INTEGRATION OF GIS WITH USER MODELS

by

Pinde Fu
Department of Geography
GEMLab/Kansas Applied Remote Sensing Program
The University of Kansas.

Paul Rich
Department of Ecology & Evolutionary Biology
Kansas Biological Survey & Environmental Studies Program
The University of Kansas

Jue Wang
Department of Geography
GEMLab/Kansas Applied Remote Sensing Program
The University of Kansas.

Citation:

ABSTRACT

In many areas of scientific and engineering research, there is a fundamental need to integrate GIS software with users' own models and programs. Integration is difficult because of differences in data structure, function, methods for data I/O, user interface, and procedure communication. This leads to inconveniences and inefficiency. Herein, we demonstrate our approach to these problems using our design of the Solar Analyst and TopoView, tools for calculation of incoming solar radiation model (insolation) over landscape scales. The calculation engine of these models was implemented in DLL format using C++. The DLLs were then loaded into ArcView to create the Solar Analyst extension. ArcView GRIDIO library was used to read a DEM as input and write insolation GRIDs as output. This paper explains such techniques as ArcView GRIDIO usage, procedure communication (system call/DDE/DLL/ActiveX), design of DLLs, and on-line help systems for use with ArcView.

1. INTRODUCTION

A major obstacle to GIS applications is the difficulty to integrate GIS with users' own models (Goodchild 1993). This difficulty is caused by incompatibility in data structures, data input and output, user interface, and capabilities of procedure communication. The use of GIS with other models is generally time consuming and costly. Both GIS and user models are underutilized due to the absence of a bridge between them. There is a crucial need to integrate them at the operational level.

This paper discusses issues concerning the integration of GIS with user models, including extending point-specific models to landscape and regional scales, as well as techniques for integration of model components.
2. METHODS

2.1 Extending point-specific models

Many user models, in particular non-GIS based models, are point-specific. Most physical and ecological models are developed for use in specific locations at a limited number of study sites. These models are often not easily generalized to other locations and broader spatial scales. Integration of these models in a GIS environment and extension to larger areas is essential for a more complete understanding of spatial patterns of key physical and ecological processes, as well as consequences of these processes (Forman and Godron 1986, Goodchild et al. 1996).

For example, incoming solar radiation (insolation) intercepted on the ground surface is influenced by topographic effects, including elevation, surface orientation, and topographic obstruction. For specific locations, the topographic effects are typically accounted for by visual estimation, or by the aid of upward-looking hemispherical photographs (Rich 1989, 1990, Rich et al. 1999). How can such point–specific models be integrated with GIS to calculate insolation for every location over an area? Two new insolation modeling tools, the Solar Analyst and TopoView, were designed for iterative calculation to produce insolation maps at landscape and regional scales.

Often it is difficult to acquire site–specific information needed to run point-based models in different locations. The key question is how to parameterize such models.

For example, to calculate insolation for any location using point-specific models, the following parameters are needed: elevation, slope, aspect, upward-looking hemispherical viewshed, atmospheric transmittivity, and diffuse proportion. The area-based model, the Solar Analyst or TopoView, requires these parameters for each location in a landscape (Fu and Rich 1999, Fu and Rich 2000, Rich and Fu 2000).

There are two approaches to address the issue of providing appropriate parameters for different landscape positions:

1. **Allow the user to specify appropriate parameters when running the model.** Such parameters must be easy for the user to measure, or some typical values should be readily calculated or available for choice. This approach is relatively simple, and can be the best choice for certain circumstances.

For example, in the design of the Solar Analyst and TopoView, atmospheric parameters (transmittivity and diffuse proportion) are parameters the user must specify. The user can choose different values that apply to different atmospheric conditions conditions (clear sky, overcast…).

2. **Calculate the values based on interpolation, extrapolation, or derivation from readily available data sources.** The values of each location can be represented as polygons (for vector data) or surfaces (for raster or TIN data). Raster data is best for representing high heterogeneity and for enabling fast calculation.

There are various ways to create parameter surfaces. The following are examples:

- **Field measurement:** Data layers for some parameters can be measured for each location in a study area, or can be measured for some polygon units. These data can then be converted into raster format.

