

The submitted manuscript has been created by the University of Chicago as Operator of Argonne National Laboratory (“Argonne”) under Contract No. W-31-109-ENG-38 with the U.S. Department of Energy. The U.S. Government retains for itself, and others acting on its behalf, a paid-up, nonexclusive, irrevocable worldwide license in said article to reproduce, prepare derivative works, distribute copies to the public, and perform publicly and display publicly, by or on behalf of the Government.

## **OPENLINK: A FLEXIBLE INTEGRATION SYSTEM FOR ENVIRONMENTAL RISK ANALYSIS AND MANAGEMENT\***

by D.J. LePoire, J. Arnish, E. Gnanapragasam, T. Klett,  
R. Johnson, S.Y. Chen, B.M. Biwer, and C. Yu  
Argonne National Laboratory  
9700 S. Cass Ave, Argonne, IL 60439

### **ABSTRACT**

Most existing computer codes for modeling environmental pathways were developed to satisfy a specific objective (e.g., perform analyses to demonstrate regulatory compliance). Over time, the codes have been written in various computer languages and software environments that are often not compatible with each other. In recent years, largely driven by advances in industrial software, a new concept for software development based on modularization has emerged. This approach entails the development of common modules or components that can be shared by and used in different applications that have certain common needs. Although modularization promises advantages over the traditional approach, a number of issues must be fully addressed and resolved before the approach can be accepted as a new paradigm for environmental modeling. This paper discusses these issues, provides demonstrations of open integration techniques, and provides recommendations and a course of action for future development.

### **BACKGROUND AND INTRODUCTION**

Most existing computer codes for environmental pathway modeling were developed to satisfy a specific objective (e.g., perform analyses to demonstrate regulatory compliance). Over time, the codes have been written in various computer languages and software environments that are often not compatible with each other. In recent years, largely driven by advances in industrial software development, a new concept for software development based on modularization has emerged. This approach entails the development of common modules or components that can be shared by and used in different applications that have certain common needs. For instance, an air dispersion model could be written into a common component to be shared by several different applications, each with the need to model air dispersion of some release. When fully developed, the modeling application would become an exercise of selecting, integrating, and applying a consistent combination of appropriate modules for a specific problem. Although modularization promises advantages over the traditional approach, a number of issues do exist. These issues must be fully addressed and resolved before the approach can be accepted as a new paradigm for environmental modeling. This report discusses these issues and provides recommendations and a course of action for future development.

Traditionally, model connections have been made by the end user, who would align one model’s output data with another model’s input. Often the model assumptions and conceptualizations were stretched to accomplish the linkage, resulting in greater uncertainty in the results. Also, the connection usually required the user to invest effort in manipulating the data for proper communication (e.g., taking data from the first model’s output and manually editing the input for the next level), resulting in inefficient use of resources and introducing another potential source of error. It is generally difficult to connect models because of their disparate assumptions about scale, conceptualization, aggregation, process, reality, and objectives.

### **OPPORTUNITIES AND GOALS**

\*Work supported by the U.S. Department of Energy under Contract No. W-31-109-ENG-38

In the environmental field, modeling plays a critical role in connecting current data and knowledge with predictions of future events and environmental states. Environmental problems are quite challenging to solve because of the complex relationships among many contributing factors, both natural and man-made (1). Moreover, these problems need to be addressed not only by environmental engineers and regulators but also by concerned members of the public and nongovernmental organizations. Their demands on environmental modeling often conflict because predictions need to be accurate yet easily understood, communicated, and explored. The increasing complexity of environmental codes also places a demand on the end user, who must translate the real environmental problem into the conceptualization allowed by the model and its options. Information on assumptions and options must be conveyed to the user to ensure that the model is applied and interpreted correctly. Open communications about the model, interface, and data components would enable software applications to be more easily developed.

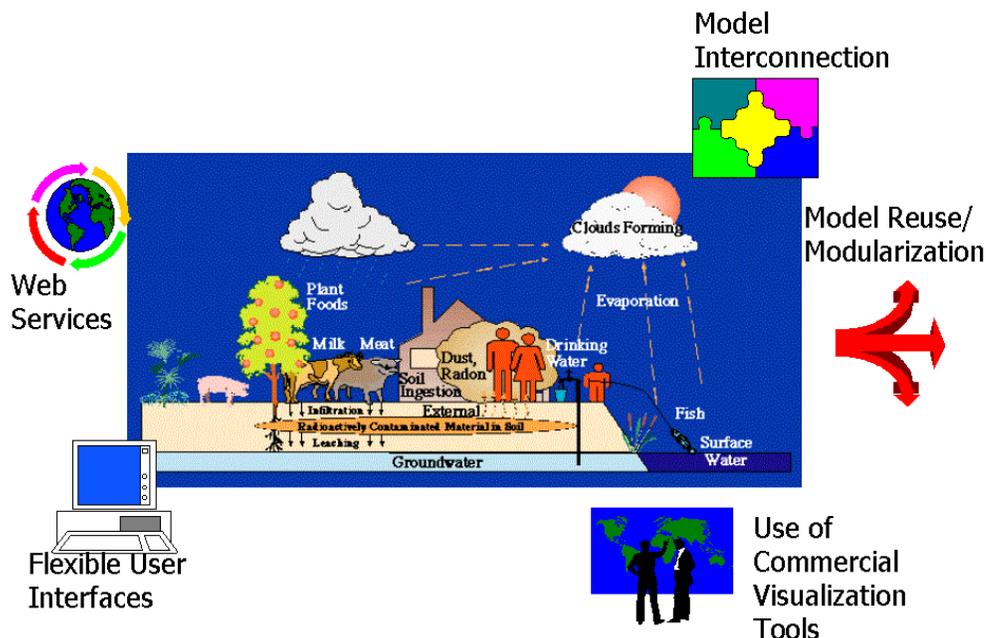
Many modeling approaches could be used to realize these goals. For some purposes, the most detailed models and data are appropriate to predict a situation. However, the results from these models might not facilitate a good understanding of the situation, or they might place an undue burden on the stakeholders as they try to learn how the models work and they try to gather sufficiently verified data.

