Geospatial Interoperability in Modeling Frameworks - The 'GEOLEM' Approach

O. David[a] **, R. J. Viger**[b]**, I. W. Schneider**[c]**, and L. Garcia**[a]

[a] *Colorado State University, Fort Collins, CO, U.S.A.*

[b] *US Geological Survey, Denver CO, U.S.A*

[c] *USDA Agricultural Research Service, Great Plains System Research, Fort Collins, CO, U.S.A*

Abstract: Environmental models and modeling frameworks (MF) typically do not represent geographic information in a way that enables the direct translation of this information between geographic information systems (GIS) and the model or modeling framework. Parameters of the characteristics of geographic features are processed as part of a model's mathematical solution and are thus explicitly represented in the model. In addition to the lack of semantic definition of geographic information in the environmental modeling process, current modeling approaches suffer from a lack of interoperability. The GEOLEM (Geospatial Object Library for Environmental Modeling) project forms a interagency working group which implements GEOLEM as a middleware solution (i) for the definition, storage, and manipulation of geographic metadata and (ii) for the transformation of information from the form of one context into another based on metadata specification (e.g. from the spatial data formats of GIS into the parameter organization of an environmental model). The purpose of this system is to eliminate the need for GIS-specific knowledge in the modeling framework and model-specific knowledge in the GIS. More specifically, GEOLEM will result in Modeling Frameworks being able to specify methods conceptually for the (i) Delineation of geographic features, (ii) Parameterization of geographic features, (iii) Visualization of model and GIS data entities, and (iv) the Exploration of model and GIS data entities.

Keywords: Modeling Frameworks, GIS, Interoperability, Modeling

1. INTRODUCTION

Environmental models and modeling frameworks typically do not represent geographic information in a way that enables the direct translation of this information between geographic information systems (GIS) and the model or modeling framework. Parameters of the characteristics of geographic features, such as area and volume, are processed as part of a model's mathematical solution and are thus explicitly represented in the model. Information about the specific geometry or location of the geographic features whose behavior is being simulated is rarely represented explicitly in the model. The specification of methods needed to generate parameters describing the geographic features, or to actually delineate those geographic features in the first place, are never represented within the model. Further, the semantic definition of geographic feature types and their parameters are not represented within models. While environmental models are not expected to actually derive input geographic information, they should provide information about how to derive that information.

In addition to the lack of semantic definition of geographic information in the environmental modeling process, current modeling approaches suffer from a lack of interoperability. In all cases where GIS data are used in an environmental modeling context, it is necessary to develop algorithms or methods that provide for the translation between the two representations of spatially relevant information. Translation algorithms developed to date have been tightly designed to the needs and characteristics of specific models and GISs. As a result, two increasingly important types of problems present themselves: (i) resource, (ii) technical. The resource problem occurs because the translation algorithms are not, by and large, reusable. This

means that each connection between a specific pairing of an environmental model and a GIS requires a unique translation algorithm, which, in turn, requires new resources to repeatedly solve the same conceptual problem

The more significant problem resulting from incompatible translation algorithms is technical in nature. Because each translation algorithm is unique to a model/GIS combination, sharing and coupling of environmental models and GIS methods is hindered. Once an environmental model or modeling framework has been "wired" to a GIS, it should automatically gain access to a wide array of community-developed geoprocessing libraries and geographic visualization software. To achieve this, standard protocols should be established for translating spatial information between GIS, environmental models, and other tools common to environmental modeling tasks. In addition, a metadata nomenclature should be established for referencing the array of geographic feature types relevant to environmental modeling. With these standards in place, generic sets of information translation adapters can be developed such that once a modeler has "mapped" their unique nomenclature to the standard nomenclature, transfer of information to and from GIS can more readily be automated. In addition, whenever a new geo-processing or visualization tool is developed in conformance with these standards, the tool is immediately available to the larger community.

2. BACKGROUND / RELATED WORK

Despite the lack of semantic description about geographic information in simulation models, tighter integration of GIS and simulation models has been sought. One approach has been to put "the model in the GIS". Although GIS and computing resources have evolved to a point that implementation of an environmental model within a GIS is possible, programming within the confines of a GIS has not become the norm. This is largely due to the relatively poor computational efficiency of the programming languages of most GISs and the burden of interacting directly with GIS data structures. Developers typically prefer to design models that ingest the simplest form of spatial data possible and concentrate their development efforts on the simulation of environmental processes. Examples of this have ranged from using GIS as a map-based interface for the selection of pre-existing data and model execution (US Environmental Protection Agency) to the creation of model inputs and model execution (Robinson and Mackay 1995) to a full integration of environmental models into spatial decision support systems (Taylor, Walker et al. 1999).

