Using M&S Representations of the Natural Environment in a C4ISR System for Decision Support

Glenn S. Iwerks
The MITRE Corporation
1820 Dolley Madison Blvd.
McLean, VA 22102
703-883-5306
iwerks@mitre.org

Abstract

Models developed for Modeling and Simulation (M&S) and other programs can be used in Command and Control (C2) decision-support systems to compute visibility and mobility of forces under current and anticipated environmental conditions. To help show this, an experimental infrastructure and architecture is created to integrate a modeling system with a C2 system in a seamless manner. The models can then be used by the C2 system to make predictions concerning the effects of the environment on mission outcomes. In this research, we build a system based on a tightly coupled architecture. Previously independent components are made interoperable using middleware. Interfaces are created to promote independence of component systems and to promote flexibility in their interactions. This paper describes the architecture, interfaces, and algorithms used in this research.

1. Introduction

Environmental modeling involves models of the natural environment populated with data that are used for environmental reasoning. *Environmental reasoning* involves models of man-made systems, such as vehicles and sensors, modeling their interactions with the natural environment. These models are used to answer questions in a Command and Control (C2) decision-support system. For example, a simple terrain line-of-sight computation is only useful if one is willing to assume the viewer can see infinitely far as long as the terrain does not intervene. In many cases this is not acceptable. There may be clouds, smoke, fog, or rain obscuring a target. It could be nighttime with only starlight present. The viewer may be using night vision goggles or infrared sensitive sensors. The vehicles may be emitting electromagnetic radiation. All these are factors in whether or not an object or point on the terrain can be seen by a viewer some distance off.

Our hypothesis is that environmental models developed for one program can be integrated into other systems for the purpose of environmental modeling and reasoning for decision-support. A Command, Control, Communications, Computer, Intelligence, Reconnaissance, and Surveillance (C4ISR) system is a C2 decision-support system (DSS). A DSS is an information system designed to simplify and accelerate the decision-making process by collecting, obtaining, providing and assessing information [Tolk and Schiefenbusch, 1999]. To help support this hypothesis, a proof-of-concept demonstration was developed to integrate existing modeling software into an example C2 system. Extensive work has been conducted by the United States Department of Defense

(DoD) M&S community in modeling of the natural environment and in modeling the effects of the natural environment on military equipment. For this experiment, environmental models developed for the M&S community are integrated with a C4ISR system and used for environmental reasoning within the system.

The use of Modeling and Simulation (M&S) technology in C4ISR systems for decision-support has been a topic of recent study [Tolk, 1999]. The M&S community has spent much time and effort on implementing models of the natural environment [Whitney et al., 1998]. The natural environment has a profound impact on the outcome of military missions, but currently, C4ISR systems are somewhat limited in their ability to reason how the natural environment will affect military systems. Recently the C2 community has recognized the need to represent the natural environment in C4ISR systems. The Integrated Weather Effects Decision Aid (IWEDA) project [McGee, 1999] is an example of an environmental tactical decision aid. As some systems have modeling capabilities, they are generally not interoperable or integrated with respect to their modeling capabilities, their ability to share ground truth data, or query results. The result is a number stovepipe of systems and duplication of effort between programs with similar requirements.

In this research we develop interfaces, algorithms and architecture for an integrated system. This proof-of-concept experiment is developed from specific systems; however, other systems may be used in their place. For this work the Common Object Request Broker Architecture (CORBA) is used to access models from a C4ISR system; however, other means are possible. Likewise, the choice of C4ISR system need not be limited to that used here. For the modeling component, any system, C2 based or M&S based, with suitably fast and accurate natural environmental modeling capabilities could be used.

In the following sections, Section 2 describes some previous work. Section 3 describes the system architecture, algorithms, and interfaces. Section 4 discusses future work, and Section 5 concludes the paper.

2. Previous Work

This section discusses IWEDA, DSS-model coupling, and the tactical decision-support system (TDSS). The TDSS is most closely related to the research described in this paper.

