHITECTURE AND APPROACH

There are two basic approaches in GIS development today - the Spatial DBMS approach and the Spatial Tool Kit approach.

In the spatial DBMS approach, the GIS is considered to be a query processor operating on a spatial data base. Users and applications get information by passing a request to the query processor, which navigates the data base to find the answer, which is then returned to the application. In this way, details of the data base implementation are hidden from the application, and other useful functions like concurrency control and crash recovery can be managed. To be useful to the user, however, such a query processor must be very sophisticated - knowing polygon overlay, thematic mapping, attribute modelling,etc. In practice, query processors often simply retrieve geographic data using spatial and attribute keys, leaving more complex geographic modelling and cartographic tasks for the application programmer. Some data modelling problems can be solved within the data base. For example, the polygon overlay problem can be solved by overlaying all data as it is added to the data base. In this way, all queries involving multiple data layers can simply become attribute based queries.

The spatial DBMS approach typically involves an interactive database editor which is used to load and edit the spatial data base. It establishes the necessary topology, spatial indexes, and between layer links necessary for the query processor.

This is the classic Data Base Management System approach to the GIS problem: "How can we extend the (choose one: network, relational, object-oriented) data base approach to support geographic information?" This approach is popular in recent GIS designs (see, for example [Frank, 1982], [Herring, 1987], [Bundock, 1987], and [Carlwood et al, 1987]). It is also pursued by computer scientists seeking to extend relational and object-oriented data base management systems [Stonebraker, 1986].

The principal drawback of the spatial DBMS approach is the difficulty of application development. If the problem cannot be solved by the query processor, then an application program must be written that extracts the relevant information and does the geoprocessing problem itself.

The other basic approach is the application development or tool box approach. The central paradigm of this approach is "application oriented tools operating on objects". It is more closely related to work in document processing and software development environments (e.g. UNIX) and to fourth generation languages than to the DBMS approach. In the tool box approach, we define objects, which are pieces of geography, together with a set of geoprocessing operators for these objects (see figure 1).

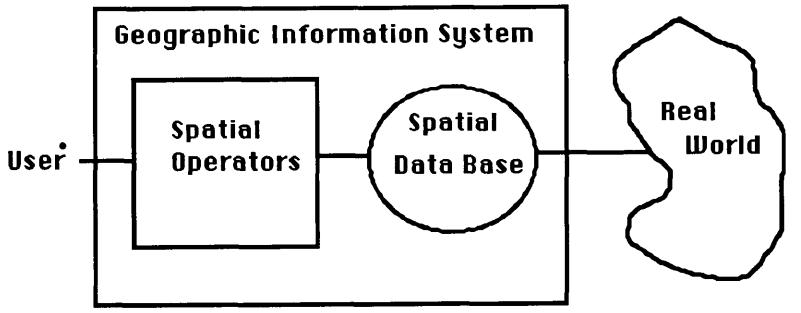


Figure 1: The GIS as a Geoprocessing Tool Box

Objects are stored in a data management system which provides for storage on disk, backup, concurrency control, etc. Operators (tools) are organized into a high level language system which provides a standardized user interface and a mechanism for combining tools into higher level tools.

Unix is one example of the tool box approach. In unix, the

objects are files. A file is simply a string of bytes with a name. Files are organized into directories and stored on disk. Files can contain any sort of data, although some tools may assume files contain text, object code, or executable code. The operators are unix commands. Most unix commands act as file processors, reading in files and writing transformed files. Commands are organized by the command shell. The shell provides the user interface along with a mechanism for writing command procedures (shell scripts). Unix is a powerful text processing and software development system because complex operations can be easily created by combining generic predefined operations.

There are other examples of the tool box approach in mapping applications. Most image processing systems follow this approach with the images as objects and image transformations as being the operators. The Map Analysis Package (MAP) is a geographic information system where the objects are grids and the operators are grid cell analysis commands [Berry, 1987].

These systems illustrate some features that are important attributes of the tool box approach.

Uniform Objects. If operators are to be combined flexibly, they must input and output data in the same format. It should be possible to take the output from any operator and use it as input to any other.

Object Management. There must be a data management system for objects which allows them to be organized and managed with security, backup capabilities, distributed data base functions, etc.

User Interface. There must be an environment which manages the user interface to operators and allows new operators to be easily created from existing ones.

General Operators. Operators should be designed as general purpose functions. This allows them to be combined flexibly for a variety of different applications.

