A Pilot Facilities and Management Information System (FMIS) for Texas A&M University-Corpus Christi

A Project Report
Presented to The Faculty of
Texas A&M University-Corpus Christi
In Partial Fulfillment of the
Requirements for the Degree
Master of Science

Michael Garrett B.Surv., ARICS, AISA
December, 1993
A Pilot Facilities and Management Information System (FMIS) for Texas A&M University-Corpus Christi

Michael Garrett

Approved:

Patrick R. Michaud, Chairman  Roy S. Ellzey
Assistant Professor of Computer Science  Professor of Computer Science

Herbert R. Haynes
Professor of Computer Science
Acknowledgements

I would like to thank Dr. Patrick Michaud, Dr. Roy Ellzey and Dr. Herbert Haynes for serving on my committee and for providing advice throughout my masters studies. Many thanks are also due to Dr. Gary Jeffress of the Blucher Institute and Ms. Aggie Armstrong of TAMU-CC Physical Plant Department, for their wholehearted support of my initial idea to carry out this project. James Kenefick of the Physical Plant Department is owed a great deal of credit for his help in the X-Windows/Motif programming and database development for the project.

It is often difficult to find such competent people as Dr. Patrick Michaud who are also so willing to share their knowledge and 'art'. Dr. Michaud deserves a hearty slap on the back for his friendship and guidance during this project, delivered in his characteristic good nature, at all hours of the night.

Special thanks go to the staff of the Conrad Blucher Institute and the Blucher Institute itself for financial assistance during my masters studies, and use of computing resources and survey equipment. Don Waechter of the Blucher Institute deserves special thanks for the many times he assisted in solving system-related problems and for sharing his late night food hampers. Mike Susey and Peter Curtis of Genasys II, Inc., also deserve honorable mention for their support for the project.

I save my most wholehearted thanks for my wife Loretta Cummings, who has been far more supportive than I thought possible and my greatest apologies to my daughters, Jessica and Katherine, for the many nights when "Daddy went back in to Blootch."
Table of Contents

Acknowledgements ................................................................. iv

List of Tables ........................................................................ vii

List of Figures and Plates ........................................................ vii

Abstract .................................................................................... viii

Section 1 - Introduction ............................................................ 1
  1.1 Project Rationale and Justification ...................................... 1
      The Campus Survey .......................................................... 1
  1.2 A GIS Primer .................................................................. 2
      Electronic Map Data Types ............................................... 3
  1.3 Use of GIS for Facilities Management ................................. 4
  1.4 Project Objectives ............................................................ 5

Section 2 - FMIS System Concepts ............................................ 6
  2.1 A Loosely Coupled Database/GIS Approach ....................... 6
  2.2 FMIS Maps ....................................................................... 8
  2.3 FMIS and Distributed Databases ....................................... 9
  2.4 Spatial and Attribute Data as an Information Resource  ...... 10
      Security Considerations .................................................. 11

Section 3 - FMIS Interface Concepts ........................................ 12
  3.1 General Interface Concepts .............................................. 12
  3.2 User Types ...................................................................... 12
  3.3 Map Display and Manipulation ........................................ 13
  3.4 FMIS Interface Operation ............................................... 14
      FMIS Interface Start-Up .................................................. 17

Section 4 - Overview of Genamap GIS Software ....................... 18
  4.1 General Concepts ........................................................... 18
  4.2 The Genamap Scripting Language .................................... 19
  4.3 The Genius Interface Builder ............................................ 21
      Single Command Entry in Scripts .................................... 21

Section 5 - The FMIS Map Database ......................................... 22
  5.1 Organization of Spatial Data ............................................. 22
      GIS Map Types ............................................................. 23
List of Tables

Table 5.1 - Subset of the FMIS map numbering scheme ........................................... 28
Table 7.1 - LP0171 placed in the generic attribute table ....................................... 36

List of Figures and Plates

Figure 1.1 - Types of Geographic Data ................................................................. 3
Figure 2.1 - The loosely coupled approach and the database access interface ........ 7
Figure 2.2 - The distributed database concept ..................................................... 10
Figure 7.1 - A traditional database table approach ................................................ 32
Figure 7.2 - Generic structure of the attribute database table ................................ 33
Figure 7.3 - Description code file structure ......................................................... 34
Figure 7.5 - Table lookups for LP0171 ................................................................. 37
Figure 8.1 - FMIS user interface components ...................................................... 41
Figure 8.2 - Display control menu structure ......................................................... 42
Figure 8.3a - FMIS main menu structure chart .................................................... 42
Figure 8.3b - FMIS main menu structure chart .................................................... 43
Plate 3.1 - Components of the FMIS Graphical User Interface .............................. 16
Plate 7.1 - FMIS screen showing display of attribute data for LP0171 ................... 38
Plate 8.1 - Spatial queries available in the FMIS ................................................... 49
Abstract

Texas A&M University-Corpus Christi will commence operation as a four-year degree institution in 1994. Operation of the campus as a four-year degree institution will heighten the need for efficient management, maintenance, and allocation of campus facilities. This report describes the design of a comprehensive Facilities and Management Information System (FMIS) for the university campus using Geographic Information System (GIS) technology. The FMIS described by this project greatly enhances access to information about campus facilities and aid efficient management of the campus. Database requirements are identified and a model for a database infrastructure that permits the greatest flexibility for a distributed database network is presented. The graphical user interface developed for the pilot FMIS implementation is described and sample applications for the retrieval of lighting circuits and facilities attribute data demonstrated.

The pilot FMIS implementation establishes a map database consisting of external lights, storm and sanitary sewer facilities, road network, buildings, grassed areas and a floor plan for the first floor of Corpus Christi Hall. Attribute data was input for sewer manholes, rooms in Corpus Christi Hall and external lighting facilities. These data are available for query and display by the pilot FMIS project and are used to demonstrate system capabilities.
Section 1 - Introduction

1.1 Project Rationale and Justification

Texas A&M University–Corpus Christi will commence operation as a four-year degree institution in 1994. This transformation from a two-year upper division level institution will lead to a dramatic increase in the workload of both administrative and maintenance staffs. The efficient maintenance, scheduling, and use of accommodation facilities, classrooms, laboratories, and service facilities will depend on the ability of these various departments to coordinate their resources and efforts. Success of the transition will also depend upon accurate information of the physical facilities.

The Campus Survey

The Conrad Blucher Institute for Surveying and Science (CBI), located on the campus of Texas A&M University-Corpus Christi, has undertaken a detailed topographic survey of the campus for the University administration. The purpose of this survey is to provide engineers and architects involved in new construction with accurate data for design and construction of new buildings. All data for this survey reside in an AutoCAD drawing on computers at CBI. The spatial accuracy of data collected by this survey is generally better than 0.05 feet.

In view of the pending increase in complexity of the university's facilities and added strain on both administrative and maintenance staff, this project was proposed to implement the logical and software infrastructure which would allow facilities management to be aided by a computer-
based combination of digital map graphics and relational database technology. These two technologies are collectively called a Geographic Information System (GIS). The system designed by this project is called the Facilities and Management Information System (FMIS).

1.2 A GIS Primer

A Geographic Information System can combine graphics in the form of digital maps with a relational database. This combination of graphics and data allows a user to identify an object on an electronic map displayed on a computer monitor and retrieve attribute data about that object from the database. This approach allows rapid retrieval of information and permits analysis of spatial relationships between features. The graphic nature of a GIS exploits the human affinity for maps and other visual data, generally results in a more user-friendly system than the traditional tabular database approach, and is capable of complex queries which are often transparent to the user.