2.1 *IWEDA*

The Integrated Weather Effects Decision Aid (IWEDA) [McGee, 1999] is a good example of the current state-of-the-art in environmental modeling and reasoning for DoD C4ISR systems. IWEDA is fielded with the Integrated Meteorological System (IMETS) and is available from most Army Battle Command System (ABCS) components, including the Maneuver Control System (MCS). IWEDA acquires gridded weather data over a 24-hour forecast period from the IMETS server at a 10-km grid resolution. A Weather Effects Matrix (WEM) is displayed to the user, indicating environmental impact on weapons systems. The IWEDA WEM displays a grid where rows correspond to particular weapon systems and columns correspond to different times. Each

cell in the matrix is color coded to indicate the impact of weather on a weapon system at a particular time. Red indicates unfavorable, amber is marginal, and green is favorable. The user can query the system for a detailed description of the type of impacts indicated. For example, "freezing rain, greater than light, may freeze the missile to the chaparral launcher rails and reduces the overall system effectiveness". The user can also display a map overlay to show the distribution on impacts over an area of interest. The models used by the system consist of a set of rules derived from field manuals and other sources.

2.2 GIS-Model Coupling

GIS-model integration is closely related to C2 DSS-model integration. The most extensive work to date, in GIS-model integration, has been in the area of *environmental management information systems* (EMIS) [Gunther, 1998]. EMIS uses *geographic information systems* (GIS) and environmental models to support environmental management activities. Although the problem domains of command and control and environmental management are different, the fundamental issues of C2 DSS-model coupling and GIS-model coupling are the same.

The terms *loose coupling* and *tight coupling* [Fedra, 1996] are used to describe different approaches to GIS-model integration. Loose coupling is a method of integration in which common files are used as the exclusive means of data exchange between otherwise independent GIS and environmental modeling systems. Using this approach, the GIS serves as a preprocessor or postprocessor to the modeling system. The advantage of this approach is simplicity. This level of integration between a GIS and modeling software is easy to achieve. Typically all that is needed is a simple interface and perhaps some common file data format conversion routines. The disadvantage is inefficiency and a lack of versatility.

The *Spatial Decision Support System* (SDSS) shell, as presented by Djokic [Djokic, 1996], is an example of loose coupling. It is a decision-support tool consisting of a collection of components. In [Djokic, 1996], a general method of creating interfaces by linking sub-components of the system via a common command shell is presented. The shell allows the user to compose functionality of components in shell scripts, or interactively at the command line. System components consist of a GIS, an expert system (ES), and numerical models (NM). For a system (GIS, ES, or NM) to be included as a component of an SDSS shell it must meet four criteria: 1) The component should be as close to state-of-the-art as possible; 2) Each component must have an open file format for data transfer between components; 3) Each component must be controllable via standard keyboard input; and 4) Components must run in a common computer environment. For systems that store data in different file formats, a common intermediate flat ASCII file format is used. This intermediate file format serves as a "data bridge" between systems. A common "command bridge" shell command interpreter must also be developed.

The SDSS shell described in [Djokic, 1996] uses the ARC/INFO GIS, Nexpert Object expert system, and the HEC-1 rainfall runoff model as components. This particular example SDSS shell is not given a name. The purpose of the example system is to analyze watershed rain runoff. In the example, the expert system extracts parameters from the GIS and passes them to the HEC-1 numerical model. The results are then stored back into the GIS. The expert system is used to

control the interaction of the GIS and NM. The GIS acts as a preprocessor to the numerical models. The expert system replaces an expert user who would otherwise be controlling this interaction. Models are not used within the GIS component itself.

The second approach, tight coupling, involves a common user interface to system components, common data structures, and GIS-model interaction through mutually accessible procedures. The primary advantages to this approach are efficient interaction between models and GIS, and versatility of model use. The GIS and modeling components of the system exchange data structures via shared memory, network communication, and shared files. Model component functionality can be invoked within the GIS at the procedure level. A disadvantage of many tight coupling approaches is the difficulty involved in modifying and integrating system components. The CLIMEX system, presented by Fedra [Fedra, 1996], is an environmental decision-support system used for global change modeling and assessment. It incorporates natural environment models, such as agricultural and atmospheric models, into a GIS using a tight coupling approach.