The ARC/INFO geographic information system is based on the tool kit philosophy. It was inspired by earlier developments in unix, the Map Analysis Package [Tomlin, 1983], and the Odyssey geographic information system [Chrisman, 1979][White, 1979].

In ARC/INFO, the objects are vector locational data and the operators are geoprocessing commands for editing, analyzing, and displaying these objects.

THE ARC/INFO DATA MODEL

The ARC/INFO data model, together with its goals, is described in detail elsewhere [Morehouse, 1985]. It will be outlined briefly here. A fundamental goal in the development of the data model is that it perform well in the tool kit approach. This requires a simple yet general data model.

The basic unit of data management in ARC/INFO is the coverage. A coverage defines locational and thematic attributes for map features in a given area. The coverage concept is based on the topological model of geographic information and may contain several types of geographic features. Figure 2 shows the principal feature classes that may be present in a coverage. These feature classes form the basic vocabulary used to define geographic information in a coverage. By varying the types of features contained in a coverage and the thematic attributes associated with features, the coverage can be used to represent many types of map information.

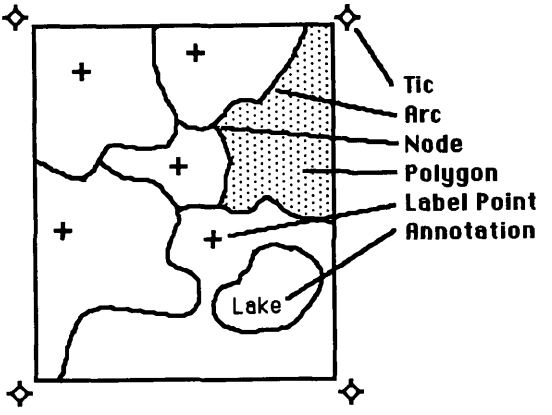


Figure 2: Feature Classes in an ARC/INFO Coverage

Each feature class may have an associated feature attribute table. Each table defines the attributes (called "items") for all features of that class in the coverage. There is a record for each individual feature. The feature attribute tables are an integral part of the coverage and are processed by ARC for all ARC/INFO commands which affect the coverage. For example, when two polygon coverages are overlaid to create a new composite coverage, the polygon attribute tables of the input coverages are joined and written as the polygon attribute table of the output coverage.

ARC/INFO provides a mechanism for the management of very large geographic data bases through the Map Library. The map library organizes data as a set of layers and tiles. Layers are, in most respects, like coverages except that they are partitioned spatially by tiles. The Arc/Info data base is implemented using relational data base modelling techniques. A coverage is defined by a set of relations. Some of the key relations in the coverage are:

```
ARC: (arc#,f-node#,t-node#,l-poly#,r-poly#,xy...xy)
AAT: (arc#,item-1...item-n)
LAB: (label#,poly#,xy)
PAL: (poly#, arc#...arc#)
PAT: (poly#,item-1...item-n)
```

These relations define the geometric, topological, and

attribute values of the various features in the coverage. We have found the relational approach to be very valuable for a number of reasons. First, each Arc/Info tool can choose to access and create coverage relations in the way most appropriate to that tool - there isn't a single method used by all tools to access and update the data base. One example is writing new arcs to a coverage from a bulk data process (e.g. polygon overlay). In the bulk process, some relations, such as the list of arcs around polygons, can be created via more efficient algorithms than would be possible in an interactive editor.

Second, and more importantly, the relational approach allows us to grow the data base schema by simply adding new relations to the coverage model. For example, spatial indexes for all feature classes were added to Arc/Info by simply creating some new relations in the coverage and then teaching the spatial search module how to use them.

THE GEOPROCESSING TOOLS

Given the definition of objects in the Arc/Info data model - coverages, map libraries, tics, arcs, etc. - Arc/Info can be defined as the set of appropriate and useful tools which operate on these objects. This is an open-ended definition. The Arc/Info tool box is intended to grow and develop with GIS technology and with our users needs.

The Arc/Info tools operate at a variety of levels. There are tools which operate on entire map libraries, others which operate on coverages, and finally tools for manipulating individual features.

Map library tools. These tools all operate on map libraries (see figure 3) and are collected as the librarian function.