  *For example, the Solar Analyst and TopoView can optionally use slope and aspect surfaces created based on actual field measurements (which are more accurate than those calculated from a DEM), and can use different calculation assumptions (e.g. surface orientation corresponding to sensors or leaf orientation).*

- **Spatial interpolation:** Surfaces of many parameters, such as temperature and precipitation at regional scales, can be generated using interpolation, such as nearest-neighborhood, trend surface, inverse distance weighted interpolation (IDW), spline, and Kriging. Interpolation techniques should be chosen based on the features of interest, characteristics of spatial distribution, number of available samples, and the required spatial resolution (Burrough and McDonnell 1998).
- Existing GIS functions: Many existing functions can use existing data to generate surfaces of data values needed in models. Slope, aspect, curvature, path distance, watershed, flow accumulation, and flow length can all be derived from digital elevation models (DEMs).

For example, in the Solar Analyst and TopoView, slope and aspect can also be calculated from the DEM (Hodgson 1995).

- Programming of New Functions: Some parameters cannot be obtained using existing GIS functions. These parameters must be calculated using customized programs.

For example, point-specific models such as Canopy (Rich 1989) and HemiView (Delta-T 1999) account for the effects of sky obstruction by surrounding ground features using upward-looking hemispherical viewsheds obtained from photographs. Such approach are not practical for areas, and the calculation of upward-looking hemispherical viewsheds is not available in existing GIS software. The Solar Analyst and TopoView calculate upward-looking hemispherical viewshed for each cell of a DEM using customized C++ programs.

2.2 Component Integration and Communication

Integration of GIS with user models takes many forms, including file conversion, system calls, dynamic data exchange (DDE) / Remote Procedure Calls (RPCs), dynamic link libraries (DLLs), and ActiveX Control. These approaches offer different advantages and disadvantages, and vary in sophistication and effort required.

- Common Files: The easiest and most commonly used approach to integrate two sets of programs involves using common format of data files. For example, GIS can be used to generate or preprocess a number of different data layers; next these data are feed to user models; and then the results of user models are processed using GIS again for visualization and output. In this approach, GIS and user models are separate components and there is no direct interaction between them. This approach requires almost no software modification or programming, but is often cumbersome. Usually data conversion is needed. If a model does not use standard formats, data conversion can be complicated, and even prohibitive. It may involve a large amount of manual work, which is both time-consuming and error prone. In general this types of integration is simple to implement, but limited and sometimes ineffective.

- System Calls: System calls activate one program in another program. For example, a model can be called within ArcView using System.Execute(aCommandString) or System.ExecuteSynchronous(aCommandString). Parameters can be passed to a user model as CommandString, but only at a limited length. The results of the user model can not be passed back directly. In this approach, GIS and user models can appear to be tightly linked, but in reality this link is still weak. Once a user model is started, it can not interact with the GIS program that started it.

- DDE/RPC: This approach actually sets up a client/server relationship between user models and a GIS. The application initiating the request or conversation is the client, and the application that fulfills the request is the server. This integration provides capabilities for data exchange at a function-level. It allows communication of programs that reside on different computers within a network. Its disadvantage is that sometimes it is not stable.

- DLL: DLLs are executable modules that contain functions for use by other programs. For example, DLLs of the ArcView GRIDIO library allow users to develop GIS capabilities within their models. Similarly, users can build their models in DLL formats for use within GIS software. Information can be passed back and forth as parameters. Application integration through DLLs is tighter and faster than DDE/RPC because it occurs at a lower-level than DDE/RPC communications. This method desirable for many models, but involves considerable programming effort.

- ActiveX Control: ActiveX controls (previously called OCX) are components that can be plugged into many different types of applications called ActiveX control containers. They are different from DLLs in that they
normally have a visual component. ESRI MapObjects is an example of ActiveX. The ActiveX control approach allows users to add GIS functions within their own applications. So far, GIS software does not accept user models in ActiveX control format.

2.3 Examples of DLL usage with ArcView

2.3.1 Usage of ArcView GRIDIO Library

ArcView GRIDIO is a grid data set input/output library that allows users to read and write ESRI grid data sets directly. It includes a run-time library and C API. The usage of GridIO is not documented in the ArcView manual.

