THE GEOLINK SYSTEM, INTERFACING LARGE SYSTEMS

T.C.Waugh
Department of Geography
University of Edinburgh
Drummond Street
Edinburgh, EH8 9XP

R.G.Healey
Department of Geography
University of Edinburgh

ABSTRACT

There is a great increase in the use of packaged systems of all types. In particular, there is an increasing use of database systems to store vast amounts of data. Many of these systems generate output which is intended as input to another system. In most cases, the output format from one system is incompatible with that of the second system and will require reformatting. The GEOLINK system is a general purpose interface system to take output from one system and prepare it for another system. It is used where fairly standard conversions are required and can be set up by the end user who need have little programming experience.

INTRODUCTION

Over the last few years, the use of large software systems has been on the increase. This is especially true due to the dropping of memory prices for micro-computers and the use of virtual memory on mini and mainframe computers.

These large software systems are designed to provide solutions to a vast array of problems. However, no one system is able to provide all the solutions for an individual, far less an organisation.

Even with the advent of integrated systems such as SAS or MAPPER, primarily for mini and mainframe machines, and EXCEL, FRAMEWORK, and SYMPHONY etc. for micro-computers, no one system fulfills a complete range of functions. In practice, most 'integrated' systems are designed to be good at a specific function and are less than optimal for other purposes. For example, EXCEL, FRAMEWORK, and SYMPHONY are primarily spreadsheet systems with a variety of other functions available, usually in a somewhat constrained way.

Thus a professional will use a variety of tools to achieve solutions. Musciano (1986), while talking about programmers and choice of programming languages states:

> "A fatal flaw among programmers is the desire to use one language for everything. Do you drive nails with a pair of pliers? Cut wood with a butter knife? Write numeric-intensive applications in C, screen editors in

> COBOL, or system code in APL? It is the programmer's responsibility to learn several languages to keep in a 'toolbox'. When a programming problem arises, the programmer can then make an intelligent choice as to which tool will best solve it."
>
> (Musciano,1986)

The same philosophy is also true of systems for geoprocessing. One should be able to choose the system, package, or program that is most suitable for the job in hand. There should be no need to compromise on the quality of software used.

SCALE OF THE PROBLEM

There are, however, many problems with using different systems for different functions. Some of them are:

Compatibility of the data. Inevitably, if different systems are to be used then data must be transferred from one system to the other. While many efforts have been made to produce standard mechanisms for data transfer (e.g. SIF, IFF, and DIF), none has been entirely successful. Currently, there are major national efforts to promote data transfer standards. It should be noted however that these transfer mechanisms are for the transfer of bulk data and may, of course, be limited by incompatible data structures.

These mechanisms do not, however, work very well if ad hoc data, such as command data are being transferred. Thus the problem of output from one system which is 'command' input to another system, as opposed to data on which the 'command' will operate, is not really addressed by data transfer standards.

Compatibility of the user interface. If multiple systems are to be used for multiple functions, the problem of the incompatibility of different user interfaces arises. The user must be able to understand how to take output from one system and input it to another. Getting data output from a command driven structure and input to a menu driven system may be very difficult. For example, many menu driven systems have no mechanisms for inputting data, other than through the menus. This makes it almost impossible to 'import' data from another system.

Selective user understanding of systems. A major problem when using multiple systems, be they hardware or any kind of software, is the requirement that users understand both systems in a data transfer. This is complicated by the common situation, that getting data into and out of a system created by, or for, another system, can quite often be one of the most difficult things to do and cannot be expected of a casual user. This can be alleviated by the users modifying skeleton command sets provided by more experienced users however.

A common response to this problem is to attempt to solve problems by attempting to coerce one system into solving all problems. We all know people who are convinced that their hardware, operating system,

computer language, or application system can solve all problems, given a little effort. Apart from producing an inefficient result, the 'little effort' may turn out, and often does, not to be 'little' at all.

It can therefore be seen that there are various problems associated with using multiple systems to solve a particular problem where data must be transferred from one system to one or more others.

A TYPICAL PROBLEM

At the Department of Geography of the University of Edinburgh, we have several different, but complimentary software systems.
These include : ORACLE - relational database
GIMMS - statistical maps and graphs
SAS - statistics and report generation
ARC/INFO - geoprocessing system
None of these systems offer complete solutions to all of our problems and each has facilities that the others lack.

One of the biggest problems is the database. We use ORACLE for all our database applications. The supplied utilities provide good input and retrieval capabilities, but, apart from a graphics module not yet available, the system basically prints out results or puts them in a file for further processing.

