Linking GIS and hydrological models: where we have been, where we are going?

STEPHEN M. KOPP
Environmental Systems Research Institute (ESRI), 380 New York Street, Redlands, California 92374, USA

Abstract Hydrology and GIS are two technologies that, in different ways have tried to solve some of the same kinds of problems. As the synergy of these two technologies was realized, many successful attempts were made at integrating them. As the computer industry has evolved, so has the ability to create integrated applications of GIS and hydrology. Emerging standards for data and communication, and a shift in development philosophy, are providing developers the means to do what they have wanted, and the desire to do even more.

WHERE WE HAVE BEEN

Ten years ago, GIS was a tool for managing and analysing spatial data for everything from urban planning to forest stand management. At this time, hydrologists were collecting their own data and storing it in a format specific to the model they worked with. The GIS was not analytical enough for their needs, and the model was not effective at data management or display. The integration of the technologies began slowly as GIS was used to perform overlays and aggregate information such as basin characteristics to pass to an external FORTRAN program or statistical package. These links evolved to more complex and robust implementations, where simple models were embedded in the GIS, or input/output (I/O) routines of the GIS were embedded into complex models. During this time, the software industry graduated to multiple window systems and menus, and started to develop integrated applications. By the early 1990s, developers of GIS software began to include functionality of particular interest to hydrologists. As a result, connections between GIS and hydrological models have grown steadily stronger.

The integration of GIS and modelling has evolved being GIS-centric. This is mainly the result of the GIS being able to perform so many tasks for the modeller; data manager, pre and post-processor, display. With the emergence of UNIX and X, GIS software became not only a geodata server and geoprocessing tool box, but also a graphical user interface (GUI) builder and application development environment. If we needed some other tools for statistics or 3D visualization, we either linked other software packages to the GIS or wrote a data conversion program to move the data between packages. GIS provided a common ground on which people and their data could interact (Lakhtakia et al., 1993).

Current developments in the software industry will revolutionize the way integrated hydrology-GIS applications are developed and the way users interact with them. These are not major advances in hydrology or GIS, but the evolution of industry standards and development tools that enable software from these and many other technologies to interact more easily.
Methods of integration

There are three approaches to GIS-model integration and some issues to consider when determining which is the most appropriate method for a particular application. While the methods are constantly evolving, the general approaches and issues one would consider remain the same.

**GIS based modelling** This approach involves programming the model within the available tools of the GIS. Modelling within the GIS is effective for models such as the Universal Soil Loss Equation, DRASTIC or TR55. These are mathematically and conceptually simple models, requiring the least amount of developer expertise. This approach also works well for home grown models and for components of larger data systems (DePinto *et al.*, 1993). It is the easiest approach because the only software one needs to know is the GIS. As GIS software has evolved to include more hydrology-specific tools this level of integration has increased in popularity.

**Data bridge** By far the most common approach to linking models and GIS has been through data conversion. People have been quite resourceful and creative in writing custom programs to pass spatial data from the GIS to a model, then convert the results back to display and further analyse in the GIS, using models such as AGNPS, WASP4, HEC2, SHE and many others.

**Embedded code** The tightest method of integration, and the one requiring the greatest programming resources, is to embed the code of one program into another. Commonly this involves the embedding of the input/output routines of the GIS into the model, allowing the model to read and write GIS data in its native format. The lack of intermediate conversion steps creates an application with a speed that allows development of interactive applications not previously possible, with models such as MODFLOW and SWMM.

Some developers have used the GUI tool kit of the GIS to develop turnkey applications that launch the model from within the GIS, so that it is hidden from the user who simply interacts through a menu. While great for the end user, this type of interface can be a nightmare for the developer to create and maintain.

**Linkage issues** Even though the methods of integration change, the issues remain the same. How complex are the calculations? How much user interaction is necessary? Are you using models that may be regularly revised, or end up in litigation?