2.3 Tactical Decision Support System

In [Iwerks and Samet, 1999a] the authors describe the *tactical decision support system* (TDSS). Models of the natural environment and how the environment affects vehicles and sensors, are integrated into a GIS with a tight-coupling technique using CORBA. The purpose of the system is for *tactical decision-support* (TDS). The TDSS is a decision-support system for operation in a dynamic real-time setting where the decision-maker is responsible for guiding and directing deployed personnel and equipment.

Note the distinction between a TDS system and an EMIS. An EMIS is used in support of the decision-making process to help manage and make predictions about changes in the natural environment. The user of an EMIS is primarily concerned with how man-made or natural phenomena might affect aspects of the natural environment, such as how a new bridge will impact the ecosystem around the Chesapeake Bay, or how a longer hunting season might affect the local deer population over time. TDS, on the other hand, is concerned with managing people and their equipment. In other words, how will the natural environment impact the performance of machines? For example, how will snow affect the visual range of an aircraft, or how will rain impede the mobility of a tank? The natural environment does play a significant role in TDS, but a TDS system is not an EMIS.

In [Iwerks and Samet, 1999a], a tightly coupled approach is used to integrate a GIS and environmental models. The components of the TDSS consist of a modeling component, a spatial database, and a system front end. The spatial database used is known as SAND [Esperanca and Samet, 1996]. SAND provides the fundamental storage mechanisms, access methods, data structures, and operation primitives for the system. The modeling component of the system, referred to as the *environmental model server* (EMS), was derived from JointSAF developed for the STOW'97 ACTD [Whitney et al., 1998]. JointSAF implements a set of integrated models of the natural environment, vehicles, and sensors. The third component is called the Spatial Spreadsheet [Iwerks and Samet, 1999b]. It serves as the TDSS front end. It is used to visualize changes in a dynamic spatial database. It also serves as a means to organize large amounts of

spatial data, quickly formulate queries on data, and propagate changes in the source data to query results via a spreadsheet paradigm.

The coupling of the EMS component to the rest of the system is achieved through the Common Object Request Broker Architecture (CORBA) [OMG, 1998]. In general, the main drawback to a tightly coupled architecture is the amount of work needed to adapt new environment models to an existing system. To overcome this drawback a set of interface specifications is used for information transfer between models and the rest of the system components. The idea is to make the interface general enough so new models can be incorporated into the GIS using the same interfaces as existing models. If successful, no additional work need be done to adapt new models. The only requirement is the new models must conform to the interface specification. To make the specification useful it should be independent of the model and GIS implementation languages, and independent of computer hardware architecture. The CORBA *interface definition language* (IDL) [OMG, 1998] fulfills these requirements for a specification language. The Spatial Spreadsheet acts as a CORBA client to the EMS. The advantage of this architecture is independence between the EMS and the spatial database components. This allows for different model servers to be used in different applications to fulfill different user requirements for fidelity and efficiency.

3. Predictive Battlefield Environments

The *Predictive Battlefield Environments* (PBE) research project described in this paper is an extension of the work done in [Iwerks and Samet, 1999a]. One significant difference between the two research efforts is the nature of the client to the EMS. Instead of a general application GIS, this work focuses on C2 decision-support system clients. A second difference is in the handling of time. The TDSS does not support temporal data. There was no attempt in the TDSS research to handle predicted events. The PBE research described in this paper does support future predicted environmental conditions. This introduces a temporal component to the data and how it is handled. For simplicity, the non-temporal interactions between the client and EMS will be described here first, then the element of time will be introduced.

The focus of our research is on C2-model interaction. It is assumed that software implementing an integrated set of environmental models and equipment models is available. *Tactical decision aids* (TDAs) are used to test and demonstrate the system. The TDAs used here are described to better illustrate how the models interact with the C2 system. Our focus is not on the development of new TDAs.