The GIS concept is fundamentally similar to relational database technology. Features which comprise an electronic map are assigned a unique feature identifier or tag. This feature tag is then used as an index into an associated database. In one sense, the relationship between primary feature tag and database lookup that the GIS paradigm exhibits is similar to the relational database concept. This similarity arises because the user still selects data from the database but uses the geographic location of the feature instead of alphanumeric keys as the selection criteria.
Electronic Map Data Types

The fundamental data types of an electronic map are now introduced and explained. All geographic features can be classified into points, lines and polygons. Points and lines are relatively intuitive and can be illustrated through the examples of a fire hydrant or stop-valve for a point feature, and a sewer pipe or electrical line for a line feature. It is important to note the concept of an edge as it relates to a line segment. An edge consists of one or more line segments which form a connection between two points (or nodes). A polygon feature is of a sequence of connected line (or edge) segments which start and end at the same point and enclose an area. Figure 1.1 illustrates these three data types.

![Diagram of a point and a polygon made up of line segments](image)

**Figure 1.1 - Types of Geographic Data**

A geographic feature can be assigned a tag which indexes data in an external database. Using the above three data types, the initial notion of selecting a feature based on a specific geographic location can be modified to include a geographic selection criteria such as: "all features which lie within a specific soil-type or slope-class polygon," or "within five hundred feet of a particular pipe line." From the foregoing, it can be seen that addition of GIS map graphics can significantly
enhance the capabilities of existing databases by providing selection functions and analysis
techniques which are difficult to implement in a more traditional, alphanumeric, database. The
map data of the GIS is a spatial database and this concept will be further discussed in relation
to user views of available map data in Section 5.2.

1.3 Use of GIS for Facilities Management

The facilities of large institutions, particularly those of large engineering complexes such as oil
refineries, rely heavily on paper maps and plans of existing facilities. These plans show the
location of facilities such as pipes and are annotated with attribute data about these facilities
(e.g., pipe diameter or valve type). The maintenance and cataloging of these documents is
generally a significant undertaking for these organizations, and consequently, many companies
have begun to rely on GIS as a repository for facilities data. This industry is known as facilities
management (FM) and represents a branch of the GIS industry. The annual Automated
Mapping/Facilities Management (AM/FM) conference is devoted solely to use of GIS technology
for facilities management and attracted nearly 2,000 attendees in 1992. The need for such a
conference is indicative of the general acceptance among FM professionals of the usefulness of
GIS technology as a tool to support efficient use and maintenance of facilities infrastructures.

The qualities which make GIS such a useful tool for FM can be summarized as follows:

- Comprehensive cataloging of map and attribute data,
- Greater access to stored data through on-line terminals resulting in greater overall access
to the stored data and more informed decision making,
Centralized attribute and map database eliminating duplication of data recording and maintenance effort, and

Reduced reliance on recollections of long-term employees, resulting in a more comprehensive knowledge of the facilities infrastructure for short-term employees.

The foregoing benefits of the GIS approach to facilities management have prompted this study. The additional demand on university resources as it moves to a four-year degree institution will dictate an efficiency in resource management which can be catered by a GIS-based facilities management system.

1.4 Project Objectives

The objectives of this project are to establish a model for the implementation of the FMIS across the entire campus. In fulfilling this objective, the individual components of the FMIS are identified and the logical structure of these components established. A pilot FMIS with a Graphical User Interface (GUI) has been implemented to demonstrate the underlying design style and capabilities of the system. The pilot implementation and system design comprise the work carried out for this project. An objective of the pilot implementation project is to achieve a loosely coupled approach between the various components of the FMIS. This approach permits the greatest flexibility and lowest maintenance overhead for the fully implemented system.

This report describes the FMIS model and implementation as designed by this project and is divided into eight sections. The first seven sections describe the system components of the FMIS and the concepts behind each of these. Section eight describes, at the source code level, several
components of the system which illustrate the underlying design style. This design style is fundamental to the functionality of the system and ultimately defines the flexibility, ease of maintenance, and extendability of the system.

**Section 2 - FMIS System Concepts**

2.1 **A Loosely Coupled Database/GIS Approach**

Most available GIS packages are supplied with interface software that permit functions within the GIS to make queries of an external database. Most GIS software directly interface to the (currently) popular relational database products. However, there are several disadvantages in relying on this link:

- Reliance on GIS software vendor-supported links to third party databases. The database interfaces are generally implemented at the binary code level and may change with each release of the database software. This requires continual code modification by the GIS software vendor to retain database compatibility.

- Restriction to the subset of queries, query structures, and file structures supported by the GIS software vendor.

- Duplication of database access and update routines.

The last point requires explanation since it is an important part of the FMIS concept. Recall that the two fundamental components of a GIS are spatial (map) data and attribute data stored in an external relational database. Much attribute data can be added to the database without the need for the graphics capability of the GIS. Constraining all database access to be performed by the
FMIS is an inefficient use of expensive graphics resources. It follows that it is desirable, if not mandatory, to allow data entry and update routines of a FM system to exist both as part of the integrated FMIS and as applications which access the database independently of the GIS. In order to give users a similar interface for both situations, it is necessary to develop database access routines which rely on neither the GIS software nor the database software for functionality and "feel." The FMIS has achieved database and GIS interface independence by creating a separate X-Windows/Motif application to provide an intuitive, scrollable display for database query and update functions.

![Database Access Interface Diagram](image)

**Figure 2.1 - The loosely coupled approach and the database access interface**

The database access interface mentioned above accesses the database manager directly and returns data to the requesting process via a pipeline or file created by the database manager. This results in a loosely coupled approach as depicted in Figure 2.1. The FMIS seeks to adhere
as closely as possible to a loosely coupled approach for all FMIS components since it gives the final system the greatest flexibility and standardizes the user interface to the database.

2.2 FMIS Maps

As mentioned in Section 1.2, the available map data represents a spatial database which can be queried, updated, and displayed. The idea that map data comprise separate databases is further supported by the notion of "layering" as used in a GIS. The layering approach attempts to segregate map data into logically related classes based on the type of feature. This segregation is accomplished through use of separate maps that cover the same geographic area but are comprised of different physical features. For example, all data relating to lighting should be limited to a specific map or group of maps. The layering of map data allows display and query of only the data which are relevant to a user's current work needs and can be considered a "view" of the spatial database.

Extending the notion of database views for map data, it follows that individual users should have the ability to define their own relevant view(s) of the map database. Specific user views of the map data are necessary because of the vast amount of spatial data that will eventually be part of the map database. For example, maps relating to sanitary sewer and gas facilities would hinder administrative staff performing classroom allocation. The need for views, or subsets, of the map database is of great importance in lowering the perceived complexity of the system. The FMIS addresses this concern through the use of mapset files. These files effectively
implement a view into the map database and permit users to either define unique views or to use one of several predefined views. The map set concept is discussed in detail in Section 5.2.