- **Installation:** GRIDIO DLLs come with ArcView while GRIDIO C API comes with the Spatial Analyst. Therefore, model developers using GRIDIO should have the Spatial Analyst installed while users of these models just need ArcView installed.

GRIDIO DLLs are always installed. By default they are located at c:\esri\av_gis30\arcview\bin32. To install the C API, select custom installation while setting up the Spatial Analyst, then check the C API for grid data set I/O. This installation installs gioapi.h, avgridio.lib, several examples of C programs at c:\esri\av_gis30\arcview\gridio.

- **Usage of the C API with C++:**

  **Include gioapi.h:** C++ uses the same calling convention and parameter-passing techniques as C, but its naming conventions are different because of C++ declaration of external symbols. The extern "C" syntax is needed in order to use GRIDIO with C++. The following is an example:

  ```
  #ifndef _ESRIGRIDIO_
  extern "C" {
  #include "c:\esri\av_gis30\arcview\gridio\gioapi.h"
  }
  #define _ESRIGRIDIO_
  #endif
  ```

  **Link avgridio.lib:** avgridio.lib should be specified as link library.

  **Specify Path to AVGRIDIO DLLs:** The directory to the GRIDIO DLLs (e.g. c:\esri\av_gis30\arcview\bin32) should be included in the PATH environment variable.

Other important details involve memory allocation and use of large grids. Before reading grids into computer RAM, memory needs to be allocated. The GRIDIO library requires that the underlying data array for a two-dimensional buffer is 1–Dimensional contiguous. GetWindowRow, GetWindowBand, and GetWindowRectangle can not read gridbbs that have more than 1024 columns. In such situations, GetWindowCell should be used.

2.3.2 Design of DLLs for use with ArcView

The topic of designing DLLs for use with ArcView is not well documented in the ArcView Manual. The following is our approach:

- **Exporting functions**: ArcView uses DLL functions that are exported. Previously, module-definition (.DEF) files are used to export DLL functions. In current C++ programming, typically dllexport rather than the .DEF file is used to export functions. But we found Avenue can not call functions exported using dllexport. Therefore, .DEF should be used. The examples concerning DLLs given in the ArcView on-line help do not work for us unless a def file is added to export DLL functions. For example, the following example is given in ArcView help:

```c
// C Source Code for math.dll
#include <windows.h>

long AddLongs(long a, long b)
{
    return a + b;
}

'Avenue Script to Call AddLongs
myDLL = DLL.Make("c:\lib\math.dll".AsFileName)
add = DLLProc.Make(myDLL, "AddLongs", #DLLPROC_TYPE_INT32,
    (#DLLPROC_TYPE_INT32, #DLLPROC_TYPE_INT32))
x = add.Call({500, 200})
MsgBox.Info("The result is " + x.asString, ")
```

In our experience, the following `math.def` should be added and linked at compilation time in order to make the example work.

```
; math.def
LIBRARY math.dll
CODE PRELOAD MOVEABLE DISCARDABLE
DATA PRELOAD SINGLE
EXPORTS
AddLongs
```

- **Calling C++ class member functions**: For programs in C++ style, most functions are member functions of various classes. Unfortunately, member functions cannot be directly called by Avenue. For example, it is not possible to call member functions such as solar.SetInput, or Dog.Bark directly. The workaround for this problem is to create some non-member functions, which call the member functions. Then Avenue can be used to call the non-member functions, which in turn call the member functions.
3. CONCLUSION

GIS does not provide all functions needed by all users. Most GIS users face the challenge of integrating their own models with GIS software. Some user models can be implemented using the programming language of the GIS software, while some cannot. Many user models are point-specific and not easily ported to a GIS environment. Extending these models to large areas in a GIS environment can be accomplished by obtaining data layers of parameters for the areas. System integration can be accomplished by using common files, system calls, DDE/RPC, DLLs, or Active X controls, according to specific user needs. There is growing interest in extending point-specific processes to broader spatial scales, using spatially explicit raster models. ArcView GRIDIO provides an easy way for non-ESRI products to directly access ESRI grid data. This paper addresses ways to export C++ functions and workarounds to call C++ member functions when developing DLLs for use with ArcView. As GIS is used in more and more ways to solve more complex problems, the topic of integrating GIS with user models becomes increasingly important.

4. REFERENCES