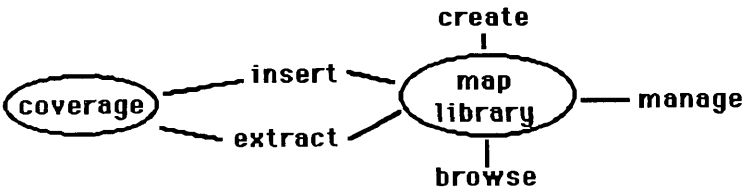


Figure 3: Map Library Tools

The librarian defines and manages map libraries. The librarian has a number of tools which operate on libraries. Geographic data in the map library is managed like software in a source code management system. To perform an update, the relevant layers and area of modification are extracted to an operators workspace where the geographic data is edited and the edits verified. The verified data is then reinserted into the map library.

The librarian manages this entire process as a transaction on the map library and prevents simultaneous extraction of the same layer and affected area for modification by other users of the library.

This extract/insert approach to transactions on geographic data bases is necessary because revisions to geographic data generally involve long highly interactive processing and verification of the data. In most ways, update transactions on a geographic data base are more like transactions on a source code library than transactions on a tabular data base [Aronson, 1989].

The librarian also provides browse functionality. In this situation, most query and display tools which operate on coverages in a read-only fashion can also operate on entire map library layers as if they were a single seamless coverage.

Coverage Tools. These tools all operate on entire coverages, managing all feature classes and associated attributes in the coverage as a single unit (see figure 4).

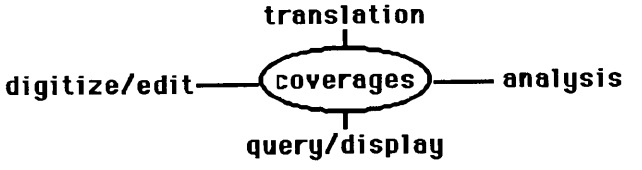


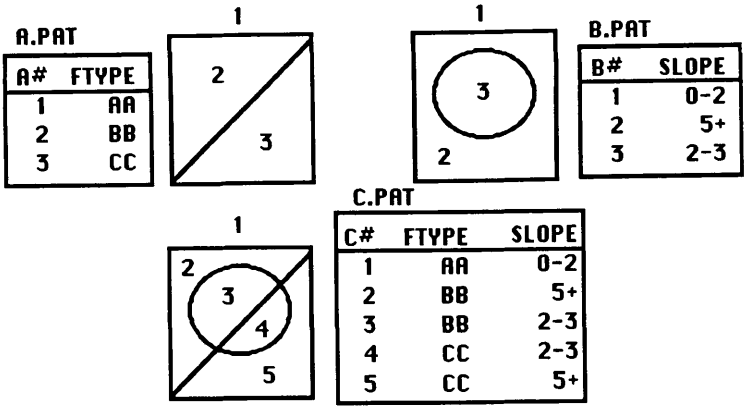
Figure 4: Coverage Tools

These tools can be loosely grouped into four categories: translation, digitize/edit, analysis, and query/display.

The translation tools perform the conversion of data between a variety of spatial data formats and Arc/Info coverages. Translators which are presently supported include DLG-3, DXF, IGES, Moss, SIF, ascii, and various raster formats.

The digitize/edit tools support creating and editing coverages. The primary tool (or collection of tools) here is Arcedit. Arcedit is an interactive graphics editor for coverages. Other tools exist to support bulk generation of topology, data verification, form driven attribute data entry, and a number of other tasks necessary in creating a geographic data base.

The analysis tools perform spatial analysis functions involving one or more coverages. Generally the results of the analysis are written as a new coverage or as additional attributes on an existing coverage. The classic example of this type of tool is the polygon overlay or "spatial join" tool. This class of tool takes two coverages, finds all intersections between features and writes the resulting integrated coverage as a new coverage (see figure 5).



UNION A B C

Figure 5: Coverage Overlay

Arc/Info has an extensive set of spatial operators at the coverage level. These include:

coverage overlay:
- polygon on polygon
- point on polygon
- line on polygon

thiessen polygon generation
contour interpolation
buffer zone generation
network allocation
map projection and coordinate transformation
rubber sheeting
generalization
feature selection and aggregation
arithemetric and logical attribute combination

The Arc/Info data model has been specifically designed to support these coverage level spatial analysis tools as well as query and edit tools which operate at the individual feature level.