2.3 FMIS and Distributed Databases

The loosely coupled approach to the GIS/database relationship introduced in Section 2.1 permits access to virtually any database product. The requirements for access to other databases are knowledge of the database file structure(s) and the means to return data from the database(s) to the GIS. The latter requirement is accommodated in our loosely coupled approach which relies on either a pipeline or file creation for the data transfer between the external database and the GIS, and the former requirement is accommodated by reference to appropriate data dictionaries during application development. Under this more generalized approach, the FMIS database can be expanded to a distributed database network which can accommodate virtually all data relating to the campus, including classroom scheduling and room allocation, student and staff records, courses available, and physical inventory. The advantage of a distributed approach is that the database network becomes an information resource for all administrative and physical facilities management. The distributed database idea as it relates to the FMIS is summarized in Figure 2.2 and will be discussed further in the following section.
2.4 Spatial and Attribute Data as an Information Resource

In reviewing the FMIS developed thus far, it is important to realize that the system represents far more than a solution to a specific set of reporting and mapping problems. The database proposed by this project will become an all-encompassing information repository for data relating to the campus. While the implementation demonstrates some real and specific solutions
to current facilities management problems experienced on campus, specific problem solution has not been the focus of this project. The solutions developed as part of this project are presented to illustrate the advantages and flexibility of a comprehensive system such as the FMIS.

The FMIS reduces duplication of data collection and maintenance. Once the database responsibilities of each department are defined in relation to the whole FMIS, overall database collection and maintenance efforts will be minimized and the usual problems of data redundancy eliminated. The distributed database approach will result in greater access to data which will lead to more informed decisions for both facilities management and strategic and administrative planning.

Security Considerations

A distributed database is not without potential security problems for confidential information such as personnel and student records. Confidential records can be protected using a combination of Login ID and a DBMS's own access control measures to regulate read/write permissions on individual database tables. Notwithstanding the need for confidentiality of some data sets, the entire distributed database system should remain as open as possible in order for the full benefits of greater information access to be realized.
Section 3 - FMIS Interface Concepts

3.1 General Interface Concepts

The guiding principles of the FMIS interface place the user friendliness of the system as the primary goal. System usage can generally be measured by how much users "like" using the interface. The perceived success of the system can sometimes exclude issues relating to data integrity. The most desirable combination is obviously both high data integrity and ease of use. The pilot interface implemented as part of this project demonstrates that both data integrity and user friendliness can be compatible. In order for new and casual users to quickly gain a useful level of proficiency, the interface is intuitive and adheres to a common logic in the way functions are executed and arranged on the screen.

3.2 User Types

In designing the user interface, there are inevitably decisions which relate to the expertise of different users who will be accessing the system. Given the all-encompassing nature of the FMIS, it is probable that the majority of users will have little or no experience in GIS and for the purposes of this project are classed as "casual" users. The remaining users will be relatively well versed in system functions and GIS and can be classed as "expert" users.

The challenge for the FMIS interface is to provide enough predefinition in commands to the allow a casual user to do productive work while still providing enough flexibility for the expert user to fully exploit the power of the system. The FMIS interface piloted by this
project provides for both levels of expertise. By predefining many basic FMIS functions and not requiring command options to be specified, the casual user is able to become acquainted with, and productively use, the system relatively quickly. The FMIS interface includes a "terminal window" into which an expert user can enter commands directly into the GIS and circumvent the menu system altogether.

3.3 Map Display and Manipulation

Two general ideas must be introduced at this point in order to understand the underlying logic of the pilot interface. These are the concepts of an active layer and a feature set.

Users have the ability to interactively select which maps they wish to view as part of their work session. The GIS does not assign priority to one map over another, and in order to search for a particular primary feature tag, the GIS requires the name of a map to be searched. The pilot interface uses the idea of an active map or layer to specify which of the currently displayed maps should be used in map query functions. The user can interactively select the map to use as the active layer, and only one layer can be active at any time.

During GIS queries and operations on different map features, it is often necessary to define a subset of the available map features to apply a particular command or query to. A feature set is a subset of map features that can either be defined by individual selection or returned as a set from the results of a complex query or geographic selection. An example of a feature set is a group of light poles which all reside on the same electrical circuit. Once created, feature sets
can be saved to a file and used as input to subsequent commands which operate on subsets of map features.

The FMIS pilot interface includes routines for selecting maps to be displayed, selecting the active layer, creating feature sets, manipulating the map display, and panning and zooming to different areas of the geographic map extents. Buttons are arranged in three functional groups:

♦ General, menu-activated functions that manipulate and display maps and feature sets, execute spatial queries on the displayed maps, and change the units of data output from GIS reports,

♦ An applications menu for the inclusion of external applications which may be executed from within the FMIS, and

♦ A display control menu which allows panning, zooming, and changing the geographic extent of displayed maps.

3.4 FMIS Interface Operation

Discussion is now directed to operation of the interface developed by this project. A discussion of the tools used to build the interface can be found in Section 4.3. Plate 3.1 is an example screen from the FMIS interface and should be viewed in conjunction with the following descriptions of the interface components. The numbers in parentheses appear on the interface screen and help identify the individual components. The FMIS interface consists of several functionally discrete elements:
(1) A main graphics window for display of map data.

(2) An FMIS main menu. This menu hosts the various predefined functions mentioned in Section 3.2. These functions include: selection of maps to display, selection of the current mapset file, (mapsets are discussed in Section 5.2), settings relating to currently displayed units, selection of text maps for display, selection of maps to be labeled, selection of the active layer to be used in query commands, spatial and attribute query of maps and hardcopy output.

(3) An applications menu. This menu launches programs which may include external applications such as an interactive classroom scheduling module, other FMIS applications, or external database applications.

(4) A display control menu. This menu permits the user to change the geographic extents of the display.

(5) A terminal window. This window is used for direct, command line interaction with the GIS. An expert user can take advantage of the full system functionality by entering specific GIS commands through this window.

(6) A status window. This window is used for displaying the current settings for the system. These include: current active layer, map set file, output units for linear and area reports, scale of the currently displayed maps and the current search criteria.
Plate 3.1 - Components of the FMIS Graphical User Interface
FMIS Interface Start-Up

Upon entry into the FMIS, users are presented with a view which consists of the area surrounding the university island. The maps from the previous work session are displayed unless it is a new work session, in which case a default set of maps are displayed showing Ward Island, campus buildings, and campus roads. This view of the FMIS is shown in Plate 3.1. The FMIS user interface serves primarily as a data query interface and includes an array of buttons to permit the user to select and display different maps, change settings for displayed data, and change the active layer. All commands are entered into the interface by clicking the mouse on an interface button and this relieves the user of responsibility for the underlying routines which are invoked to perform an FMIS function.

The interface includes a terminal window which allows the user to enter a command into the GIS directly. This is included for expert users who require access to lower level routines to complete their work. It is expected that frequent users will progress toward expert usage and the terminal window. As the user base of the FMIS grows, the interface can be expanded to meet the demands and requests of the users. The ease of changing the FMIS interface and the flexibility of the whole system will make modification of the interface a relatively easy task for the system FMIS administrator.
Section 4 - Overview of Genamap GIS Software

4.1 General Concepts

The FMIS project uses GIS software developed by Genasys Systems Inc., Fort Collins, Colorado. This software, called Genamap, uses a "toolbox" approach to GIS. Genamap provides a full and comprehensive set of GIS functions for the display, manipulation, and editing of spatial data. Genamap also provides functions which can be used to select items from a displayed map based on logical and spatial expressions.