All together, the components of our proof-of-concept demonstration system consist of an *environmental model server* (EMS), an *environmental data server* (EDS), and the C2 system. For a source of integrated models of the natural environment and military equipment, JointSAF [Whitney et al., 1998] is used. JointSAF is a piece of modeling and simulation software used for military training and mission rehearsal. It provides a set of integrated models of the natural environment, vehicles, and sensors modeling their interactions with each other. These models are computationally light and of sufficient fidelity for training and mission rehearsal. The EDS is used to feed spatio-temporal data, consisting of gridded weather information, to the EMS. For this

research a simple EDS was constructed to deliver data through a *CORBA interface definition language* (IDL) interface. Other systems, such as the *Integrated Meteorological System* (IMETS) [McGee, 1999] server, also have potential to serve as the EDS component. An experimental *Maneuver Control System* (MCS) application called the Advanced Command and Control System [Kinkead and Roberts, 1998] serves as the client. It provides the user with a point and click interface and a map window. The user may pose queries by selecting menu items and clicking on the map. To integrate the system components, we implement a tight coupling approach using CORBA. A CORBA wrapper is created around the EMS so other processes may access its modeling functionality. Below we present some example TDAs, then explain the architecture in terms of the examples.

3.1 *Non-temporal Query*

Suppose we wish to know how visible a given location on the terrain (*target*) is to some other location (*observer*). Providing the observer and target locations to the following EMS function will give us the answer.

`float` SimplePointToPoint(*observer, target*)

This function returns a number in the range [0,1] where 1 is completely visible and 0 is not visible at all. This function is provided as a ready to use feature of the EMS. The EMS uses its integrated set of models to compute the result. For example, when JointSAF is used as the EMS it considers clouds, smoke, dust, precipitation, lighting, sensor types, and terrain to compute the result. In this case, *observer* and *target* are 2D coordinates. These coordinates are projected onto the terrain surface to find their height above the vertical datum.

Now consider the PointToPoint() function below.

`list_of_floats` PointToPoint(*observer, target, sample_distance*)

In the algorithm given below, Δ is the Euclidean distance metric between two points. Operator "|" concatenates an element to the end of a list. Function line(*p,q*) returns the line between two given point locations. The function intersects(*p,l*) returns TRUE if point *p* lies on line *l* otherwise it returns FALSE. The function floor(*n*) returns the integral component of some floating point number *n*.

The pseudo code notation is a mixture of procedural and declarative statements. For brevity, assignment of a set is expressed using first-order predicate calculus to define constraints on set members. For example, "**let** $S \leftarrow \{s::P(s)\}$" defines a set of elements such that predicate P(*s*) is true for all elements in set *S*. Ordered sets, or lists, can be expressed using the expression "**let** ordered $T \leftarrow \{t_0, t_1, \ldots, t_{n-1}::P(t_0, t_1, \ldots, t_{n-1})\}$". Set *T* maintains the ordering imposed by the constraints expressed in predicate $P(t_0, t_1, \ldots, t_{n-1})$. We can then iterate over each element in *T* with the statement "**foreach** $t \in T$ in order **do** <statements> **endfor".**

`function` PointToPoint(*observer, target, sample_distance)* : return list of floats

let ordered $S \leftarrow \{s_0, s_1, \ldots, s_{n-1} :: P(s_0, s_1, \ldots, s_{n-1})\}$
let ordered $R \leftarrow \{ f \}$
foreach $s \in S$ in order **do**
    $R \leftarrow R \mid \text{SimplePointToPoint}(observer, s)$
endfor
return R

where

$$P(s_0, s_1, \ldots, s_{n-1}) \equiv$$
$$n = \text{floor}(\Delta(observer, target) \div sample\_distance) + 1$$
$$\wedge\ (\forall\, 0 \le i < n)\ (\text{intersects}(s_i, \text{line}(observer, target)) = \text{TRUE})$$
$$\wedge\ (\forall\, 0 \le i < n\text{-}1)\ (\Delta(observer, s_i) \div (i+1) = sample\_distance))$$
$$\wedge\ s_{n-1} = target$$

The PointToPoint() algorithm iterates over a list of points S along the line from the *observer* to the *target* (see figure 1). The distance between those points, before projecting them onto the terrain surface, is given by the *sample_distance* parameter where *sample_distance* > 0. For each point $s \in S$ the algorithm computes the visibility of s from the *observer* using the SimplePointToPoint() function. A list of visibility values is computed and returned.