The query/display tools are used to view the geographic data base and to perform ad hoc queries on the data base. Tools are provided to define and edit catrographic symbols, to scale and position map graphics, to associate catrographic symbols with geographic feature attributes, and so on. As a brief example of these tools, imagine the problem of selecting and drawing all roads in a given area which pass through hardwood forests. The Arc/Info tools which could be applied to this query are:

```
reselect forest polygons type = 'hardwood'
reselect road lines overlap forest polygons
arclines roads type type.symbol
```

The first operator performs an attribute selection on polygon features in the forest coverage. The second operator is a

overlap query which finds all line features in the road coverage which overlap the previously selected forest polygon features. Finally, the third operator displays the selected roads using cartographic symbols derived from the road type attribute.

Nearly all query/display tools operate on map library layers as well as individual coverages.

Feature level tools. These tools operate on individual features within a coverage (see figure 6).

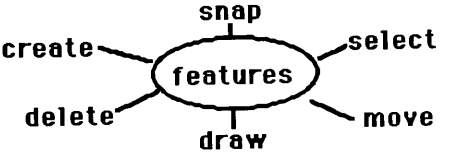


Figure 6: Feature Tools

The primary collection of these tools in Arcedit, the Arc/Info coverage editor. Arcedit provides tools for selecting features, then modifying them in various ways.

THE USER INTERFACE

The Arc/Info user interface can be defined at two levels - the base user interface and the application-oriented user interface. The application-oriented user interface is built on top of the base interface using the Arc Macro Language (AML). Figure 7 illustrates this concept.

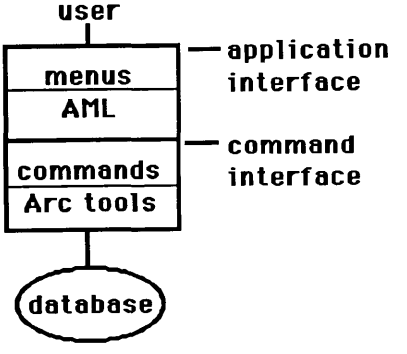


Figure 7: The Arc/Info user interface

The base user interface is a simple command language similar in purpose to operating system command languages found in unix, MS-DOS, VMS, and other operating systems. It is based on the paradigm of tools and objects. Each tool in Arc/Info has an associated command in the Arc command language. This language is very simple, consisting of a verb followed by one or more objects or command qualifiers. For example, the command to invoke the polygon overlay tool is:

```
intersect <in_cover> <intersect_cover> <out_cover>

(where <in_cover>, <intersect_cover> are names of two
       existing coverages and <out_cover> is the name
       of a new coverage to create with the results.)
```

This level of the interface is designed for generality, extensibility, and flexibility. Commands can be either entered interactively from a keyboard by the user or can be supplied from AML procedure or menu. The command language can also easily grow through the addition of new commands and command qualifiers.

The application-oriented user interface is built on top of the base command interface using the AML language. AML is a procedural language interpreter designed specifically for Arc/Info. It has all of the features typically associated with operating system command languages, such as named variables, flow of control directives, numeric and string operators, and so on. AML also defines a set of user interface objects. These include pull down menus, pop up menus, and forms. These objects can be used to develop a user interface which is designed for a specific application or to provide a non command driven user interface.

THE SOFTWARE ENGINEERING APPROACH

Any GIS is a significant software engineering problem. To be useful, any GIS has to be well engineered. A clever data model or powerful user interface is useless unless the software performs correctly and will work effectively with large collections of geographic information. Software engineering issues are central to the viability of any GIS; they are also very interesting problems in their own right. Geographic Information Systems are ideal software engineering test cases - they involve database, graphics, computational geometry, user interface, and operating system technology. All of these technologies as well as geographic data modelling and analysis and user requirements are experiencing rapid and continual change.

To thrive in this environment, Arc/Info has been designed as an system which can grow and change. It is not a static system which meets a fixed set of preordained specifications. Arc/Info 5.0 is very different from Arc/Info 4.0 of just two years ago, Arc/Info 6.0 will be different again. The central goal in engineering Arc/Info has been to develop an architecture and programming methodology which supports this process. Once you realize that a system must evolve over time a number of other principles follow [Meyer, 1988].

The system must be maintainable. It will be continally modified, extended, and optimized.

The system must be portable. who knows what the computing/operating environment of the future will be?

The system must be as simple as possible. Simple systems can evolve much faster than complex systems.

The system must be reusable. Code and algorithms must be designed in a way which supports reuse in future, unforseen applications.

Clearly, the system must also be expandable and correct. Designing for correctness in an evolving system is different than ensuring that a system functions correctly for a single fixed design.