Commands are entered into Genamap through the Genamap Shell which is modelled after the UNIX Bourne Shell. The Genamap Shell interprets the entered commands and calls the appropriate program to execute or act upon the data. The shell approach to command entry permits access to all available routines within Genamap; however, it is an inefficient user interface. Most commands within Genamap require several options to be passed as parameters to the program, and users generally must execute several commands in a specific sequence to achieve desired results. Consequently, command line entry results in a great deal of typing for even the most rudimentary operations. From the foregoing, it can be seen that Genamap, in its most basic form, represents a toolbox of primitive functions which can be combined and executed to achieve a specific result. Genamap also supplies a scripting language and an interface/application building tool called Genius; these are now discussed.
4.2 The Genamap Scripting Language

As mentioned previously, the Genamap scripting language is modelled on the UNIX Bourne Shell. Script source-code files are standard text files containing Genamap commands. Once a script has been called, each program line is sequentially interpreted and executed by the Genamap Shell. The scripting language supports the following features:

♦ Logical testing of data through an "if..then..else.." construct which includes compound tests,

♦ Assignment of function results to local variables,

♦ Looping capability through "while..do" and "for" constructs,

♦ Basic arithmetic functions which support floating point data types,

♦ String concatenation and manipulation commands, and

♦ The ability to pass variables from within the Genamap Shell to an external UNIX command and redirect the output from either Genamap or external UNIX commands to either a UNIX system file or as input to another Genamap command.

The capability of Genamap to execute UNIX commands provides a great deal of power to the language and represents a well-conceived approach to the problem of providing language functionality. Rather than attempting to implement complex instructions and functions, the software architects have implemented I/O redirection and rely on standard UNIX tools. For example, it is common for Genamap Shell scripts to redirect output from a Genamap command to *awk, sed, perl, tr*, or *cut*. The output from a UNIX command may then be redirected into the Genamap Shell as the arguments to a subsequent Genamap command. I/O redirection capability provides the Genamap Shell with an extensive set of data manipulation functions (those
supplied by UNIX) and reduces the learning curve for users who are already familiar with UNIX.

Using the control and I/O redirection capability of the scripting language, it is possible to combine Genamap commands into sophisticated applications that call many of the Genamap primitive functions but are transparent to the user. For example, when a user selects the "Zoom Window" button from the display control menu, a script is executed which performs the following:

- Appends the geographic extents of the currently displayed window to a window list file (this allows the user to restore a previous view),
- Calls the Genamap Zoom function to allow the user to define the new geographic window,
- Calls a script to redraw the screen based on four files which define which maps to draw and in what order they should be shaded and plotted.

The only input requested from the user following the initial button press is specification of the new geographic window with the mouse.

The pilot application developed by the FMIS project relies heavily on the Genamap scripting language and its I/O redirection capabilities. The overall design style of the pilot application tends toward use of existing UNIX tools to achieve data and parameter reformatting. This results in elegant solutions for several parts of the project and some of these are discussed in Section 5.2.
4.3 The Genius Interface Builder

The Genius interface builder, distributed with Genamap, permits interactive design of a graphical user interface (GUI). This tool falls into the category of a CASE tool for application development. Genius allows the user to call a variety of predefined widgets including:

- Scrolling selection lists,
- Button and pulldown menus,
- Text entry and text display forms, and
- Terminal windows which can be used to enter commands directly into the Genamap Shell.

Genius permits the user to define and name buttons, scrolling lists, or pulldown menus and assign to each a series of Genamap commands. When the user activates a widget, the command string associated with that widget is appended to the command string of the parent widget. Once menu and button selection is complete, the command string which has been constructed by selection from the various menus is executed by the highest level menu. The concatenation of command strings and results of intermediate menus permits the user to define not only commands to be executed but also the arguments to these Genamap commands.

Single Command Entry in Scripts

Constructing Genamap command sequences in the above manner provides a convenient way of eliminating the need for users to have detailed knowledge of command syntax. However, the disadvantage of this approach is that the interface widgets are responsible for storing the
command sequences. If the menu interface is changed, the associated command sequences must be migrated to a new widget. This creates programming development overhead during interface development/update; however, the FMIS alleviates this problem by restricting each menu item to execution of a single Genamap script. Constraining each menu item to single command execution decreases the interface flexibility because the user cannot build a command sequence as interactively as they potentially could. However, recall from Section 3.3 that the interface needs to accommodate the casual user and that this is accomplished by restricting the options available during interface interaction. The casual user consequently experiences no significant degradation in interface functionality from the single command restriction. The primary benefit of the single command approach is that interface and script files are easier to maintain because all commands exist as either Genamap system commands or script files; command sequences are not stored in the Genius interface files. A parallel which can be drawn from the single command approach is that it represents a looser coupling between the interface and the command execution, and this is a desirable characteristic and goal of our design.

Section 5 - The FMIS Map Database

5.1 Organization of Spatial Data

Recall from Section 2.2 that map data within the spatial database is layered into logically similar feature classes. The FMIS establishes the following basic classes of physical objects:

- Lighting (Internal and External),
- Sanitary Sewer,
♦ Storm Sewer,
♦ Telecommunications,
♦ Computer Communications,
♦ TV/Cablevision Links and Multimedia,
♦ Electrical,
♦ Gas,
♦ Water, and
♦ Miscellaneous.

**GIS Map Types**

There are many different map types used by Genamap; this project uses three types: vector maps (type 4), text maps (type 10), and discrete cell maps (type 7). The map type values in parentheses is a scheme used by Genamap to identify the type of data stored in map files. Example data types are as follows:

♦ Type 4 Vector Maps - Spatial data in this map type, as its name implies, is represented by a series of vectors in a defined Cartesian coordinate system. These maps comprise the majority of the maps that will be created for the FMIS project.

♦ Type 10 Text Maps - A text map consists of text strings that are attached to a particular geographic location. For example, text maps are used in the FMIS to annotate bay, building, and road names.
Type 7 Discrete Cell Maps - Spatial data in a discrete cell map consisting of raster data which can be displayed on the screen. FMIS cell maps are used to display scanned aerial photographs of the campus which can then be overlayed with vector data.

FMIS Maps Created
The FMIS has created vector maps for the following layers: external lighting, sanitary sewer, storm sewer, internal walls of Corpus Christi Hall (level 1), computer connections for Corpus Christi Hall (level 1), roads, grassed areas, and the shoreline of Ward Island and part of Oso Bay. All data for these maps have been imported from the AutoCAD drawing files for the campus survey (see Section 1.1) with the exception of the shoreline which was digitized from a U.S. Geological Survey Quadrangle Sheet with a scale of 1:24,000. The methodology employed by the campus survey dictated the creation of similar feature layers to those subsequently created for the FMIS, and this assisted the migration of the survey to the FMIS. Features were exported from the campus survey drawing in AutoCAD DXF format by feature layer. Genamap provides commands for reformatting DXF files; these were used to create Genamap import file(s) from the campus survey DXF files. Each layer exported from AutoCAD was used to create a single map in the FMIS. Common feature types (point, line, and polygon) for each of the map layers mentioned previously were grouped into the same map.