Another approach is to use simplified models with conservative values for data to facilitate an understanding of the results and to place bounds on them. This approach enables the end user or interpreter to focus on the important issues. Many regulatory processes include this type of analysis to help users explore the issues and decide whether a more detailed analysis is justified.

There is a wide gap between these two approaches and between their contributions to understanding and the decision-making process. Both can be enhanced with tools that allow sensitivity analysis, uncertainty analysis, and visualization and manipulation of the data.

When these considerations are taken into account, it might be reasonable to expect that a range of models and also a range of integration strategies are available. On the detailed side of modeling, the emphasis might be placed on manipulating large, standardized data sets covering spatial and temporal dimensions. The emphasis on the user interface and regulatory requirements could be less, since the end user would be assumed to be more cognizant of the assumptions and therefore more responsible for constructing valid model input. Other times it might be important for the user to just "scope out" a situation and determine the major exposure and risk pathways. This situation might be enhanced by a very flexible system that would allow the end user to easily explore the simplified models. Another emphasis might be on the need to develop some components that could be used in detailed and mid-level modeling environments that require intermediate modeling sophistication and in guiding the user through the regulatory process. These different modeling needs might require some shared models but different integration environments. This requirement could be met by developing various user interface components and standards, process flow standards, data management and sharing tools, decision support tools, and model linkages so that these models could become more flexible, efficient, and effective.

Not only are the demands for environmental modeling changing, but also the means to accomplish this modeling are being developed and quickly changing. Many of these means to accomplish modularization and integration are based on the supply of new information technologies (ITs) originally developed for businesses. Information technologies, such as those for object-oriented code development, network-based distributed processes, database storage and manipulation, graphical user interfaces (GUIs), work-flow processing and communication, geographic information systems (GISs), and graphical visualization, have become available for integration into environmental modeling in recent years. These tools and techniques, which required large investments by commercial vendors, are available relatively inexpensively because of the demand from the business community. Software packages for environmental modeling should take advantage of these tools and techniques in responding to the field's unique needs and demands. The emphasis should not be on redeveloping industry standards but on meeting environmental modeling's unique needs by leveraging these tools and techniques. A diagram of some of these goals is shown in Figure 1.



**FIGURE 1**  
Goals for

### the Open Environmental Modeling System, OpenLink

## TECHNOLOGY ASSESSMENT

### Options

Various options are available for implementing a more flexible environmental modeling environment. These include (1) continuing with the current status quo approach; (2) adopting a single model-, data-, and user-interface-integrated framework; and (3) using separate tools to integrate models, data, and interfaces. On the basis of our experience and discussions about three types of software architecture, we created a table listing the criteria and attributes of each architecture (Table I).

In the current approach, features, tests, and documentation are added in a piecemeal fashion. The interface, data, and model are integrated for a specified set of objectives. The end user's experiences and contexts are considered when the model options, interface, and result visualizations are being designed. This process tends to encourage code that is unwieldy, as new features are added without the code being redeveloped and modularized. Software breakdown is not similar to mechanical breakdown (i.e., initial break-in period, period of stable performance, and then mechanical wear), because software does not wear down or change. However, by introducing changes to the software, undesired entropy can be easily introduced into the original design. Technologies in the software application environment can change, causing more effort to be expended just to maintain operation of the software. This situation does not necessarily occur in object-oriented software. Reliance on some commercially developed tools can lead to a need to constantly upgrade even well-written software, just to maintain its operation in a changing technological environment.

In a single integration framework, development and testing are divided into two levels: (1) development of modules and (2) end-user integration and implementation through a single visual programming framework. This framework works as long as it is flexible enough to meet various needs. However, it is very difficult to leverage new technology within the framework, since the user-interface, data manipulation, and modeling connections are already specified and implemented. This system can facilitate the exploration of a specific environmental problem by a single end user but can cause difficulties for a user community whose members are trying to follow a regulatory process. Also, the burdens of model integration and application are on the end user. (Frames 1.1 and GoldSim [2,3] are examples of this type of system.)