Alternatively, putting "the GIS in the model" has largely been rejected because the complexity of implementing GIS within models or model frameworks is not cost-effective. This approach has been most clearly adopted in groundwater models, where the modeling response units are delineated according to relatively simplistic methodologies such as finite difference or finite element meshes (McDonald and Harbaugh 1988). Although these models do not normally generate the original maps of modeling response units, they do exploit the spatial topological connections between units.

As a result, a third approach has gained popularity. GIS is used as a standalone pre- or post-processor for a model, reducing spatial data to the simplistic descriptions expected by the model. The USGS GIS Weasel is an example of this (Viger, Markstrom et al. 1998). A GIS operator usually works with a modeler to manipulate and digest spatial data into a file or set of files that will eventually be read by the model. The knowledge that was used to apply the GIS appropriately usually resides in the mind of modeler and the GIS operator. This knowledge is not normally formalized or codified. In the case of well-established models, dedicated GIS software applications may be developed as pre- or post-processors. Although these applications do serve to codify the knowledge used to delineate geographic features and derive parameters of those features, they fail to enable the re-use of those geo-processing methodologies in newly created models.

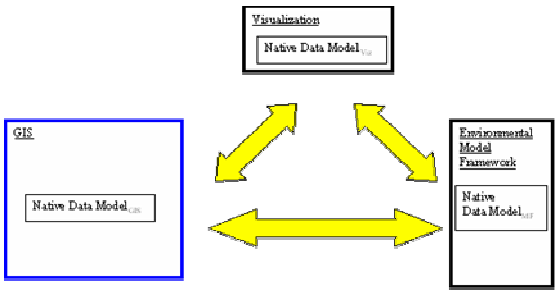

Figure 1. GIS, VIZ, and MF components with distinct internal "native data models" and custom integration directly mapping between specific native data models.

A fourth approach, depicted in Figure 1, seeks a looser coupling of GIS, MFs, and visualization software (Viz). This configuration relies on communication between discrete software components, rather than merging functionalities of

disparate components into a monolithic piece of software (Leavesley, Grant et al. 1996).

This approach towards the integration of GIS and environmental models will be used as a starting point for the research proposed here.

3. GEOLEM DESIGN

In order to be able to allow software components to more readily interconnect in a generic way, Figure 2 shows a middleware architecture that allows the most effective data model for each software component to continue being used by each respective component, yet facilitate the movement of information across these contexts. This middleware is referred here as the Geographic Object. The authors seek to leverage the ideas of the OGC Geographic Object initiative (OpenGIS Consortium 2003), and participate, if feasible, in this effort. One way to describe the role of the Geographic Object is that it maps the relevant details of one context to those of another.

3.1 Objective

The objective of GEOLEM is to enable the systematic integration of MFs and GIS during (i) pre-run, (ii) run-time, and (iii) post-run phases of modeling. In addition, this effort seeks to eliminate the need for GIS-specific knowledge in the environmental model and environmental model-specific knowledge in the GIS. More specifically, this effort will result in MFs being able to specify methods for the

- Delineation of geographic features
- Parameterization of geographic features
- Visualization of model and GIS data entities
- Exploration of model and GIS data entities

Central to achieving these goals is the development of template metadata specifications for geographic information to be used by models developed in the MFs. The developer of an earth science model will be able to use previously specified types of geographic features in that model. These template specifications will not significantly alter the style of data model (e.g. arrays of parameters) typically found within traditional earth science modeling components.

3.2 Approach

This mapping is represented by chains of arrows that connect one external component to another. Consider the chain at the bottom of Figure 2, representing the MF requesting information that is generated in the GIS. The MF communicates with the Geographic Object, requesting information in a format suitable to the MF. The Geographic Object understands that the information requested by the MF corresponds to some GIS-based information. Based on this understanding, the Geographic Object requests the appropriate information from the GIS. Once this GIS information is returned from the GIS to the Geographic Object, the Geographic Object then uses its understanding of the MF-GIS correspondence to return a set of MF-appropriate information to the MF. The broad arrows represent component specific communication. The thin, black arrow represents the work that the middleware does to translate information from one context into another. In our example, this could be the reduction of shape files to arrays of parameters.