To accomplish these goals, we have adapted the techniques of modular software design and development [Parnas, 1972] in Arc/Info.

We organize all software development around the concept of the module. A module is simply a collections of routines which work together to define a data structure or to perform some function. Modules are entirely self-contained - the code within one module only interacts with code outside the module by well defined function calls. We have identified a number of module types in Arc/Info:

data structure module - defines and implements the behavior of a data structure (e.g. BITSYS - a bitmap manager).

device interface module - define and implement the behavior of a virtual hardware device (e.g. DIGSYS - the digitizer interface).

processing module - define and implement a functional process. This can either be a generic process (e.g. SRTFIL - a disk based sorter) or a specialized process (e.g. OVRSEG - find all intersections between two, potentially huge, sets of line segments).

program module - define and implement an executable program. Typically defines the user interface and functionality of a high level Arc/Info function (e.g. ARCPLOT - the cartographic display and query system).

Each module is typically the work of a single programmer and is designed as an independent unit. Modules only depend on the functional behavior of other modules that they may use. This means that the internal workings of a module can be freely replaced or extended. For example, we have replaced the internal workings of the segment intersection module a number of times without affecting the modules which use it (other than increased performance and reliability).

This modular software engineering approach, together with the simplicity and extendibility of the basic Arc/Info data model are what allows us to continually grow Arc/Info as a software product.

CONCLUSION

Two popular approaches to GIS design are the spatial database management system and the geographic tool box. Arc/Info is an example of the tool box approach. The data model of Arc/Info is based on a combination of the topological network approach for locational information with the relational approach for feature attributes. Arc/Info has an extensive

set of tools which can operate on this data model. Users can interface with the system either at the basic tool level or through applications and interfaces layered on top of these tools. The primary goals in the development of Arc/Info as a software system has been generality and extendibility. All aspects of the system from the data model to the user interface to the internal engineering of the system have had these goals in mind.

This approach and GIS architecture has been very successful. Arc/Info is presently in production use at over 1000 sites worldwide. It is being used for a wide variety of applications including natural resource planning, catrography, tax assessment, and facilities management. The system is a mature system which will continue to evolve and grow to support the changing needs of our users and the GIS community as a whole.

REFERENCES

Aronson,P. (1989), "The Geographic Database - Logically Continuous and Physically Discrete", Proceedings, Auto-Carto 9, Baltimore, Md.

Berry,J. (1987), "Fundamental Operations in Computer-Assisted Map Analysis", International Journal of Geographical Information Systems, v.1, n.2, p. 119-136.

Bundock,M. (1987), "An Integrated DBMS Approach to Geographic Information Systems", Proceedings, Auto-Carto 8, Washington, D.C., p. 292-301.

Carlwood,G., G. Moon, and J. Tulip (1987), "Developing a DBMS for Geographic Information: A Review", Proceedings, Auto-Carto 8, Washington, D.C., p. 302-315.

Chrisman,N. (1979), "A Many Dimensional Projection of Odyssey", Laboratory for Computer Graphics and Spatial Analysis, Graduate School of Design, Harvard University.

Frank,A. (1982), "MAPQUERY: Data Query Language for Retrieval of Geometric Data and their Graphical Representation", Computer Graphics, 16, p.199-207.

Herring,J. (1987), "TIGRIS: Topologically Integrated Geographic Information System", Proceedings, Auto-Carto 8, Washington, D.C., p. 282-291.

Meyer,B. (1988), Object-oriented Software Construction, Prentice-Hall, London.

Morehouse,S. (1985), "ARC/INFO: A Geo-Relational Model for Spatial Information", Proceedings, Auto-Carto 7, Washington, D.C., p. 388-397.

Parnas,D. (1972), "On the Criteria to Be Used in Decomposing Systems into Modules"", Communications of the ACM, vol. 5, no. 2, pp. 1053-1058.

Stonebraker,M. (1986), "The Design of POSTGRES", Proc. 1986

ACM-SIGMOD Conference on Management of Data, Washington, D.C.

Tomlin,C.D.(1983), "Digital Cartographic Modelling Techniques in Environmental Management", Doctoral Dissertation, Yale University, School of Forestry and Environmental Studies, New Haven, Connecticut.

White,D. (1979), "Odyssey Design Structure", Harvard Library of Computer Graphics, 1979 Mapping Collection, Vol. 2, pp. 207-215.