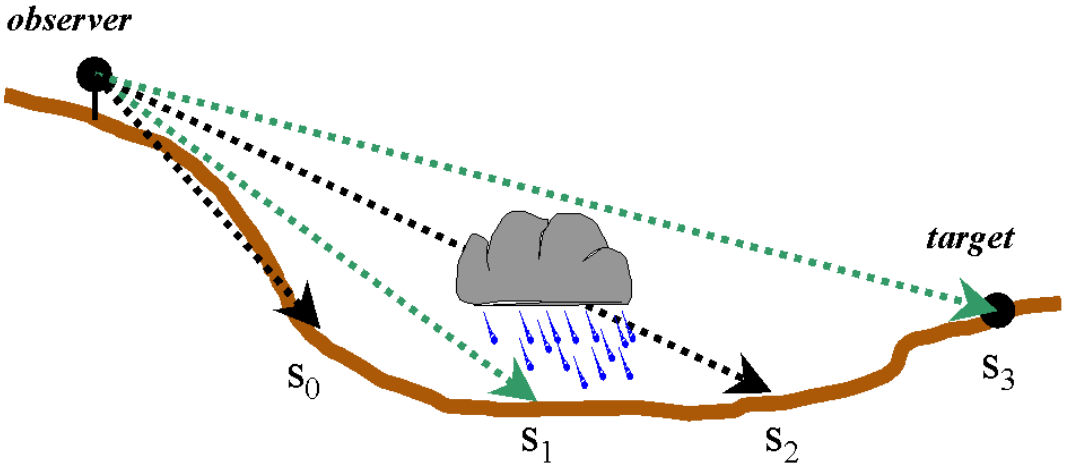

Figure 1: PointToPoint() uses SimplePointToPoint() to compute environmental effects on visibility to points along a line between the *observer* and *target*. Terrain cross section is shown.

The C2 client invokes the CORBA interface operation for PointToPoint(). It passes the necessary parameters (*observer*, *target*, and *sample_distance*) to the EMS through the interface (see figure 2). The EMS computes the query result using the PointToPoint() algorithm described above. A list of floats is returned to the C2 client as the return value of the CORBA interface call. The client then uses the returned data to render the query result in the map window. Figure 3 shows example output in a C2 client map window. The degree of shading along the line indicates

the amount of degradation. Dark shading indicates less visibility to that portion of the terrain under the line.

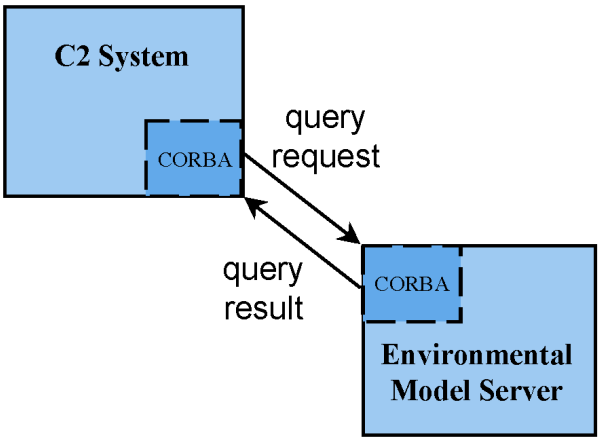

Figure 2: Simple client-server interaction.