The benefit of this mapping is that the MF does not need to understand how to make low-level requests to the GIS, nor does it need to know how to extract what it needs from the alien data formats of the GIS. It needs only to know how to make request to the Geographic Object for the higher-level information that it needs.

To extend the sample explanation of the arrow chain started above, let the information requested by the MF be the elevation of hillslopes within a watershed. The Geographic Object will understand that the MF ultimately wants to receive an array of real numbers, because the name of the parameter within the MF, say `elev`, has an association within the Geographic Object middleware to a description which states the numerical format of elev. The Geographic Object will also have a specification that the method used to derive the real numbers in the array `elev` is to find the median value in the distribution of elevations within each hillslope. In addition, the Geographic Object middleware will know about a hillslope and how it should be derived. Although it is obvious to human users that in order for the median elevations of hillslopes to be derived, hillslopes must first be delineated, this information is not known to the MF which simply knows that it needs an array of real numbers. The Geographic Object will manage this relationship

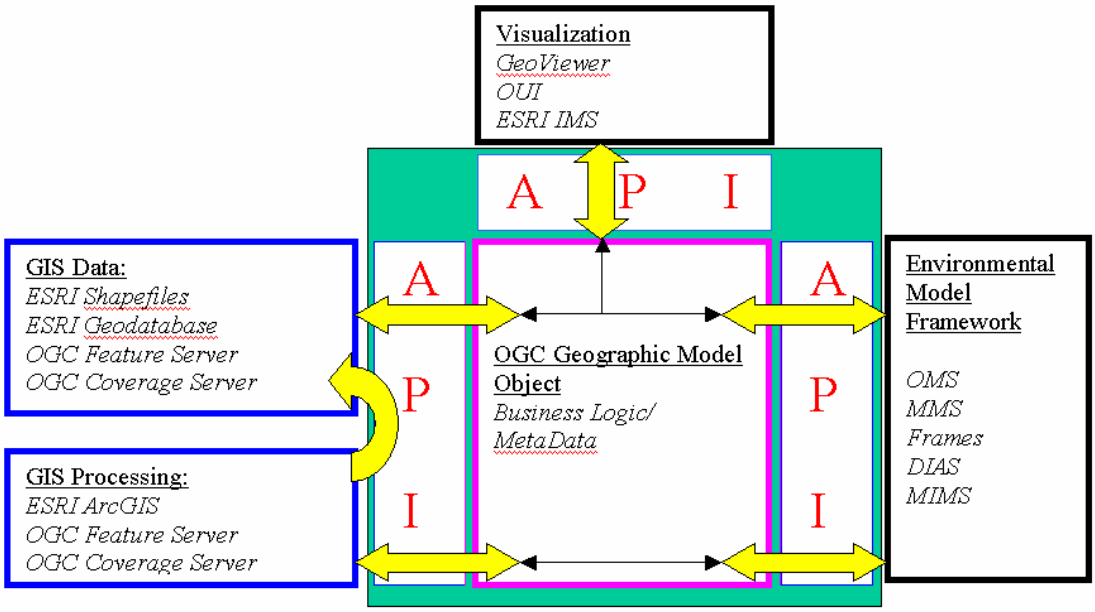

Figure 2. Integration of software components based on middleware (OGC Geographic Object) conceptual model.

4. IMPLEMENTATION

Figure 3 shows how an environmental simulation model can access GIS functionality. The model is represented on the left, in this case by PRMS (Precipitation Runoff Modeling System, Leavesley et al.,1983). The model is intended to make high-level calls to GEOLEM, based on the internal vocabulary of the model. The rationale is to avoid as much as possible changing the internal workings of the pre-existing model or modeling framework. Requests will typically be for parameters. GEOLEM will be able to take these high-level requests and translate their meaning into generic terms.