We have many requirements to port selected data from the database to other packages. In particular, we require to move data from the ORACLE database to GIMMS for mapping and graphics. This requires format conversion of ORACLE output to GIMMS input and the generation of a GIMMS run for each ORACLE query. These queries may produce different amounts of output and different types of output.

TYPICAL SOLUTIONS

There are various approaches to the typical problem where one system outputs a file which must be used by another system. Some are:

(a) Force one system to produce input compatible with another system. Many systems add a great deal of ancillary information to any output generated. For example, unless it is switched off, ORACLE (and SAS) will output at the top of each page the names of the columns being output and will output the number of rows retrieved at the end of query.

Much of this ancillary information may be switched off but sometimes not all can be. In addition, this does not address the problem of adding user choices, as do (b) and (c) below.

(b) Edit the output run to produce a run for another system. Since the output file from ORACLE is a text file containing data, it can be edited. Thus the file can be modified and other information added.

For an ad hoc problem, this is probably the most effective method but becomes inefficient when many queries are very similar, resulting in tedious editing, over and over again, of files that are essentially similar.

This method does, however, have the advantage that any extra information can be added easily. For example, having edited the file, and run the result, then changing for example, pen colours is merely a matter of re-editing the file and running it again.

(c) Write a special purpose conversion program.
If the process of doing format conversion is a common occurrence then a format conversion program could be written. It would take output from one source (e.g. ORACLE), massage it, and prepare it for input to another source.

Extra information can be added by the program and, indeed, the user could be queried for any choices or additional information.

The disadvantages of this approach are related to the writing of, and subsequent modifications to, the program. Any time a change is required then the program must be modified and the compile, link, run, and debug cycle must be repeated.

In many cases, this cycle is acceptable and is a reasonable way to proceed. It is, however, inefficient if there are many changes or many different types of conversion.

In the first scenario, the problem is excerbated if it is not the user doing the programming and made even worse if the language used for the program is unfamiliar to the actual user. The programmer spends a great deal of time making minor changes to a program because the user is unable to make the changes themselves.

In the second scenario, there will proliferate many similar programs with their associated object, listing, and executable files.

Thus, while there are several ways to achieve the required result, some work only for certain types of multiple system usage, and others are less than optimal in many cases.

THE GEOLINK APPROACH

Overview

The development of the GEOLINK system began as an attempt to solve several of the above problems and, more importantly, to move certain types of effort to the end user who knew the data and the systems involved, in some cases, better than the programmer. The system has ended up as a general purpose tool in a multiple system environment.

The system, as envisaged, had several functions.

(a) to do any required format conversion.
(b) to use formatted data to create a run of a specified system.
(c) to use skeleton files of common runs of the system as a base resource.

Thus the basic GEOLINK system has two input files and one output file (see Figure 1). The input MASK file is a skeleton run of the target system with embedded GEOLINK commands. The input SPOOL file contains data from the source system. GEOLINK takes these two input files, and, by merging them and obeying the GEOLINK commands in the MASK file, creates a TARGET file which is a complete run of the target system. The TARGET file may then be run producing the desired output.

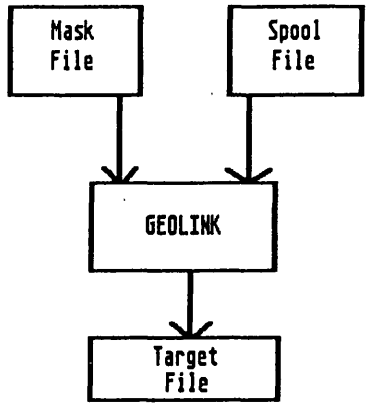

Figure 1 Basic GEOLINK structure

A Simple Example

For example, with a SPOOL file containing

JAN	FEB	MAR	APR	MAY
54	63	27	36	74
29	31	35	43	38
84	76	81	67	69

and a MASK file containing

```
$GIMMS
  *FILEPARM 19,'GRAPH.PLOT',TEXT,OUT
  *GRAPHICS
!SK
!DE JAN FEB MAR APR MAY
!BL
!RE
    *GRAPH !JAN !FEB !MAR !APR !MAY
!EL
    *END
  *STOP
```

then the following TARGET file is produced

```
$GIMMS
  *FILEPARM 19,'GRAPH.PLOT',TEXT,OUT
  *GRAPHICS
    *GRAPH 54 63 27 36 74
    *GRAPH 29 31 35 43 38
    *GRAPH 84 76 81 67 69
    *END
  *STOP
```

which would produce three graphs when run through GIMMS.