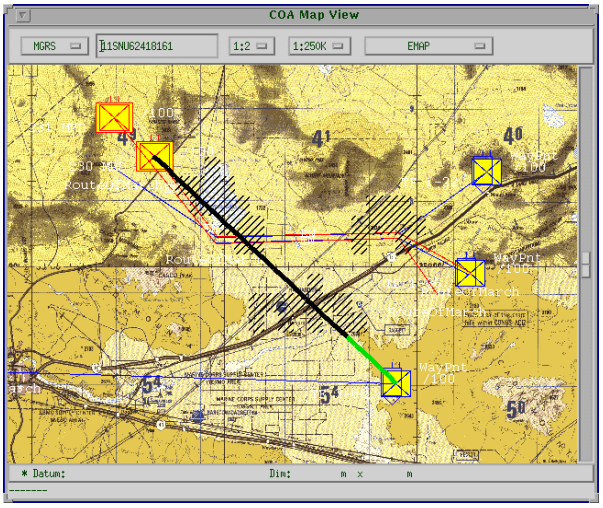

Figure 3: Map window display of Point-to-point intervisibility output. Atmospheric obscurants interfere with visibility to an enemy unit.

The point-to-point TDA is an example of a simple client-server interaction. In the next example, the temporal dimension is introduced.

3.2 *Time and Visibility*

Consider again the problem of intervisibility. Sometimes it is desirable to know how visible an area is from a given location. This is known as a *masked area plot*.

`2d_array_of_floats` MaskedAreaPlot(*observer, ll_corner, ur_corner, sample_dist_x, sample_dis_y*)

In the algorithm shown below, function coordX(*p*) returns the x-coordinate of the given point location *p*. Function coordY(*p*) returns the y-coordinate of the given point location *p*. Function length(*S*) returns the number of elements in some list *S*. Function ord(*t,T*) returns the ordinal number of a list item *t* within some list *T*. The function floor(*n*) returns the integral component of some floating point number *n*. The function location(*x,y*) returns a point location created from the given (x,y) coordinates.

As with the PointToPoint() function, the C2 client invokes a CORBA interface operation for MaskedAreaPlot(). The result is then returned from the EMS in a 2D array, which is then rendered as an overlay in the map window.

`function` MaskedAreaPlot(*observer, ll_corner, ur_corner, sample_dist_x, sample_dis_y*) : 2D array of floats

let ordered $X \leftarrow \{x_0, x_1, \ldots, x_{n-1} :: P_X(x_0, x_1, \ldots, x_{n-1})\}$
let ordered $Y \leftarrow \{y_0, y_1, \ldots, y_{m-1} :: P_Y(y_0, y_1, \ldots, y_{m-1})\}$
array R[length(X)][length(Y)]
foreach $y \in Y$ in order **do**
  foreach $x \in X$ in order **do**
    R[ord(x,X)][ord(y,Y)] $\leftarrow$ SimplePointToPoint(*observer*, location(x,y))
  endfor
endfor
return R

where

$$P_X(x_0, x_1, \ldots, x_{n-1}) \equiv$$
$$\wedge\ n = \text{floor}((\text{coordX}(ur\_corner) - \text{coordX}(ll\_corner)) \div sample\_dist\_x) + 1$$
$$\wedge\ x_0 = \text{coordX}(ll\_corner) + (sample\_dist\_x \div 2)$$
$$\wedge\ (\forall\, 0 < i < n)\ (x_{i-1} + sample\_dist\_x = x_i)$$

$$P_Y(y_0, y_1, \ldots, y_{m-1}) \equiv$$
$$\wedge\ m = \text{floor}((\text{coordY}(ur\_corner) - \text{coordY}(ll\_corner)) \div sample\_dist\_y) + 1$$
$$\wedge\ y_0 = \text{coordY}(ll\_corner) + (sample\_dist\_y \div 2)$$
$$\wedge\ (\forall\, 0 < i < m)\ (y_{i-1} + sample\_dist\_y = y_i)$$

Parameters *sample_dist_x* and *sample_dist_y* are used to determine what points to test within a rectangular region for visibility where *sample_dist_x* > 0 and *sample_dist_y* > 0 (see figure 4). The rectangular region is defined by the lower left corner, *ll_corner*, and the upper right corner, *ur_corner*. It is assumed CoordX(*ll_corner*) < CoordX(ur_*corner*) and CoordY(*ll_corner*) < CoordY(ur_*corner*). The first two lines of the algorithm compute an ordered set of coordinates over which to iterate. Points created from those coordinates fall within the given rectangular region of interest. In the third line of the algorithm, a 2D array is declared to store the result. The **foreach** loops iterate over test points by iterating over the x and y coordinates in list *X* and *Y*. In the body of the inner loop the coordinates are combined to create a test point location. The test point is passed as the *target* parameter to the SimplePointToPoint() function. The result of the function is stored as an element in array *R*. Array *R* is passed as the return value of the MaskedAreaPlot() function.