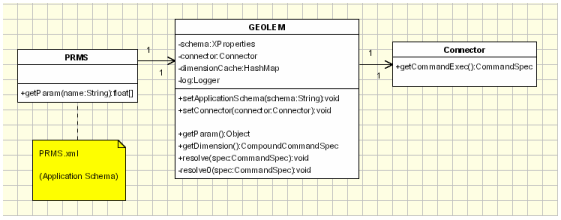

Figure 3. General structure of the connections between an environmental simulation model, GEOLEM, and a GIS server.

There are three different linguistic contexts in the scenario described above: (i) the environmental simulation model, (ii) GEOLEM, and (iii) the GIS server. In order to communicate across these contexts, two different translations are made. The first is from the language of the model into that of GEOLEM and the second is from the language of GEOLEM into that of the GIS server. The first translation relies on what is referred here as the conceptual schema for the environmental model. This is a metadata store that encodes the conceptual model, described in the previous chapter, for geographic information within the environmental simulation model. The conceptual schema is configured to relate the elements of conceptual model to analogs available within the GEOLEM library.

4.1 GEOLEM Functions

The GEOLEM library, which is not shown in Figure 3, contains several types of functions: commands, compound commands, and parameter providers. These functions are intended to provide a generic way to use basic GIS functions. These basic functions can be grouped to develop more complex geo-processing methodologies or they can be used to extract information from a GIS. As noted in Figure 3, GEOLEM is not a GIS itself and relies on the existence of a GIS server with which it can communicate.

4.1.1 Commands

Commands are used here to denote simple, single-step functions. Examples of some commands could

be “calculate aspect” or “derive a watershed”. Figure 4 depicts the class hierarchy used to create such a command.

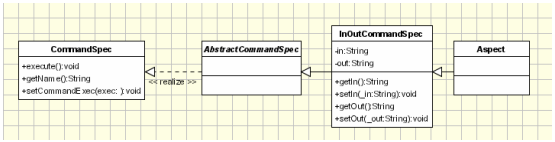

Figure 3. Inheritance Hierarchy of the simple GIS Command, Aspect.

The *CommandSpec* box on the left is an interface, meaning that it is merely a general definition that is not actually implemented. It serves to set a minimum level of functionality that all descendent functions must support. The *AbstractCommandSpec* is the basic implementation of the *CommandSpec* interface. Note that this implementation is denoted by the dashed arrow accompanied by the <<realize>> label. At the right of Figure 3, the Aspect class is what could be thought of as an actual GIS command. It extends the *InOutCommandSpec*, which in turn extends the *AbstractCommandSpec*. Extension, a basic principle of object oriented design, effectively allows new functionality and properties to be added to a general class. The new functionality and properties are encoded in a sub-class. The sub-class, by referencing the more general super-class, will gain all the functionality and properties that existed in that original class. *InOutCommandSpec* merely serves as a helper to add some functionality that is likely to be widely used by other actual GIS commands. Commands like Aspect, Slope, *FlowDirection*, etc. can simply extend *InOutCommandSpec* and avoid having to re-implement the exact same functionality.

This helper implements the setting of the names of inputs and outputs to the command.

- Commands as indivisible, atomic units of geoprocessing functionality
- These devices allow the GEOLEM compound commands to develop sophisticated methods of reasoning, sometimes described as business logic, without the constraints common to many of the languages associated with GIS.

4.1.2 CompoundCommands

Compound commands are intended to allow sequences of simple commands to be created. In addition, compound commands allow logic to be associated with these sequences. Implementations of this interface are intended to provide a way to encode high level representations corresponding to the semantics of a type of geographic feature. The *CompoundCommandSpec* box at the lower left of Figure X shows that this is an interface which extends the *CommandSpec*. Classes that implement *CompoundCommandSpec*, such as *HillslopeMethod*, are able to enumerate all of the *CommandSpec* objects that will be referenced within that class. Put another way, a compound command is able to reveal all of the simple GIS commands that it will use (with the *getCommandSpec*() method). The significance of this will be discussed below. In addition, the implementation of a *CompoundCommandSpec* interface is expected to extend the *AbstractCommandSpec*, thereby gaining access to standard methods such as *execute*().