**TABLE I Criteria for Three System Architectures**

| Criteria                          | Current Approach  | Open Architecture  | Specified Visual Programming Framework   |
|-----------------------------------|---|--|--|
| Description                       | Features, tests, and documentation are added in piecemeal fashion. Interface, data, and model are integrated for a specified set of objectives.                 | Development and testing are divided into three levels: modules, integration, and end use. This approach allows module reuse and swapping and provides the ability to develop flexible end-user interfaces and data management. | Development and testing are divided into two levels: (1) modules and (2) end-user integration and implementation through a single visual programming framework.                                      |
| Maintainability                   | Since modular design was not the main focus, it is sometimes unwieldy to integrate and test new functions.  | Modules are maintained by the developers. Standards are agreed to and followed in module and data specification. Integration can be done in a number of ways depending on the requirements.                                    | The framework must have one specified standard. All codes must go through the standard to be incorporated.   |
| Dissemination                     | Downloading and installation are done at user's initiation  | Modules can be distributed with the integrated application. Later modules can be maintained on distributed servers.  | Framework and modules are installed separately.  |
| Validation and verification (V&V) | Verification is done for a complete specific version and maintained incrementally. Validation can be done on limited aspects of the model.                      | Each module maintains its own V&V. Applications connecting the modules are implemented by integrators who ensure that assumptions are appropriately compatible for the application V&V.  | Modules can be V&V'd, but V&V of the integration process is up to the end user.  |
| Flexibility                       | Adding new features and functionality is difficult, requiring changes throughout the code and careful consideration of many of the obscure details of the code. | Modules can be added, substituted, and modified with flexible connections to other modules. This practice allows for flexibility in both the module level and the integration level.   | Modules can be added as long as they fit the framework's fixed structure. Modules cannot be flexibly integrated for other potential integrating frameworks.  |
| Use of legacy software            | It is very difficult to integrate two legacy codes and maintain the assumptions and user interface.   | Legacy code can be "wrapped" for use with other codes. Some functions can be called separately. A modularized version of the model would be more flexible.   | It is difficult to incorporate legacy code without a large effort to modularize it to conform to the framework's fixed structure.  |
| Support for cooperation           | It is very difficult to maintain one code that serves two agencies with different requirements.   | Modules and data can be shared for different applications. Different applications can be constructed with the shared modules to accommodate the different requirements of the agencies.  | All agencies can develop their own modules, but they must conform to the framework structure. It may be difficult to construct one structure to satisfy the needs of all agencies and organizations. |

**TABLE 1 (Cont.)**

| Criteria  | Current Approach  | Open Architecture   | Specified Visual Programming Framework  |
|---|---|---|---|
| Development costs                               | Development is inefficient because new features must be highly customized for each application.   | Modularization and structural flexibility lead to efficient reuse and development of modules while maintaining an efficient user interface. | Modularization leads to more efficiency, but effort can be expended on conforming the modules to a structure that is not efficient and effective.                                 |
| User interface support for regulatory processes | Support is good since the user interface can be tailored to control the input and reporting process.  | The flexibility of the connections allows process and user interfaces to be well defined.   | The end user can gain understanding through quick exploration of models; however, to implement regulatory processes, the user must invest effort to visually program the process. |
| Platform independence                           | Platform is dependent on the operating system (OS) and developer's tools. Development can be hindered by uncontrolled changes by the supplier of the OS and integrated development environment (IDE). | The modules can be platform independent. The connections between them are flexible and can be implemented with many technologies.           | Platform is very dependent on the structure of the framework. Development can be hindered by incompatibilities of the model or process to be used and the framework.              |

Quite a large set of tools is being developed to further separate the roles of modelers and integrators and the four components (data, models, interface, and connection). Some model integration tools include the Argonne National Laboratory (ANL) DIAS system (4) and the U.S. Environmental Protection Agency (EPA) MIMS system. These tools offer a system of utilities for model integration and data communication. The DIAS tool is based on the concept of using models to provide methods for a higher-level conceptualization of an object. This allows both new development and the wrapping of existing models. However, there are many other ways to accomplish this wrapping and object integration with commercial tools (J2EE, ColdFusion [5], Microsoft [MS] .NET [6]) that might not supply the same utility support but that do allow a flexible integration with commercial components.

Besides these model integration tools, there are also tools that allow user interface and data management module reuse and swapping. The user interface and visualization aspects are crucial and difficult to construct with automated tools. The interface usually requires a great deal of customization to ensure that users understand the data, options, work flow throughout the process, and model assumptions.

One commercial system that seems to have a good approach is the Environmental Systems Research Institute ArcIMS system (7), an Internet-based system for supplying GIS maps and data. The main map-rendering application is deployed on a server. The developer works with this service and is supplied with a default set of tools to develop a user interface for the manipulation and display of the maps. The interface components are object-oriented but written in a client scripting language (JavaScript). This allows the component provider (ESRI) to provide a flexible template to the integrator to customize the user interface for the end user.

### **Proposed Solution**

The proposed solution includes developing strategies and guidelines for separating the software package components into a set of layers and identifying roles for model development and use. The strategies can apply to

both the modification of existing codes and practices and the development of new models and components. While the traditional method integrates these elements into a single code, the modularization approach instead aims at building a code system consisting of components that can be used and reused for various purposes.

Over the last six years, a group of agencies (U.S. Department of Energy [DOE], Nuclear Regulatory Commission [NRC], EPA, U.S. Department of Defense [DOD], and others) has been informally discussing the problems in linking models for complex simulations of the environment. This group met in March 2000 for the Environmental Software Systems Compatibility and Linkage Workshop (2). The group has continued its work, and members attended a meeting in June 2001 to discuss understandings, accomplishments, and future paths. One objective was to establish categories of attributes for software systems. These included a set somewhat similar to the standard layers described above: (1) model connectivity (model layer), (2) information architecture (data layer), (3) framework connectivity (presentation layer), (4) web-based access (network layer), and (5) system functionality (application layer).