Unique Feature Tags
Unique feature tags were automatically assigned to point features by use of a routine written in awk that reformats the master coordinate file from the campus survey to a Genamap import file.
It should be noted here that the general organization of the campus survey greatly aided the creation of the spatial database for the FMIS and is, itself, a good example of the way survey work should be carried out in order to allow subsequent population of a spatial database for GIS. Features in the campus were assigned unique feature tags at the time they were initially surveyed. These feature tags consist of an alphabetic prefix which designates the feature type and a numeric suffix which designates a sequence number within a particular feature class; e.g., LP0171 designates light pole number 0171. Each light on the campus was assigned the light pole number on its base as its primary identifier. All utility control structures (stop-valves, meters, etc.), storm sewer inlet structures, and sanitary sewer manholes were assigned unique identifiers. Some features were renamed as part of the transfer to the FMIS to provide prefixes more commonly used in standard engineering practice. The advantage of assigning identifiers at the time of survey is that all features carry the identical name from the time of survey through to the final map database. This permits tracking a feature's spatial history. For example, in the FMIS it is relatively easy to retrieve the original survey observations for a campus light pole including the date and time that it was originally surveyed.

Line features have been tagged manually, and this is one aspect of the initial survey which can be improved to aid transfer of data to the FMIS. These data are difficult to uniquely code in the field because line segments are often surveyed at different times and the attribute data of all parts of a particular line segment are often unknown at the time of survey. For example, a sub-surface electrical conduit may be located on the ground and indicated as a straight line between two node features such as a pair of light poles. The actual line may include a junction box,
however, which changes the conduit type to a different class along the route of the line. In this situation, the conduit should be represented in the FMIS as two distinct line segments to permit a user to resolve the change in conduit type during query operations.

5.2 FMIS Map Naming Conventions

Genamap restricts map names to a total of eight characters, making it difficult to define meaningful map names for many of the maps. The FMIS has overcome this problem by using several ASCII system files to catalog the available maps and associate a description with each map. This convention forms a fundamental part of the FMIS map management system and illustrates the reliance on UNIX tools as a basic component of the FMIS design style.

FMIS Map Management and Numbering Scheme

Recall from Section 5.1 the different layers created as part of the spatial database. One of the functions of the FMIS will be to provide floor plans of buildings on the campus. In order to manage all maps effectively and permit the implementation of user views of the map database, a scheme was developed to catalog maps based on a map number. This number forms part of a general map description associated with each map name and is stored in a master maplist file in an FMIS project directory. This file, called Maplist_Master, consists of the following data:

mapname|nnnn.map_description|drawing_priority

The mapname field is the name assigned to the map in Genamap, and, as mentioned previously, is restricted to eight characters in length. The map description field consists of a description of
the map and is essentially used to alias the eight character map name permitted by Genamap. The Maplist.Master file for the FMIS implementation can be found in Appendix A. The important part of this description is the first 4 characters which are separated from the remainder of the description by ". . ". These numbers identify the type of FMIS data stored in the map. Specific numeric ranges have been defined for each layer created by the FMIS. Table 5.1 presents a subset of these ranges and a full listing of ranges defined by this project are given in Appendix C. The possible range of values for the map number is from 0000 to 9999. The first 100 map numbers are reserved for general administrative maps such as the Oso Bay/Ward Island map and road and bay names text maps. Additional maps can be added to this range but they should be "miscellaneous" maps that do fall into any of the remaining map number categories. Maps 0100 through 0299 are reserved for the external facilities of the campus. This range is further subdivided into blocks of 20 maps for each FMIS feature layer. Maps over 300 are reserved for buildings on the campus with a block of 100 maps for each building. Within each block, the layers for a building are organized by tens into feature groups.

It can be seen from Table 5.1 that this method of numbering maps permits the third character of the map number to be used to resolve identical map layers across multiple buildings. For example, the lighting maps for Corpus Christi Hall all lie in the range 0300 to 0309. The lighting maps for the Administration building and the Library lie in the ranges 0500 to 0509 and 0400 to 0409 respectively.
<table>
<thead>
<tr>
<th>Map Number Range</th>
<th>General Category</th>
<th>Description of Maps in this Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001 - 0029</td>
<td>Administrative</td>
<td>Ancillary Maps such as Oso Bay, Survey Control</td>
</tr>
<tr>
<td>0030 - 0059</td>
<td>Administrative</td>
<td>Annotation Text maps for Administrative Maps</td>
</tr>
<tr>
<td>0060 - 0099</td>
<td>Administrative</td>
<td>Miscellaneous Administrative Maps</td>
</tr>
<tr>
<td>0100 - 0119</td>
<td>External Facilities</td>
<td>Lighting</td>
</tr>
<tr>
<td>0120 - 0139</td>
<td>External Facilities</td>
<td>Sanitary Sewer</td>
</tr>
<tr>
<td>0140 - 0159</td>
<td>External Facilities</td>
<td>Storm Sewer</td>
</tr>
<tr>
<td>0160 - 0179</td>
<td>External Facilities</td>
<td>Telecommunications</td>
</tr>
<tr>
<td>0180 - 0199</td>
<td>External Facilities</td>
<td>Computer Communications</td>
</tr>
<tr>
<td>0200 - 0219</td>
<td>External Facilities</td>
<td>TV/Cable Television</td>
</tr>
<tr>
<td>0220 - 0239</td>
<td>External Facilities</td>
<td>Electrical</td>
</tr>
<tr>
<td>0240 - 0259</td>
<td>External Facilities</td>
<td>Water</td>
</tr>
<tr>
<td>0260 - 0279</td>
<td>External Facilities</td>
<td>Gas</td>
</tr>
<tr>
<td>0280 - 0299</td>
<td>External Facilities</td>
<td>Miscellaneous/Text maps</td>
</tr>
<tr>
<td>0300 - 0309</td>
<td>Buildings (CCH)</td>
<td>Lighting</td>
</tr>
<tr>
<td>0310 - 0319</td>
<td>Buildings (CCH)</td>
<td>Sanitary Sewer</td>
</tr>
<tr>
<td>0320 - 0329</td>
<td>Buildings (CCH)</td>
<td>Misc. Maps</td>
</tr>
<tr>
<td>0330 - 0339</td>
<td>Buildings (CCH)</td>
<td>Telecommunications</td>
</tr>
<tr>
<td>0340 - 0349</td>
<td>Buildings (CCH)</td>
<td>Computer Communications</td>
</tr>
<tr>
<td>0350 - 0359</td>
<td>Buildings (CCH)</td>
<td>TV/Cable Television/Multi Media</td>
</tr>
<tr>
<td>0360 - 0369</td>
<td>Buildings (CCH)</td>
<td>Electrical</td>
</tr>
<tr>
<td>0370 - 0379</td>
<td>Buildings (CCH)</td>
<td>Water</td>
</tr>
<tr>
<td>0380 - 0389</td>
<td>Buildings (CCH)</td>
<td>Gas</td>
</tr>
<tr>
<td>0390 - 0399</td>
<td>Buildings (CCH)</td>
<td>Internal Walls</td>
</tr>
</tbody>
</table>

Table 5.1 - Subset of the FMIS map numbering scheme

The map number provides a convenient method to implement a user view into the map database. Using UNIX regular expressions and the `grep` command, all maps with a particular
pattern in their numeric fields can be extracted from the master maplist file. In conjunction with the other UNIX commands available to Genamap scripts, the output from this grep command can be either piped into another process such as cut (to remove the Genamap map name from the output lines) or redirected to a file.