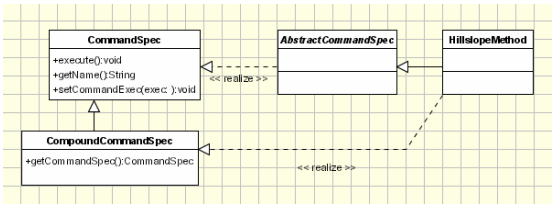

Figure 4. Inheritance Hierarchy of a Compound Command, HillslopeMethod

4.1.3 ParameterProvider

The third main type of function that GEOLEM exposes is the parameter provider. This concept is intended to be a generic way to derive new information, most likely parameters, based on an input map of geographic features and some methodology fixed within the implementation of the parameter provider. This idea is represented by the ParameterProviderSpec interface, shown at the upper left of Figure 5 below. The interface defines two significant methods. The first is getParam(). This method is explicitly designed to return a data object, some form of which will ultimately be returned to the environmental simulation model. Most commands and compound commands return only a character string indicating whether or not an operation has succeeded. The second method, setDimension(), associates the particular parameter provider with the input map of geographic features, alluded to above. The term dimension, introduced in the previous chapter, is used to refer to the map of geographic features that the methodology contained within the parameter provider will be applied to. Figure 5 shows an example of a parameter provider implementation designed to derive the median elevation for a set of geographic

features (each feature is regarded as a zone in this context).

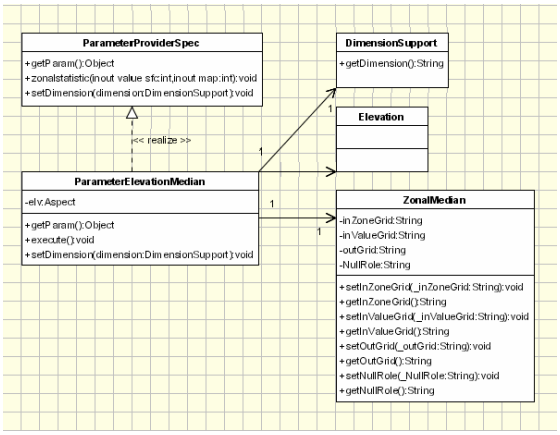

Figure 5. Inheritance Hierarchy of a Compound Command, HillslopeMethod

Besides these three concepts there is the Core GEOLEM class which maps the specifications of basic GIS commands to some GIS server. It uses XML configuration files, which describe how a generic GIS specification gets mapped into a GIS call. This can be understood by a real GIS. Up to now, there are GEOLEM prototype bindings to ARC GIS 9.0 beta via JNI/Python and COM.

5. APPLICATION PRMS

For the PRMS model a following scenario can be applied to derive the parameter hru_elev (the elevation value for each hydrological response unit) using GEOLEM:

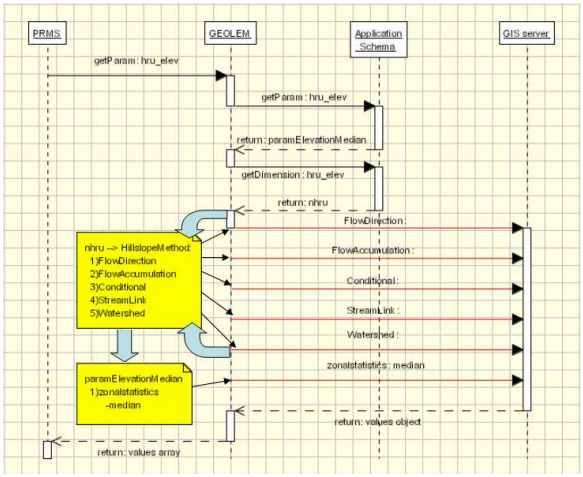

Figure 6. Scenario diagram depicting the PRMS usage of GEOLEM

The delineation *hru_elev* requires the application of *CompoundCommand* *ParameterElevationMedian*. The *HillslopeMethod* itself uses simple GIS commands such as *FlowDirection*, *FlowAccumulation*, etc. and a *ZonalStatistics* command to generate a value array, which can be consumed by the model. The sequence diagram shown in Figure 6 depicts the sequencing of interactions between several components in GEOLEM. The Application Schema object describes in XML the dependencies of the *hru_elev* parameter, its dimensions, data input, and optional unit conversion (Figure 7). Such ApplicationSchema represent the backbone of GEOLEM. They express the model/modeling framework requirements of model parameter data.

```
<xprop name="geolem">
 <xprop name="applicationschema">
  <xprop name="prms">

   <xprop name="hru_elev">
     <entry name="cmd">
   geolem.spec.gp.ParameterElevationMedian
      </entry>
     <entry name="dimension">
        nhru
      </entry>
     <entry name="type">
        double
     </entry>
     <entry name="units">
        feet
      </entry>
   </xprop>

   <xprop name="nhru">
     <entry name="cmd">
       geolem.spec.gp.HillslopeMethod
     </entry>
   </xprop>

  </xprop>
 </xprop>
</xprop>
```