### **Roles**

Three sets of roles are proposed for developing and using the system. First, modelers should develop domain-specific models and document their assumptions. Second, integrators should create an application from the available models and data. The integration environment would be up to the integrator (i.e., there would be no single integration framework, so the system could be done in a web environment [e.g., Active Server Pages or ColdFusion], as a window standalone, or as a hybrid using web services). Third, end users should then specify the data and options through the integrated user interface and communicate the results to the regulators and public.

The models should be available for all to use with technology like an application programming interface (API) for a modular class library or like a set of services (data and model) from a server. Use of generic commercial technology would allow components developed in different laboratories with different software languages to be integrated. This coordination would require the laboratories to work cooperatively in defining appropriate scales, aggregations, and assumptions. The effort would include the development of new models and data and the opening of existing codes.

To maintain the most flexibility and cost-effectiveness, the integrators should leverage commercial technologies. There are quite a few commercial tools to support these tasks, and the technology is rapidly changing, being mostly driven by business software. These tools can be utilized in these systems so that the focus can be on the models and data and on facilitating user understanding for effective decision-making.

### **New and Existing Software Development**

In this paradigm, an interagency committee specifies the standards for communicating between codes and establishes the minimum validation and verification (V&V) process required for individual components. Code developers at specific agencies or institutions maintain control over their own sources but use the specified standards for enabling their codes for inclusion in other applications. While this approach would work best with new code development, methods for "wrapping" legacy code that allow reuse of existing code are available. This structure would make it possible to embed distributed components within everything from simple spreadsheet applications, to commercial GIS packages such as ArcView<sup>TM</sup>, to a visual programming environment, if that was desired.

Taken one step further, components for modeling, presentation, and data access could be maintained at remote locations through the use of MS COM objects or Java servlets. Their integration could be accomplished with software tools such as CORBA, DCOM, and RMI. Communication between components could be accomplished with APIs using evolving standards such as Interface Definition Language (IDL) and eXtensible Markup Language (XML). The use of distributed objects would even make it possible for object components and data to exist on different distributed servers, guaranteeing that the user would have access to the most recent, validated version of that code and supporting parameter data sets.

### **TECHNOLOGY DEMONSTRATIONS**

Technology options in the various levels (data, models, presentations, applications, and network) were explored, and demonstration projects were created to show and evaluate their potential. This effort was not just model integration

(e.g., connecting the output from one model to the input of another). It constituted package integration, since each applications package usually comes with a data set, a set of models, and an interface to interact with the data and model specifications. The process of integrating packages involves model integration (connecting inputs into outputs with the appropriate control structure), data integration (determining which subset of data to use and how the remaining data get mapped), and presentation integration (determining what inputs are required, how to maintain support, and how to view and navigate the output). The connection adds one new aspect: how to integrate the components separately and then integrate the levels into a new package. Since the packages might be served from different locations, a new aspect of network connection also is involved.

Three demonstration projects were chosen both to address a current need among radiological analysts and to be potentially useful in later applications. The projects demonstrate the wide variety of integration techniques and ways to use components based on existing software packages, new models, and commercial components.

- *Low-level landfill analysis:* The NRC's DUST (8) package and a modified RESRAD-OFFSITE package (9) were integrated into a desktop application, with DUST providing a leaching source term to the groundwater and RESRAD providing the multipathway dose assessment from that point. The user-interface and model assumptions were maintained.
- *MARSSIM analysis with RESRAD:* The RESRAD model was wrapped with a preprocessor and postprocessor for web execution. The pre- and postprocessors allowed simple connections to a customized, simplified, web-based user interface and commercial visualization graphing and GIS packages.
- *Nuclide web service:* A nuclide data component was developed to allow common access to applications of data structures. The nuclide data were obtained from a distributed server and used by a local application that could then manipulate the nuclide structure in a common technique.

### **Low-Level Landfill Analysis with DUST and RESRAD**

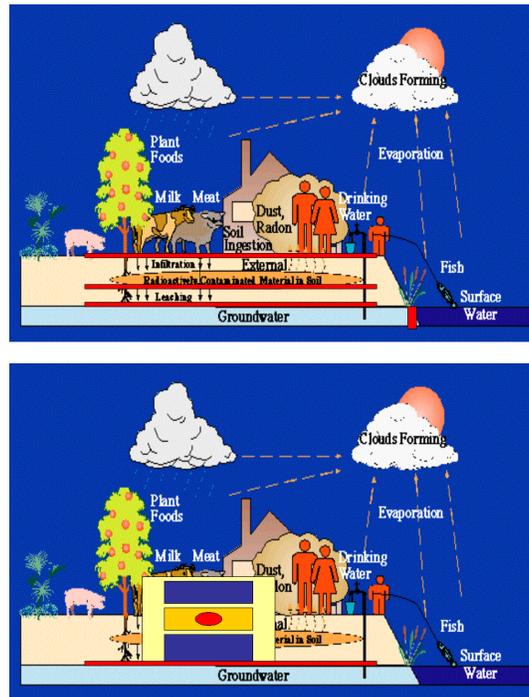
The integration of RESRAD-OFFSITE and the DUST computer code allows users to perform risk analyses of potential releases from low-level radionuclide landfills. The integration was made possible because the RESRAD-OFFSITE model has a feature that allows intermediate contaminant fluxes to be output or serve as input for the remainder of the calculation. In this case, the DUST code was used to generate a modeled release flux at the bottom of the landfill (Figure 2). This flux was then used by RESRAD-OFFSITE for dose risk analysis through the groundwater pathways, including the drinking water pathway and pathways associated with the use of contaminated irrigation water.