The extension of this idea has given rise to mapset files which implement user views into the map database. A mapset file is a standard text file of regular expressions which are applied to the master maplist file using egrep. The output from this command returns a list of files forming a user view of the map database. The regular expression approach to the implementation of the user view into the map database provides a powerful method to specify file names for individual users. For example, an electrician may wish to have only the lighting layer available for selection in the map database view. The mapset file for this view would consist of the following lines:

```
^.[3-9]0.
^01[01].
```

The first of these regular expressions specifies all maps of lighting data for each building in the master maplist file, and the second line specifies the lighting data maps for the external facilities of the campus.

Four mapsets are predefined by the FMIS: Administration, Maintenance, All, and Faculty. A user can interactively define a custom mapset file by selecting individual maps from a list, resulting in the addition of a map name to the list of regular expressions in the user's mapset file. While this map name is not as elegant as a "true" regular expression as outlined above, it
effectively provides system functionality and interactive capability to the FMIS. It can be seen from the foregoing that user views into the map database are implemented in a sophisticated yet simple manner and rely on well established and powerful UNIX tools. It is this approach which characterizes both the Genamap scripting language and development of the FMIS.

Section 6 - Overview of GIS/Database Requirements

The purpose of this section is to discuss the database requirements of the GIS. These requirements must be identified to ensure that the needs of the FMIS will be adequately met. The database component of the FMIS acts as the repository for attribute data of the spatial database. Typically the GIS makes requests for attribute data from the database manager using the primary feature tag of a map object as the key into a database table. The attribute data are then passed back to the GIS for use in subsequent commands. GIS requests of the database manager are generally standard SQL statements of the form:

```
SELECT field FROM table WHERE primary_key = feature_tag
```

This form of SQL statement is relatively simple and usually few compound statements are needed to retrieve requested data. As a consequence, the SQL functionality of the database for the FMIS implementation is accommodated by most commercially available databases and requires few, if any, of the SQL language extensions that are available. The FMIS uses Ingres relational database software for storage of attribute data. Ingres supports ANSI-SQL and consequently accommodates the query requirements of the FMIS. Ingres controls access to database tables via login identifier, providing a great deal of flexibility for database security. While most users will have access to all campus attribute data, it may be necessary to restrict
access to portions of this data. The security features of the Ingres DBMS provide this protection.

Another requirement of the FMIS attribute database is the ability to dynamically add attribute fields for a map feature. This requirement is necessary because of the dynamic nature of facilities management attribute data. For example, a supplier may change the model number of a light pole component dictating the addition of a previous model number field in the light pole attribute database. While it is possible to identify many of the required attribute fields for most map features during database design, it is unlikely that all possible fields will be identified. A requirement of the database system is a structure that permits the system to be dynamic with minimum overhead for the database administrator(s).

Section 7 - The Ingres Attribute Database

The observations from Section 6 regarding the complexity of the GIS/database relationship, and the necessity for a database structure that is easy to edit, has prompted a detailed review of the structure of traditional relational database tables. The following sections review the problems associated with implementing an attribute database using the traditional, field-oriented tabular approach and discuss the approach used by the FMIS to implement the attribute database.
7.1 Traditional Field-Oriented Database File Structures

Changing the structure of databases table is by no means a new problem for database administrators. However, the data of the physical (spatial) world are constantly changing both in data value and the type of data available. Adding a new attribute field in a traditional database structure such as that shown in Figure 7.1 would typically require the following steps:

♦ export of the data to a system text file,
♦ creation of a new database table which would include the new field(s), and
♦ loading the newly created database table with data from the system text file.

<table>
<thead>
<tr>
<th>Tag</th>
<th>Height</th>
<th>Bulb</th>
<th>Voltage</th>
<th>Breaker</th>
<th>Control</th>
<th>Last Srvc.</th>
<th>Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>LP0171</td>
<td>30'</td>
<td>400W</td>
<td>277</td>
<td>FCC157</td>
<td>Timer</td>
<td>12/18/92</td>
<td>EKG/401..</td>
</tr>
</tbody>
</table>

Figure 7.1 - A traditional database table approach

The structure of this table is essentially static since any change to the number or type of fields in the table is relatively expensive in terms of both system resources and administrative overhead. This is not compatible with our goal for a database structure that can be updated easily and does not accommodate the dynamic nature of spatial data particularly well.

A second concern with the structure shown in Figure 7.1 relates to date fields for attribute data. An object such as a light pole is made up of separate components, each of which may be updated, replaced, or destroyed during maintenance activities. If the table structure in Figure 7.1 is used, there is no convenient method to record historical data for each component of a
feature without resorting to additional tables which deal specifically with that component and includes associated date fields.

7.2 The FMIS Attribute Database Approach

From the previous section, it can be seen that traditional field-oriented database structures exhibit some real problems for the GIS database administrator. The FMIS uses a different database structure which effectively permits adding new data fields to the attribute database table with significantly less maintenance overhead than would be required by a more traditional database structure. The structure also includes a date field for each attribute to allow historical data to be recorded for all components of a map feature. The structure uses the fact from Section 6 that most queries for retrieval of attribute data are relatively simple SELECT statements with few compound, JOIN, or UNION statements. This database structure is now discussed.

The database tables implemented by the pilot FMIS project use a generic structure which consists of the fields shown in Figure 7.2. The field specifications can be found in Appendix E. Columns from the original database table, shown in Figure 7.1, now exist as records and each attribute (record) has associated starting and ending dates and conditions.

<table>
<thead>
<tr>
<th>Map Name</th>
<th>Tag</th>
<th>Start Date</th>
<th>Start Condition</th>
<th>End Date</th>
<th>End Condition</th>
<th>Description Code</th>
<th>Value Field</th>
</tr>
</thead>
</table>

Figure 7.2 - Generic structure of the attribute database table
The values of each attribute field from the original database table are now stored in the value field of the generic structure. The information that this value represents is determined by semantic interpretation of the description code field from records in a separate database table. The description code table has the structure shown in Figure 7.3 and the field specifications can be found in Appendix E.

<table>
<thead>
<tr>
<th>Description Code</th>
<th>Process Field</th>
<th>Grouping Code</th>
<th>Abbreviation</th>
<th>Text String</th>
</tr>
</thead>
</table>

Figure 7.3 - Description code file structure

The description code field of the attribute table is a key into the description code table. When an application retrieves a record for a feature from the attribute table, it also retrieves the record specified by the description code field from the description code table. The process field of the description code record determines the semantics of the value field in the attribute record. The process field currently defines three different types of data in the value field of the attribute table: literal, description and process. A literal field, as its name implies, holds a literal data value. The data value is consequently displayed and the text string from the description code table used to annotate the display and thus describe what the data actually represents. A description field specifies that the value field of the attribute record is a second index into the description code table. This is useful in situations where a string of literal text is repeated many times as the attribute of a particular feature class; for example, the model number of a light pole. Rather than repeat the text, a second entry is made in the description code table and the index of this entry is used to retrieve the text string. An example of this approach for LP0171 is
given later. This method of referencing text can be used for virtually any items which repeat throughout the attribute table. It is currently used in the attribute table to reference surface materials, model numbers, colors, descriptions of circuit breaker types, and light bulb descriptions. A process field specifies that the program specified in the text string field should be executed and the results of this process returned to the application as the data value to be displayed. The data in the value field of the attribute record are used as arguments to the called program. This field is currently not used in the pilot FMIS but will give the system a great amount of flexibility.