Now suppose the user wants to know how the overlay will change in the light of future predicted weather conditions. Predicted environmental conditions are stored in the *environmental data server* (EDS) as gridded weather information indexed by time. The C2 client informs the EMS that the user has changed the time of interest via a CORBA interface operation. The EMS uses the new time of interest to request data from the EDS using a CORBA interface between the EMS and the EDS (see figure 5). Once the EMS has updated its models, the C2 client may resubmit the MaskedAreaPlot() query and update its map overlay. Figure 6 shows a MaskedAreaPlot() result at time t_0. Figure 7 shows a MaskedAreaPlot() result at some future time t_1.

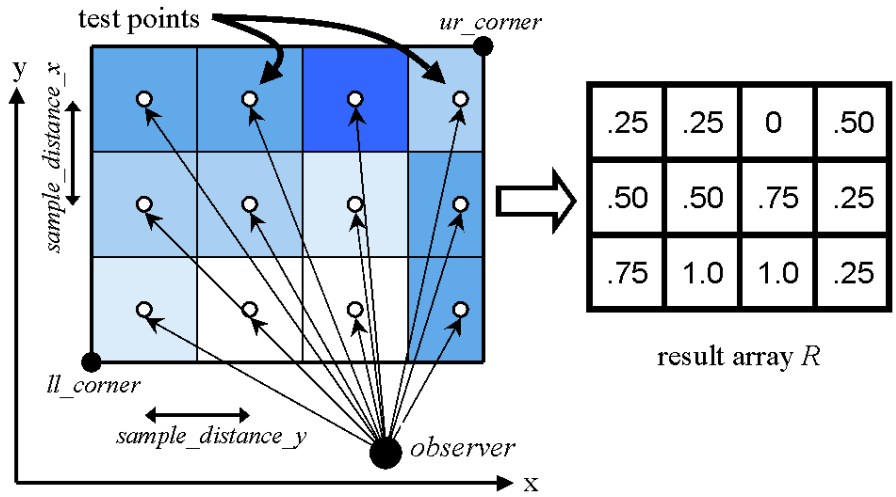

Figure 4: Points are sampled within a rectangular region for visibility from an observer.

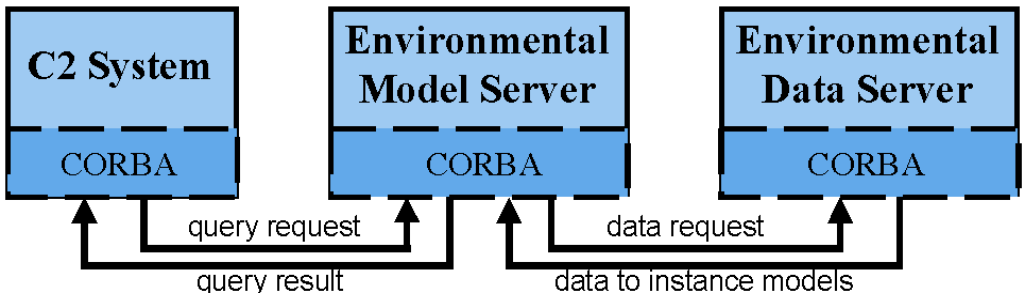

Figure 5: Architecture supporting updates to the environmental model server from data indexed by time.