To accomplish this integration, the user interface, model interface, and data components had to be separated for each model. Then each component was integrated and packaged in a new application. This practice maintained the data integrity, model assumptions, and ease of use.

The DUST data, model, and interface were separated into three components by designing a database to maintain the metadata of the input parameters. This table included names, units, values, and bounds and could potentially include the grouping and distribution of the data for uncertainty analysis. From these data, RESRAD-OFFSITE was wrapped to allow the user to specify specific configurations of input/output flux planes and the remainder of the input parameters. The output files could be parsed for specific information to be passed to the next software component. The database formed a simple common connection for managing and storing data.

User interface components were developed in both MS Visual Basic and Allaire's ColdFusion. In Visual Basic, the developer could place one of the controls on a form and then associate it with a record in the parameters database table. The appropriate data type and functionality were then automatically packaged in the control. In ColdFusion, the database was likewise used to form data input screens on the basis of a generic database table. Since the user interface is important in communicating model assumptions and facilitating the user's understanding of the model, its development is usually costly. (Sometimes more than half the development time is devoted to custom-developed

user interfaces that use standard components. The use of the custom controls and a custom database should enhance the development of user interfaces and reduce the cost of integrating them.)



**FIGURE 2 Integrating RESRAD-OFFSITE and DUST (The RESRAD-OFFSITE model was modified to accept and output fluxes at certain points in the transport process, including at the groundwater table interface. The DUST results of the fluxes at this location were passed to RESRAD-OFFSITE for further transport and radiological data and risk estimates.)**

The user interface can quickly gather data from a user and display them in the database. To wrap the code, the input data must be extracted from the database and passed to the model. Unfortunately, many existing models use a set of formatted input files to pass data to the calculations. This practice makes the conversion from the database to the model difficult, although ideally it only has to be done once to wrap the component. The conversion from the database to the DUST input file was tedious because of the formatting, data array structures, and exceptions. Some other codes like the RESRAD family of codes use the FORTRAN NAMELIST format, which allows flexibly formatted input files somewhat similar to simple XML files (i.e., the parameter name and values are passed without a highly specific format). A full wrapping of the RESRAD or RESRAD-OFFSITE code was not performed; however, in the second project, this technique was used to pass a subset of the RESRAD data through a similar database and onto the wrapped code, which used the database to modify a template NAMELIST input file.

### MARSSIM Analysis with RESRAD

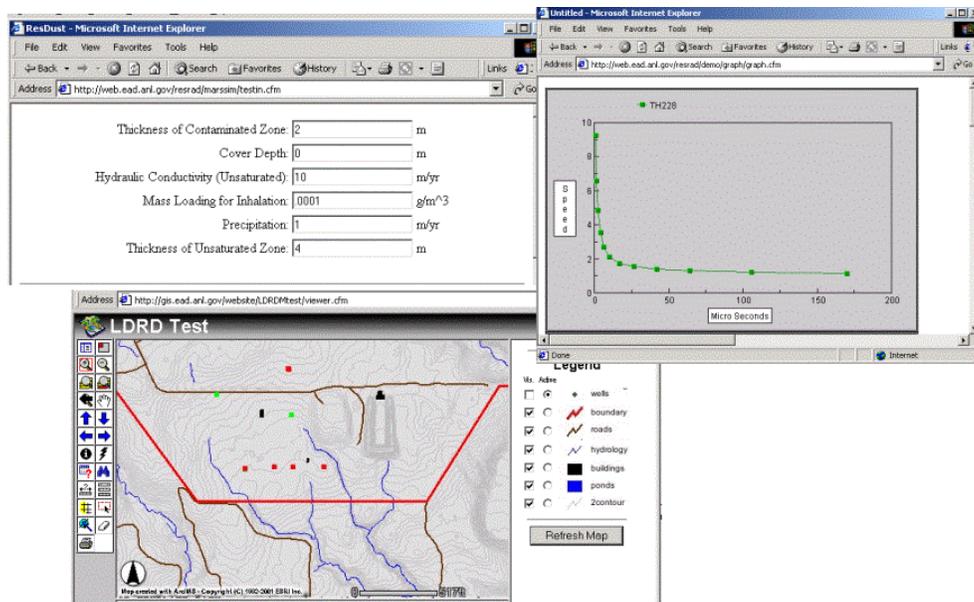
MARSSIM is a recent multiagency procedure for finding statistical determinations for radiological cleanup standards. These standards are based on the statistical level of the detector and dose estimates from finite contaminated areas. RESRAD was developed to address cleanup guideline limits for a set contaminated area. As the contaminated area increases, the dose from the contamination reaches a limit. For smaller areas, the doses are smaller as a result of various scaling factors for the different pathways, such as direct external exposure, food growing activities, dust inhalation, and ingestion. To support MARSSIM activities, RESRAD could be used to generate the dose (or guideline limit) as a function of area. These guidelines could be displayed on a graph and then interpreted on a GIS display of the site with overlain measurements.