Complex calculations such as differential equations or series approximations, should be done with the model. If the mathematics is within the realm of the GIS, consider doing the modelling within the GIS. Some models can be easily canned into an intuitive GUI, but for a model with a complex interface, or one that is readily accepted like AGNPS, it is probably best to help the model understand data from the GIS. Some models receive regular or significant updates, which can be hard to make if the model is embedded inside a large application. Some models may be required by law, or have legal implications, that could be nullified by an unauthorized developer making changes to the model. Additional discussions of methods and considerations for linking models and GIS can be found in Maidment (1993) and Fedra (1993).
Increasing hydrological functionality

As application developers and consultants continue to create more sophisticated and robust end-user applications, the developers of GIS software continue to include more core functionality specific to hydrology. The first step for many was to incorporate tools for deriving landscape or watershed characteristics. These included tools for watershed and stream network delineation from digital elevation models (DEMs), calculating direction and length of overland flow paths and surface curvature. These tools made it much easier for hydrologists to derive basin statistics for input to surface hydrological models. Research and development of tools for watershed hydrology are continuing to improve surface interpolation, land form descriptions and flow routing algorithms (Moore, 1993).

GIS has also incorporated tools to solve two dimensional groundwater flow problems. These tools allow the generation of a Darcian flow field, particle tracking and Gaussian dispersion (Tauxe, 1994). They show promise in the quick assessment of local, large scale groundwater problems and in defining capture zones for well head protection. There is much room for evolution of tools such as this within the GIS framework.

WHERE ARE WE GOING?

Even with all these developments over the last few years, creating dynamic links between a GIS database and hydrological models has still been a challenging process, requiring more time and programming effort than many were willing or able to commit. The dream of real-time decision support systems integrating GIS, hydrological models, statistics, expert systems, or visualization was still for some only a dream.

Recent developments in the software industry will simplify the integration process and move the technology in new directions. These developments are also changing the way we conceptualize GIS. Emerging trends in data access, inter-application communication, component based software and development environments are changing GIS from being the core of a spatial decision support system, or the glue that holds it all together, into GIS being just one component of an integrated custom application. Development of standards is also making such integration easier and providing developers the ability to make applications more interactive (Fig. 1).

GIS Industry Common API (OGIS?, SQL/MM?)
GIS Vendor API
Data Transfer Standards (SDTS, VPF)
Vendor Specific Data Formats

Fig. 1 Increasing levels of GIS data access.

Data standards

Just as there are de facto standards for tabular data (SQL, dBASE), raster (TIFF, GIF) and CAD (DXF), there are now also standards for topological vector data. The Spatial
Data Transfer Standard (SDTS) is a standard developed and adopted by US federal agencies and is being adopted by other countries. SDTS is an interchange standard, intended to be translated into a native application-specific format before it can be accessed. It is not a format to use when moving data around within an application, but rather defines an interchange format for converting data acquired from outside an application or organization. SDTS is emerging as an accepted interchange format for Internet based geodata servers.

The Vector Product Format (VPF) is a US military standard developed for delivery of digital map products. It is a direct use format, in that applications are written that work directly on VPF data without the need for conversion to a proprietary or application-specific format. Because formats such as SDTS and VPF are standards-based, users have some assurance that investment in them will last independent of vendor support.

In many situations, where a GIS needs to interact with a single model or application, simple file conversion will continue to be the most common approach. It is still the easiest and most straightforward approach. These standards should make the simplest form of GIS-model integration, the data bridge, much easier. In the future, access to vendor supported application programming interfaces (APIs) and development of interoperability specifications will continue to improve the speed and ease with which GIS software can be integrated with other applications.

Application programming interface

Developers in the past got their models to read and write GIS data by licensing object code from GIS vendors or working with public domain software. Though the resulting interface was fast, even with this level of access, the programming task was difficult. Vector GIS data has more internal complexity and interrelationships than simpler data formats such as tables or images. Simply knowing the data format without knowing all the rules to ensure data integrity can be dangerous. Updating your program each time the GIS data model evolves can prove expensive.

To provide access to data formats, major relational database management systems (RDBMS), and now GIS vendors, are working toward providing an application programming interface (API) for developers. The API provides an open programming interface for the developer to interact with data in its native format without having to worry about data model evolution and upward compatibility. This level of access has been requested by the modelling community (Furst et al., 1993; Nachtnebel et al., 1993), and will soon be a reality. In the short term, this is likely to appear as a documented set of C or C++ function calls for reading and writing a variety of geodata formats.