Figure 7. GEOLEM ApplicationSchema fragment for PRMS

A detailed description of the XML syntax would exceed the length and scope of this paper, but can be found under http://oms.ars.usda.gov/geolem.

A model such as PRMS which is using the ApplicationScheme in Figure 7 is only required to add the code

```
double hru_elev[] =
   (double[])GEOLEM.getParam("prms/hru_elev");
```

to obtain the HRU elevation data as a array of doubles. This call causes GEOLEM to invoke the orchestration of the lookup scenario shown in Figure 6 to generate these values.

6. CONCLUSIONS

The GEOLEM effort represents a significant undertaking to simplify the usage of Geographical Information Systems for models and modeling frameworks. It is actually prototyped with frameworks such as the Object Modeling System (OMS) (David et.al 2002) amd PRMS and will be adapted to FRAMES in conjunction with ESRI GIS products. A prototype implementation using the ESRI ARC GIS 9.0 demonstrated the feasibility to delineate and derive parameter for PRMS in OMS.

The overall expected benefits of the GEOLEM project and its implementation can be summarized as follows:

- Easier exchange of scientific expertise due to improved interoperability of spatial modeling applications in modeling frameworks among the agencies and other institutions
- Leveraging and saving the investments being made in simulation model development/GIS adaptation, data-management and visualization for all project partners and avoid duplication of development efforts.
- Highly efficient integration of new spatial simulation models into existing GIS solutions and easier adaptation of models on new versions of COTS GIS packages without model reimplementation.
- Establishing an informal and formal collaboration platform by means of a model metadata standard and template reference library implementation, which is seen as the foundation for other interagency efforts like “Data Representation and Interchangeability”
- Offer model developer a path for “better” handling of meta information for future model application by providing “executable”, operational meta-info; meta-info becomes a part of the execution model, rather being optional documentation “sugar”.

7. ACKNOWLEDGEMENTS

This work is supported by the Interagency Steering Committee (MOU) on Multimedia Environmental Models, Workgroup 1 on ‘Software Systems Designs and Implementation for Environmental Modeling’. It is funded by the Environmental Protection Agency EPA. Thanks to Gerry Laniak (from EPA) for his visionary lead of the working group and the support of this effort.

8. REFERENCES

David O., S.L. Markstrom, K.W. Rojas, L.R. Ahuja, and I.W. Schneider (2002). The Object Modeling System, In: Agricultural System Models in Field Research and Technology Transfer, L. Ahuja, L. Ma, T.A. Howell, Eds., Lewis Publishers, CRC Press LLC, 2002: 317—331.

Leavesley, G. H., R. W. Lichty, et al. (1983). Precipitation-runoff modeling system--User's manual, U.S. Geological Survey: 207.

Leavesley, G. H., G. E. Grant, et al. (1996). A modular modeling approach to watershed analysis and ecosystem management. Watershed '96 A National Conference on Watershed Management, U.S. Government Printing Office.

McDonald, M. G. and A. W. Harbaugh (1988). A Modular Three-Dimensional Finite-Difference Ground-Water Flow Model, U.S. Geological Survey

OpenGIS Consortium (2003). GO-1: Geographic Objects Initiative, OpenGIS Consortium. 2003.

Robinson, V. B. and D. S. Mackay (1995): "Semantic modeling for the integration of geographic information and regional hydroecological simulation management. Computers Environment and Urban Systems 19(5-6): 321-339.

Taylor, K., G. Walker, et al. (1999). "A framework for model integration in spatial decision support systems." International Journal of Geographical Information Science 13(6): 533-555.

Viger, R. J., S. L. Markstrom, et al. (1998). The GIS Weasel - An Interface for the Treatment of Spatial Information Used in Watershed Modeling and Water Resource Management. First Federal Interagency Hydrologic Modeling Conference, Las Vegas, Nevada.