While there is a current technique for performing this type of analysis, it is not explained in the RESRAD manual, nor is it a function that the RESRAD interface was designed for. Also, the data visualization of RESRAD was not prepared for easy incorporation into graphs and GIS systems. The purpose of this project is to demonstrate the simple reuse of an existing software application (RESRAD) by wrapping it into an object with a database-driven interface to allow use of custom interface controls. The database interface also allows for easy integration of the model into commercial visualization packages for plotting and GIS. Furthermore, the results can be made accessible on the Internet through a simple web browser interface, giving users easy access to the model, data, and visualizations.

Next an input and output interface must be developed for the system. The interface can be quickly developed through the interface components for ColdFusion. The ColdFusion template file contains setups for all enabled parameters from the specific database table. The data are collected and validated from a generated HTML page. The user sends the specified input data through the Submit button. On the server, the data are placed in the database, the RESRAD component is executed, and the results are stored in the database.

There are many alternatives for viewing the results. ColdFusion and similar technologies (such as Active Server Pages) have suites of tools for customizing a drilldown report in HTML or in a reporting tool like Crystal Reports. Other commercial tools can generate a more visual display. For this project, the PopCharts Internet graphing package (10) and the ArcIMS Internet GIS package were connected to the results database (Figure 3). PopCharts allows results data to be quickly incorporated into a graphing template to show the area factor versus area. The ArcIMS GIS package similarly allows measurement data to be displayed with symbols (type, sizes, colors) on the basis of the measurement level compared to the critical cleanup criteria identified in the analysis. Both of these tools were implemented to generate standard HTML pages. The ArcIMS tool uses extensive JavaScript to enable the user to interact with the map that communicates the data via XML to the mapping server. The measurement data are placed in an acetate layer on top of the map, while the user is still able to interact with the measurements and find out more about them by clicking on them.

This simple demonstration showed how to wrap an existing code, simplify the interface and its construction, and use commercial visualization tools to view all the data integrated on a server. Many extensions could enhance this project: the component could be optimized for performance on a web server; input options could be made available to the user; and the user could be allowed to manage the GIS data more. As is the case for most software development projects, many options are available, but the choice depends on the tradeoffs among data access, complexity, and security involved with a specific project.



**FIGURE 3** Connecting Graphics Packages to Results Database (Input and output web pages for the MARSSIM project show the simplified RESRAD input interface and the results visualization.)

## Nuclide Web Service

The previous two projects demonstrated integration on a desktop and integration on a server with web access. Some research modeling environments use components on a set of widely distributed computers. These distributed computing environments allow standard components to be maintained on a few servers that are optimized for performance and maintained with the current versions. Data and models from various sources can be easily connected, and the data can be communicated across the network.

It would be useful to have a set of data sources for contaminant standards or specific data sources from sites that could be integrated and analyzed by means of model and visualization techniques. Tools are currently being developed to facilitate these connections while still maintaining system performance and security. Early connections were difficult to develop and maintain. Even after the connections were constructed, there were frequent disruptions in them that resulted from new security techniques such as firewalls (which block communication for standard connection techniques such as DCOM, CORBA, and Java RMI). One example of a new type of tool is the MS .NET web service that was scheduled for release in late 2001. This technique utilizes a standard SOAP protocol to connect components on distributed computers. The communication between the computers is accomplished by using other standards: XML for the data structure and HTTP for the transmission protocol.

Not only do these services facilitate the use of distributed components, but they also enable simple integration of components. For example, a web service developer can generate a .NET web service by simply constructing an object in the standard MS Visual Studio Integrated Development Environment (VS IDE) by placing a <webMethod> tag in the method to be made available for the service. A test page and site are automatically generated to ensure proper testing and availability of the component.

To use or “consume” this web service, an application integrator can use the same VS IDE on a local development machine. To find out what services are available from a particular server, the VS IDE can simply request that information from the server and present the results, which are integrated into local components available to the integrator. In other words, the web service looks similar to local components. From then on, the development of the integrated package is transparent. Calls are written to the remote object, and the developer no longer has to be too concerned about the computer “handshaking” and passing of data via XML to the server.

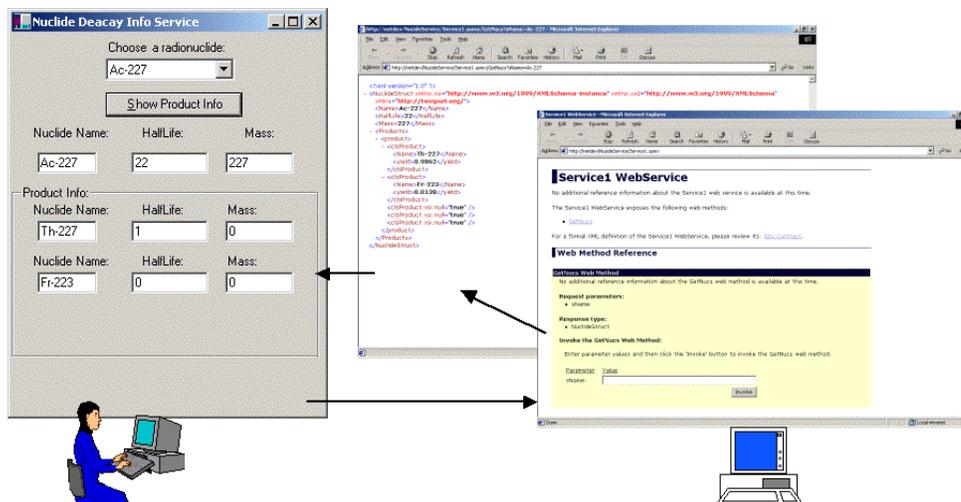
Nuclide databases might be a good candidate for a web service. In radiological assessment software, the handling of nuclide data is difficult because of the decay chains and different assumptions about secular equilibrium. Some data are quite standard, like the EPA’s Federal Guidance Reports (FGRs 11, 12, and 13). It would be useful to have a web service supply the data in a common form with common functions. Local components could then be generated and shared to handle the nuclides in a common fashion, so that radiological codes could better interact with similar assumptions and handling mechanisms.