Currently no standard mechanism has been defined for geodata access in the way SQL is the language for relational database access. There are two major efforts to define this access, SQL/MM and OGIS (Open Geodata Interoperability Specification). SQL/MM is exploring the addition of multi media extensions to SQL. The Open GIS Consortium is working to define a common data model and application interface in a distributed computing environment. This would allow an application to access data and request geoprocessing services from any system adhering to the specification.
So the role of GIS in modelling has come full circle and is back to being a geodata server. Data transfer standards and efforts toward a common geodata API will provide tighter integration with less programming effort.

**Inter-operability standards**

Inter-application communication (IAC) describes how one application can talk to another (or many others) in a client-server environment. The idea has been around for several years, first appearing as the Macintosh clipboard. In the last few years it matured into Apple Events, and Microsoft’s DDE, which define a set of protocols that allow an application to go out and ask another application for data instead of simply grabbing a piece of data sitting on the clipboard. For example when updating an automated hourly report on river stages, your word processor application can update a graphic of hydrographs by accessing the latest rainfall prediction and running that through your runoff model. DDE was once a proprietary integration solution for Microsoft Office products, but is now accessible to all Windows software developers.

Industry standards are emerging for communication between applications written by different authors running on multiple machines or on a heterogeneous network. These hold great promise for progress in the way developers conceptualize problem solving and the way end users interact with applications.

**Component based software** The demand for increasingly sophisticated end-user applications has led to the development of large feature-laden applications that are difficult to maintain, modify or revise. These applications become static snapshots of technology that cannot grow with the pace of the industry, so the trend is now toward modular or component based programming. Such an approach for linking GIS and hydrological models was proposed by Fürt et al. (1993). Component based software solutions allow developers to take a modular approach to development and maintenance. This modularity allows for easier reuse and upgrading of components or replacement of parts without releasing an entire new application.

Component based software standards such as Microsoft’s OLE (Object Linking and Embedding) and Apple’s OpenDoc, built on top of Object Oriented Programming standards such as COM and SOM, are leading the way toward component based applications development. They define protocols for allowing separately developed pieces of software to interact with one another. This allows application developers to integrate the best parts of software packages, such as the overlay and data management tools of a GIS with the rendering of a visualization package and the analytical power of a hydrological model. These standards define the methods or actions available for data objects, the context of their execution and when they execute, and allow the user to interact with the maps, charts or model parameters all from within a single document. Developers will be able to create custom software solutions by assembling components of off-the-shelf software such as a spatial data manager or statistics package, with custom in-house models.

In their current state, these standards alone do little to improve the productivity or sanity of an application developer when performing a task he has been able to do in the past. But they do provide the developer methods to present integrated information to the
end user in a way that was previously not possible. The user’s headaches and hassles of moving data and jumping from one application to another will be replaced by a document-centred approach, easier for users to learn and easier for developers to maintain.

**The document interface**

A document-centred user model can be thought of as a tightly integrated set of end user applications presented through a single graphical interface or "document". This document contains all the tools necessary to interact with each component and presents them to the user as needed by changing the menu or controls available as different parts of the document are selected.

This allows the user to focus on the task instead of the technical details of how different programs interact. The component based software standards mentioned above hide these details of document manipulation from the user, who has the impression of working with a single software product.

**Development environments** In the past, the GUI tools of the GIS were the primary means of user interface development for integrated GIS-model applications. Now, software development environments exist that are independent of application software. In fact, their purpose is to build application interfaces or documents. Software development environments such as Visual Basic, Delphi and Apple Script provide a framework for creating graphical document-centred applications. Starting from a blank form menu, these programs allow the user to add controls or widgets to the menu and attach actions to those controls. These development environments enable a developer to build integrated, graphical, end user applications or documents without having to climb the learning curve associated with OS-level APIs.

**BRINGING IT ALL TOGETHER**

These software trends and the increasing interest in standardized data structures and data transfer formats will make the development of real-time decision support systems faster and easier than ever before. Figure 2 shows how all these pieces fit together. A docu-
ment interface is created in Visual Basic. OLE/COM defines the compound document architecture that allows each component to communicate. All the components share data and the OGIS specification helps each component interpret the data format.

An ideal use of these new capabilities would be in real-time flood forecasting. Such systems have been developed in the past (Barrett, 1993), but it is only now that we can envision a truly interactive, real-time system. The technical expertise of someone who could develop such an integrated application is about the same as those who have been developing integrated GIS modelling applications, but the realm of possible applications is now much larger, and these applications can be brought to a new level of user.

REFERENCES