In retrospect, the software that is closest to the GEOLINK system is mail merging software such as Micropro's MAILMERGE.

In the above example, the sequences beginning with ! are GEOLINK commands. The ones used in the example above are:

- !SK Skips a line in the SPOOL file. In the example this is the line containing the title information.
- !DE Declares a list of variables which will be input from the SPOOL file.
- !BL Mark the beginning of a loop. This loop will be executed until the SPOOL file is exhausted of data or some specifice exit is taken. !EL marks the end of the loop.
- !RE Reads a set of values from the SPOOL file and puts them into the variables declared in the last !DE command.
- !EL Marks the end of the loop defined by !BL.

All other lines in the MASK file are output to the TARGET file as they are except where variable substitution occurs such as in !JAN where the current contents of the variable JAN is output.

Interacting with the User

A major feature of GEOLINK is the capability to interact with the user to affect the TARGET file. This allows commands to be put into the MASK file which will query the user for information to be output to the TARGET file (see Figure 2).

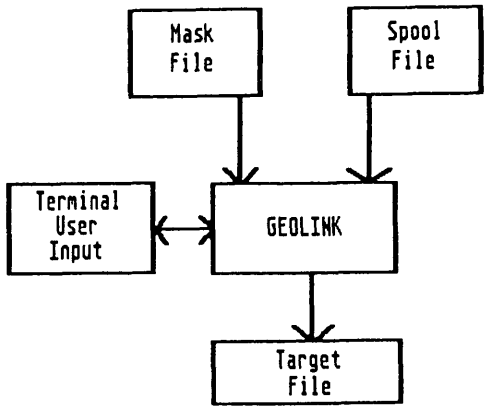

Figure 2 General GEOLINK Structure

For example,

```
!QU "Enter title string:" TITLE
  *TEXT POSITION=1,2,TEXT='!TITLE'
```

would include the *TEXT command in the TARGET file with whatever text had been typed by the user in response to the query on the screen.

With the use of the !PR (print) and !CS (clear screen) commands, complete menu or conversational systems can be set up which need not, in fact, use a SPOOL file. Therefore GEOLINK can be used as an interactive preprocessor for any system allowing simple question and answer sessions to generate a TARGET file. The uses are limitless.

More advanced facilities

GEOLINK has many facilities to allow flexible operation and make the user interface easy to control.

Conditional instructions. These allow choices to be made, usually based on a value read from the SPOOL file or queried from the user. For example,

```
!QU "Enter scale (50,100, or 250):" SCALE
!IF SCALE 50
  *SCALE FACTOR=0.002
!EI
```

where the !IF and !EI mark the beginning and end of a conditional block. Other conditional instructions are !NI (not if) and !IA (if and).

Changing input and output files. The current files can be changed at any time. Thus, one or more TARGET files can be produced from one or more MASK files, using SPOOL files as appropriate. A simple example is to set up a menu and open a new MASK file based on the result.

```
!BL
!PR "1 Select data"
!PR "2 Plot results"
!QU "Enter a choice (1 or 2):" CHOICE
!IF CHOICE 1
!NE MASK "DO:[TCW]MENU1.MSK"
!EI
!IF CHOICE 2
!NE MASK "DO:[TCW]MENU2.MSK"
!EI
!PR "**** Illegal Choice ****"
!EL
```

would loop asking for a valid choice when a new MASK file would be opened.

Computation. Various arithmetic and character operations can be done on the variables. For example,

```
!CP X_POSITION     X + 0.1
```

would calculate the value for X_POSITION.

System Functions. Several facilities are available which depend on the underlying operating system. Currently, on a DEC VAX, GEOLINK can:

(a) Spawn subprocesses based on a user response
(b) Call user defined programs
(c) Use the system date and time for many purposes

It is intended to enhance the system by providing better access to system facilities.