A simple web service was set up with a limited set of nuclide data to demonstrate their workings (Figure 4). A method that would take input on a radionuclide and deliver decay chain information was developed. The data were recursively extracted from two database tables. One had the nuclide information (e.g., mass, half-life, dose conversion factors, and distribution coefficients for various media). The second detailed the decay relationship, with primary key fields for the parent nuclide and the progeny nuclide and also a field for the yield (or fraction of the decay that followed that decay path).

These data were automatically inserted into an XML package to be sent back to the requesting computer. Once received, the computer would automatically unpack the XML package, and the data would be ready for use in the local machine. In this project, the data were organized into a linked list of nuclide structures for easy handling in codes. For example, in many radiological codes, the decay process is now handled through many indices that are very difficult to maintain and update. The linked list is simpler to handle and less error-prone, and it facilitates simpler code because indices have been removed and because it can be manipulated with recursive techniques.

These codes were implemented with the Public Beta 1 version of MS .NET. Figure 4 shows the web service method, the testing page from the server, the consumer software, and the consumer interface. Again, not much is shown,

because much of the infrastructure that is needed to implement this web service is within the .NET environment, a commercially available integration package. MS is not the only package available; there are also Java-based packages. It is hoped that through the use and deployment of standards such as SOAP and XML, these web services will be able to interconnect and operate on a variety of servers.



**FIGURE 4** Developing and Using a .NET Nuclide Web Service (The object and methods are placed on the server. A web page allows manual checking of the function and demonstrates that XML is used to deliver the results. In the MS VS IDE, the reference to the distributed components can be easily found and inserted. These components are used in the code in a manner similar to that in which the local components are used. The resulting application allows access to distributed data that are used to construct an easily handled linked-list nuclide structure.)

## CONCLUSIONS

### Demonstration Assessment Versus Criteria

The above projects demonstrated a small part of the potential for component-based environmental modeling with open integration. Components were developed for the user interface, data handling, model wrapping, connection via desktop, web server, and distributed computing environments. An assessment of this system with regard to the issues mentioned in Section 1 follows here.

- Flexibility and maintenance:* The system's flexibility was demonstrated when a similar component (RESRAD) was both incorporated in a desktop model integration package and integrated on the web with commercial visualization tools for a different purpose. This integration allows a wide variety of models and end users to be supported. As new components are developed, it is anticipated they will interact so that the applications can be modified to incorporate new model options, data input options, result viewing options, regulatory changes, and tools quickly and efficiently. The open integration environment allows commercial tools to be incorporated rapidly and new Internet options, such as web services, to take advantage of software and hardware (e.g., bandwidth) technologies.
- Software dissemination:* Barriers to the dissemination and installation of the code package (data, software, and documentation) vary. For desktop systems, some installation packages are initially large but can be reduced for upgrades. Web-based systems require little or no software to be installed other than a standard web browser. Again, web services also require little or no software to be installed and also allow local specialized maintenance of the components and data at the distributed sites, so they can be updated without requiring users to download patches. The web-based nature of the download and software package allows web-

based user communities to grow, which can generate a critical mass for the use of the component systems.

- *Quality assurance (QA)*: The quality of individual model and data components can be assured by the developers. The integrity of the model and data assumptions in the integrated package can be checked by the integrators. Thus, data validity and scenario applicability are left to be determined by the end user. This distributed QA can be more effective, but procedures should be in place to properly document the models so that QA remains intact throughout the process. Development of a large user community would allow discussions and testing of various features and application contexts.
- *Life-cycle development and maintenance costs*: There are always tradeoffs in determining how general a component should be. The flexible system of component integration allows trials to determine the granularity level to which the components should be specified. For example, some models, such as an external exposure model, might require a generally inefficient model and methods that use preprocessing to make the model more efficient for a more constrained problem. Long-term cost savings might be anticipated from this approach over the traditional development approach. However, the total life-cycle development and maintenance costs will depend on future needs and have yet to be evaluated and assessed.
- *Platform reliance*: As the projects demonstrate, there are many ways to construct, integrate, and execute components. The technologies being developed are becoming more and more open, which means there is less dependence on any one technology. However, as previous component development suggests, there are pitfalls, especially when components are not upgraded and integrated with commercial technologies.
- *Transparency*: Tools can be integrated both for exploring data and for facilitating understanding (e.g., sensitivity analysis, uncertainty analysis, and visualization [graphing and GIS]). By incorporating a middle-layer integrator role, the system will present the end user with an integrated package that can preserve the model's assumptions, level of data, and conservatism while still implementing changes in the regulatory process. This type of package will enhance public acceptance and confidence if the model needs to be publicly used or defended.
- *Ease of use*: The flexible integration technique, coupled with the development of interface components and a standard metadata database structure for input parameters, means that user interfaces can be effective (i.e., customized for the needs of the user by being facilitated through the components). Work on the user interface will still be one of the main efforts in model package development, but the use of the components and commercial packages will allow interfaces that will convey an understanding of the model and its results to the variety of stakeholders who are using the environmental modeling system.