The grouping code field allows descriptions which form part of a logical set of descriptions to be displayed together as a scrollable list. A scrolling list of available descriptions is displayed during attribute edit and entry. If the user chooses to add a new attribute to the records for a selected feature, a scrolling window of currently used attributes for that particular feature class is displayed. If the user wishes to choose from all available descriptions in the description table, they may also do so. Due to the potentially large size of this table, the list of displayed descriptions are grouped according to their grouping code.

The abbreviation is a text string which may be entered by the user to identify a specific description code without the need to resort to a scrolling list. It is anticipated that some users will be responsible for a specific section of the data entry or deal exclusively with one section at a time. This field has been included to speed data entry for these users.
To further illustrate the differences between the two database approaches, consider the attribute data for LP0171 from Figure 7.1. This data now resides in the generic database structure and Table 7.1 shows the attribute data returned from an SQL SELECT command of the attribute table. This table should be viewed in conjunction with the descriptions database table in Appendix F. Figure 7.5 shows the table lookups which result in the display of the data for LP0171. The corresponding FMIS screen display after retrieving these attributes can be found in Plate 7.1.

<table>
<thead>
<tr>
<th>Map Name</th>
<th>Tag</th>
<th>Start Date</th>
<th>Start Condition</th>
<th>End Date</th>
<th>End Condition</th>
<th>Description Code</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>180</td>
<td>LP0171</td>
<td>930907</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>21</td>
<td>30</td>
</tr>
<tr>
<td>180</td>
<td>LP0171</td>
<td>930907</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>2</td>
<td>400</td>
</tr>
<tr>
<td>180</td>
<td>LP0171</td>
<td>930907</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>3</td>
<td>277</td>
</tr>
<tr>
<td>180</td>
<td>LP0171</td>
<td>930907</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>179</td>
<td>FCC157</td>
</tr>
<tr>
<td>180</td>
<td>LP0171</td>
<td>930907</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>151</td>
<td>152</td>
</tr>
<tr>
<td>180</td>
<td>LP0171</td>
<td>930907</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>24</td>
<td>147</td>
</tr>
<tr>
<td>180</td>
<td>LP0171</td>
<td>930907</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>22</td>
<td>8</td>
</tr>
<tr>
<td>180</td>
<td>LP0171</td>
<td>930907</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>23</td>
<td>24</td>
</tr>
<tr>
<td>180</td>
<td>LP0171</td>
<td>930907</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>149</td>
<td>-</td>
</tr>
<tr>
<td>180</td>
<td>LP0171</td>
<td>930907</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>156</td>
<td>144</td>
</tr>
<tr>
<td>180</td>
<td>LP0171</td>
<td>930907</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>157</td>
<td>145</td>
</tr>
<tr>
<td>180</td>
<td>LP0171</td>
<td>930907</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>25</td>
<td>146</td>
</tr>
<tr>
<td>180</td>
<td>LP0171</td>
<td>930907</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>155</td>
<td>148</td>
</tr>
</tbody>
</table>

Table 7.1 - LP0171 placed in the generic attribute table

From Table 7.1 it can be seen that the database table is significantly different from the structure of the traditional table shown in Figure 7.1.
Figure 7.5 - Table lookups for LP0171
To clarify use of the description code table, the following example is presented. The model numbers for the external light poles are quite long, and most of the external lights have the same model number. Rather than create a separate table of model numbers, an entry is added to the description code table (see entry number 147, Appendix F). When an application returns the description code record, it determines from the process field that the value in the attribute record is a second description code. It subsequently retrieves the second description code specified by the value field of the attribute record. The text string field of the second description code is used as the displayed data value and the text string of the first description code is used to describe the data as before. This example is similar to that shown in Figure 7.5.

Attribute Database Discussion

The most important benefit of the attribute database approach employed by the pilot project is flexibility. The addition of a new attribute for a particular feature requires simply the addition of a new record to the description code table, instead of a change to the attribute table structure. This not only permits the addition of new attributes dynamically, satisfying a design objective of the FMIS database, but also provides complete control over which attributes are recorded for each feature. The ability to record date information about each attribute provides the system with an historical database and satisfies the design objective which relates to date fields and attribute records. Such an historical record is mandatory for statistical analysis both to analyze equipment failure rates and establish maintenance schedules based on these failure rates.
There are obviously some potential considerations regarding the overall efficiency of the generic database structure as employed in the pilot FMIS. In effect, a SELECT statement looks for multiple occurrences of a non-unique primary key. The pilot FMIS has not detected any significant performance problems in the database component. The attribute table presently comprises 1600 records. By way of comparison, a similar Ingres table held on CBI computers has in excess of 4.5 million records and exhibits no significant performance problems.

Section 8 - An Overview of Implemented Software

The purpose of this section is to provide a general description of the user interface, and overview some components of the pilot implementation. The first part of this section describes the hierarchical structure of the user interface, and this is followed by an in-depth explanation of the SQL link to the database for retrieving lighting circuits data. Following this is an explanation of the data reformatter developed to speed population of the attribute table and a brief overview of the spatial queries implemented for the FMIS. The section concludes with conclusions and comments regarding the system design.

8.1 User Interface Structure

The user interface is described in detail in Section 3.3 of this report. What follows is a structure chart which shows the menu hierarchy and commands executed from each widget on the menu. Note that it is a trivial exercise to change the structure of a menu or the commands executed by a menu item since all are created in the Genius environment. While
layout of these menus has placed user friendliness as the primary consideration, they are easy to modify if applications or usage dictate.

The following structure diagrams are limited to the three button menus from which the user can select predefined actions and these are shown in Figure 8.1.

![Figure 8.1 - FMIS user interface components](image)

The applications menu currently has no options available. Entries shown on the applications menu are included to indicate the location of external applications on the interface screen. The menu structure for the Display Control menu is shown in Figure 8.2. Figures 8.3a and 8.3b show the menu structure diagram for the FMIS Main Menu. Due to its size, this diagram is split over two pages. The text to the right of the arrow on the diagram(s) show the script name which is executed when this menu item is activated. A full listing of all scripts can be found in Appendix D. The scripts implemented for this project generally consist of small programs which combine Genamap, UNIX and other FMIS scripts. A complete description of these scripts is not provided as part of this project report since most are syntactically similar to the UNIX Bourne Shell and include a liberal amount of commenting.
Figure 8.2 - Display control menu structure

Figure 8.3a - FMIS main menu structure chart
Figure 8.3b - FMIS main menu structure chart
8.2 Lighting Circuit Identification

This application is described because it represents a general class of problem that will be encountered frequently during day-to-day use of the FMIS. This application also effectively demonstrates the advantages of the loosely coupled approach which the overall system design has endeavored to achieve and illustrates the SQL link to the external database.

The English language definition of the lighting circuit problem is: "Find me all other lights which are connected to the same circuit as the light I am about to select with my mouse and, show me where the circuit breaker for these lights is located." The solution can be broken down into three tasks;

1. Select a light pole from the screen and retrieve the lighting circuit data for this light pole from the attribute table.
2. Select all other light poles from the attribute table which have the same lighting circuit attributes and return a list of tags which can be used to highlight these on the screen.
3. Import the list of tags into the FMIS, create a feature set, and highlight this feature set to display all lights that lie on the same circuit.