A more Complex Example

The initial driving force behind GEOLINK was to improve the flow of data from one system to another and this example is a general purpose example using ORACLE and GIMMS. The example accesses census data on a grid square basis and produces a map of the result (see Figure 3). The user input is underlined.

```
UFI> START GRIDPLOT
UFI> SET ECHO OFF

Select table for query CENSUS.GRID71 : Scottish 1971 1km census
                       CENSUS.GRID81 : Scottish 1981 1km census
                       CENSUS.AGGRID : English 1km agrid census (part)

Select table(CENSUS.GRID71,CENSUS.GRID81,CENSUS.AGGRID)?:CENSUS.GRID81

Enter criteria for selection?:PPOPN>7000

----- Query being executed at 13:46:56

Enter title of map(QUIT to exit)?:Population > 7000 per Km (81 census)

Scaling factors are in thousands (e.g. 50 is 1:50000)
SCOTLAND for all of Scotland (1:1,750,000)
```

```
UK          for all of UK          (1:3,500,000)

Enter scaling factor(50,100,250,500,1750,3500,SCOTLAND,UK)?:500
Enter X coordinate of centre?:275000
Enter Y coordinate of centre?:670000

Select background outlines(DISTRICTS,REGIONS,UK)?:DISTRICTS

Choose output device (TEK,RAMTEK,PLOTTER)?:PLOTTER

----- Mapping result at 13:48:25

FORTRAN STOP

Ufi>
```

The above example, in fact, uses GEOLINK twice, once to create an SQL query for the ORACLE database, and secondly to create a GIMMS run using the result of the query (which has been SPOOLed to a file) and user input (such as the title, scale etc.). The lines beginning with '-----' are printed out just before exiting from that run of GEOLINK. Figure 3 shows the result.

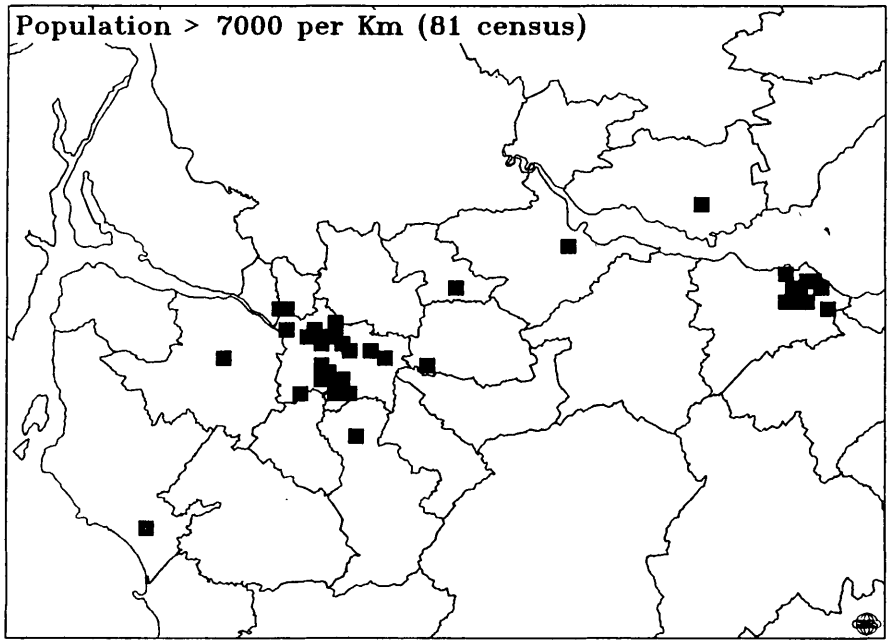

Figure 3

APPLICATIONS

The uses of GEOLINK are legion. As a tool for building applications quickly it is extremely useful, simple, and fast. The MASK files are easy to maintain and can be modified by a user using a simple text editor. Some of the current uses are:

(a) Passing output from ORACLE to GIMMS for general database access and map library work (Healey et al,1984).
(b) Creating a SAS run from VAX accounting output to provide user statistics.
(c) Providing a menu front end for an air photograph library system.

THE SYSTEM

GEOLINK was originally written in the ADA language but has subsequently been rewritten in Pascal. It is implemented on a DEC VAX 11/750.

CONCLUSION

The GEOLINK system has proved to be a very useful tool to link together disparate software systems. It is quick and as user friendly as the user wishes.

Perhaps one of its major strengths is the giving of control of the flow of data back to the real user who is not then dependent on computing professionals.

SYSTEMS MENTIONED

ARC/INFO	- ESRI
EXCEL	- Microsoft Corp.
FRAMEWORK	- Ashton-Tate
GIMMS	- GIMMS Ltd.
MAILMERGE	- Micropro Inc.
MAPPER	- Sperry Computers
ORACLE	- Oracle Corporation
SAS	- SAS Institute
SYMPHONY	- Lotus Corp.

REFERENCES

Healey,R.G.,Morris,B.A., and Waugh,T.C., 1984
'From map catalogue to database with graphics in view'
LIBER Bulletin October 1984.

Musciano,C. 1986, 'Choose your language'
Letter in BYTE magazine, May 1986, p14-16