### **Technology Uncertainties and Risks**

Many technology uncertainties and risks surfaced earlier. Many of these issues are addressed by an open system where (1) there is separation of the modeler and the integrator and (2) the modeling and integration tasks can be done with different tools. Such a system also allows a transition pathway that utilizes existing code while concurrently developing new module code. It also allows the integrator to focus on the user's specific need, whether it is for a detailed analysis without a "big picture" understanding or the ability to navigate around issues while applying regulatory requirements to analyze a specific site.

Incorporating many integration techniques also reduces the possibility that maintaining environmental models will require a large effort. The example of wrapping old FORTRAN codes into objects demonstrates that these techniques allow codes to be used over a wide range of technologies. Using commercial tools and standards allows much of the work to be done by others.

There are some drawbacks, however. Sometimes the technology can be under such rapid development that an integration system might depend on a commercial tool that is supported for only a short amount of time. It is hoped that the components could be developed to be flexible enough so they could be easily incorporated into the new system. A couple of examples of this situation are found in running the FORTRAN code within the RESRAD system. Originally a way was found to run the calculations in the background and still maintain communication with the user interface, which allowed for integrated feedback on the status of the calculations. As the MS operating systems changed from Windows 3.1 to the current Windows version, in almost all upgrades, this code for accomplishing the integration of the model and interface had to be revisited to ensure proper operation. This effort was complicated by the fact that the wide user base meant that many Windows operating systems were being used at the same time, so a way of operating the integration system in not just the new environment, but in all the prior operating systems, had to be found.

Another example was a Java applet written for Internet mapping applications that depended on the server providing the connection to the database. The language rapidly changed, and substantial effort was required to upgrade the original code. In time, the data connection utility was supported and sold only for the new versions of the language (and then dropped, as it was no longer needed in the newer versions). The original applet code was thus somewhat inflexible, depending on the obsolete commercial component, unless effort was made to upgrade the versions of the language.

Also, as technology progresses, changes in application requirements occur not only in response to changes in environmental applications or regulations but also in response to technology changes. Just recently, many requirements have been imposed on government web sites with regard to security, privacy, and accessibility to those with disabilities. However, not all the requirements were imposed with the impending increase in the bandwidth of the Internet in mind. New technologies and techniques based on the connection speeds of the networks will become available.

## RECOMMENDATIONS

Component-based environmental modeling offers many advantages as long as the hazards in developing the system are dealt with. An open system of components and integration techniques offers the hope of addressing issues in an open and shared environment to leverage existing codes in multiple integrations. An open system allows the sharing of models, data, and interface components for many integration techniques.

One application of this open system would be to integrate the models and data into a flexible system that would be able to deliver information and facilitate understanding over a long period of time. The models and data could be integrated into the decision-making process and field measurements to elicit dynamic feedback on the environmental state of the system, the model, and the supporting data.

On the basis of the above discussion, the following four recommendations are made:

1. Develop an interagency consensus on future modeling needs.
2. Maximize the use of technologies developed by the software industry.
3. Maintain the integrity of legacy codes.
4. Minimize dependence on a particular system.

## REFERENCES

1. Constanza, R., et al., 1993, "Modeling Complex Ecological Economic Systems," *BioScience* 43, Sept.
2. Whelan, G., et al., 1997, *Concepts of a Framework for Risk Analysis in Multimedia Environmental Systems*, Pacific Northwest National Laboratory, Richland, Wash., Oct.
3. Whelan, G., and T. Nicholson (editors), 2001, *Proceedings of the Environmental Software Systems Compatibility and Linkage Workshop*, Draft, Pacific Northwest National Laboratory, Richland, Wash., June.

4. Sydelko, P.J., et al., 1999, "A Dynamic Object-Oriented Architecture Approach to Ecosystem Modeling and Simulation," in *Proceedings of the 1999 American Society of Photogrammetry and Remote Sensing (ASPRS) Annual Conference*, Portland, Ore., May 19-21.
5. Forta, B., 1998, *The ColdFusion Web Application Construction Kit*, Que, Indianapolis, Ind.
6. Hollis, B., and R. Lhotka, 2001, *VB.NET Programming with the Public Beta*, Wrox Press, Birmingham, U.K., Feb.
7. ESRI, 2001, *ArcIMS 3*, Redlands, Calif., <http://www.esri.com/software/arcims/>, accessed Sept. 2001.
8. Sullivan, T.M., 1993, *Disposal Unit Source Term (DUST) Data Input Guide*, NUREG/CR-6041, BNL-NUREG-52375, Brookhaven National Laboratory, Upton, N.Y., May.
9. Yu, C., et al., 2001, *User's Manual for RESRAD Version 6*, ANL/EAD-4, Argonne National Laboratory, Argonne, Ill., July.
10. Corda Technologies, Inc., 2001, *PopChart Online Documentation*, Lindon, Utah, <http://www.corda.com/support/docs/>, accessed Sept. 2001.