For our now favorite light pole, LP0171, the traditional database approach would suggest a single nested SELECT statement such as the following:

```sql
SELECT tag FROM ATTRIBUTE
WHERE description_code = '179' AND layer = '180'
AND value = (SELECT value FROM ATTRIBUTE
WHERE tag = 'LP0171' AND layer = '180'
AND description_code = '179')
```
These statements would be piped to the Ingres SQL server which would return the results of the select statement to standard output.

In the implementation of FMIS queries, the system design seeks to break down instances of a specific problem into multiple instances of simpler, more general problems. Such an approach reduces both the size and complexity of the database interface for the FMIS. Using multiple SELECT statements and assignment of intermediate results to temporary variables, our SQL code for LP0171 now becomes two separate SQL statements;

```sql
    circuit_number = SELECT value FROM attribute
        WHERE tag = 'LP0171' AND layer = '180'
        AND description_code = '179'

    SELECT tag FROM attribute
        WHERE description_code = '179' AND layer = '180'
        AND value = $circuit_number
```

The above statements represent elemental forms of common query statements for the FMIS. Each statement can be used in other query functions because a general form is defined which permits the specification of both field names and field values at run-time. The FMIS pilot implementation has achieved this capability through use of an SQL pattern file. These files are listed in Appendix G. Pattern files specify templates for versions of the SQL statement needed by the FMIS to access the attribute table. An FMIS application calls the script callsql with the various field and value specifications as script arguments. This script calls the perl program createSQL.pl which carries out the substitution of the arguments into the pattern fields of the SQL template file. The output of createSQL.pl is piped to the Ingres SQL server and the results of this inquiry piped back through grep, sed and tail to remove extraneous characters.
The user is returned a highlighted feature set of all lights on the same lighting circuit. The room location of the circuit breaker is displayed in the terminal window, and if the relevant floor plan of the building exists, the location of the room is highlighted on the map display. These programs can be found in Appendix D. The reader is urged to review these programs since they are examples of elegant solutions implemented in the FMIS which are made possible through use of UNIX tools and the loosely coupled design approach which the FMIS has endeavored to follow.

As mentioned previously, this problem represents a specific instance of a general class of problems. It is common to request the identification of all features which have similar characteristics (attributes) to a specific feature: e.g., "all lights which are thirty feet in height" or "rooms which have a certain number of token ring connections." The ability to solve this problem means the general class of problem can also be solved using essentially the same program/SQL code. This code re-use would not be possible in a more tightly coupled GIS/database interface and demonstrates the effectiveness of the FMIS design.

8.3 The Attribute Data Reformatter

While the database access interface (developed by the physical plant department) provides a method to retrieve and add attribute data to the attribute table, it is expected that most data entry will be accomplished through use of spreadsheet programs by campus clerical staff. Spreadsheet programs allow a user to enter data in a tabular fashion similar to records in a traditional database table. Since this tabular approach is intuitive to most users, a program has been written in perl
which produces a file which can be imported into the attribute table or generates a coordinate import file for Genamap. This program, flattoimp.pl, accepts a data dictionary file name and name of the data file to reformat as parameters. If the -point option is specified, the program also accesses the file warcl.xyz which contains geographic coordinates for all points surveyed by the campus survey.

The data dictionary file for reformatting data can be found in Appendix H. This file has the form:

    feature_class layer_number field_1 field_2 ... field_n

The feature class field is used to associate a data dictionary definition with the corresponding line from an input file and the remaining fields specify description codes from the description codes table. These descriptions are used to encode description code fields in each output record.

Input file lines have the following form:

    feature_class field_1 field_2 ... field_n

An example input file for storm water manholes can be found in Appendix H. The program flattoimp.pl first reads the data dictionary file specified on the command line and loads the contents of this into an associative array indexed by the feature class field. As each line from the input file is processed, the corresponding entry from the data dictionary array is retrieved using the first field from the input line as the index into the array. If there is no corresponding entry in the array, then the record is skipped and the next record from the input file is read. If an entry is retrieved from the data dictionary array, then an output record is produced for each
attribute data field of the input line, with the appropriate description code and ancillary fields.

The implementation of the attribute data reformatter provides a method for the easy incorporation of data from many different sources into the attribute table. The data dictionary file formats can be easily added or changed as required, and this provides a great deal of versatility to the available methods of data entry for attribute data. It is expected that most attribute data will be input on personal computers using existing spreadsheet and database software. These files will then be batch processed into the FMIS using the data reformatter once data integrity checks have been performed.

8.4 Spatial Queries

The implemented interface for the FMIS provides several options to carry out spatial queries on displayed maps. These include: annotating dimensions of a polygon, measuring distances on screen with a mouse, and retrieving perimeter and area data for polygons including rooms within buildings. These spatial query functions provide a great deal of functionality to the system by permitting users to easily process spatial queries. Without FMIS, these types of queries would previously either be estimated by staff remote from the site or entail a specific site visit and survey. Plate 8.1 shows the FMIS screen after several spatial queries. The area of a section of grass has been retrieved and this is displayed along with its perimeter dimension. Also shown is a distance query which can used to measure the linear footage along the route of a proposed conduit.
8.6 Conclusions and Comments

The FMIS represents an information infrastructure. Like any infrastructure its value to the total production of an organization is difficult to assess and yet it is known to be an essential ingredient for the success of the organization. This project measures the success of the system design primarily through the ease of implementation of the pilot applications reviewed at the beginning of this section. These applications exhibit a simplicity which is only possible from the loosely coupled approach which is fundamental to the FMIS system design. Other goals of this project have been satisfied through the design and implementation of the following components; layers for segregating map data in the FMIS, a simple yet extremely powerful method of establishing user views into the map database, and, a database table structure that provides for a comprehensive historical record of map attribute data and permits allocation of new fields with no table structure changes.

The FMIS pilot implementation has been well received within TAMU-CC and the FM industry. The pilot implementation has been demonstrated to TAMU-CC administration and the physical plant department. As a result of the positive feedback from these groups, a proposal for the full implementation of the FMIS, over a two-year period, is currently under consideration by TAMU-CC administration. The annual conference for the Texas Higher Education Coordinating Board's Facilities Management group is being held in Corpus Christi this year. The conference activities include a field trip to TAMU-CC for a demonstration of the FMIS. In August of this year, the FMIS pilot project was described in a paper delivered at the Sixth Annual Genasys International User's conference. This paper was well received and Genasys II, Inc. has expressed an interest
in acquiring the FMIS pilot implementation as a demonstration application. From the foregoing, it can be seen that the FMIS project has relevance for both TAMU-CC and the FM industry, and supports the contention that the project has some significant value to the community as a whole.

The design of the FMIS at its highest level of abstraction represents a generalized design philosophy which seeks to retain as much independence between components and standardize interactions between them. The high level design which has been produced by this project is general enough that it can be applied to almost any facilities management scenario with most GIS/database software combinations. The system design of the FMIS, therefore, has benefits beyond implementation for the TAMU-CC campus. It is expected that the design which this project has produced will serve, at the very least, as a point of beginning for other FM projects.
Bibliography


Trademarks

Genamap™ is a registered trademark of Genasys II, Inc. AutoLISP™, AutoCAD™, DXF™ are trademarks of Autodesk, Inc. UNIX™ is a trademark of AT&T Laboratories. All other trademarks are the trademarks or registered trademarks of their respective companies.