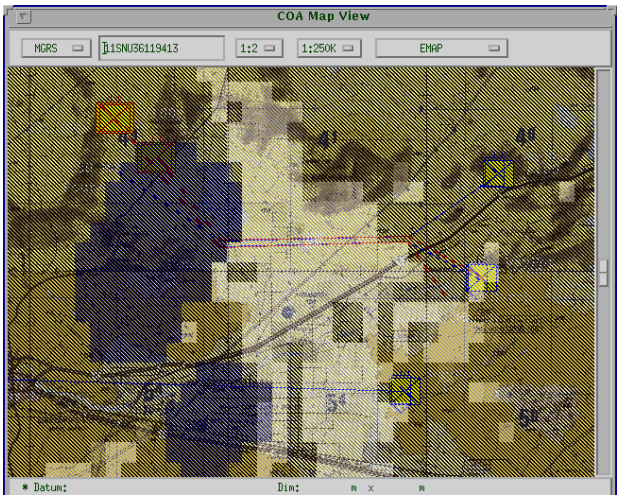

Figure 6: Masked area plot at time t_0. Observer location is at the map center. Lightly shaded areas are more visible than darker shading. Two storm clouds are shown to the left.

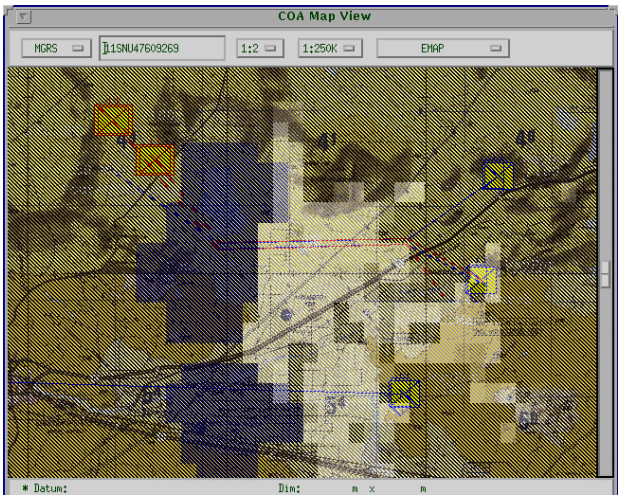

Figure 7: Masked area plot at time t_1. Observer location is at the map center. Area of visible terrain changes as storm advances.

3.3 *Time and Mobility*

To exercise mobility models in the system a mobility overlay TDA is used. The Mobility() function takes a rectangular region, sample distance and a vehicle type as input then returns an array of speed degradation values.

`2d_array_of_floats` Mobility(*vehicle_type, ll_corner, ur_corner,*
sample_dist_x, sample_dist_y)

To describe this function we will first describe a simple function to compute mobility at a single location.

`float` SimpleMobility(*vehicle_type, location*)

The SimpleMobility() function returns a number from 0 to 1. A value of 0 means the vehicle at the given location is completely impeeded. That is, the vehicle cannot move at all. A return value of 1 means the vehicle can go full speed. The SimpleMobility() function is a ready to use part of the EMS implementation. The JointSAF EMS uses soil type, slope, precipitation rate, precipitation type, hydrological features, and visibility in its calculation.

In the Mobility() function, shown below, function length(*S*) returns the number of elements in some list *S*. Function ord(*t,T*) returns the ordinal number of a list item *t* within some list *T*. The function location(*x,y*) returns a point location created from the given (x,y) coordinates.

function Mobility(*vehicle_type, ll_corner, ur_corner,*
sample_dist_x, sample_dist_y) : 2D array of floats
let ordered $X \leftarrow \{x_0, x_1, \ldots, x_{n-1} :: P_X(x_0, x_1, \ldots, x_{n-1})\}$
let ordered $Y \leftarrow \{y_0, y_1, \ldots, y_{m-1} :: P_Y(y_0, y_1, \ldots, y_{m-1})\}$
array *R*[length(*X*)][length(*Y*)]
foreach $y \in Y$ in order **do**
foreach $x \in X$ in order **do**
R[ord(*x,X*)][ord(*y,Y*)] ← SimpleMobility(*vehicle_type,* location(*x,y*))
endfor
endfor
return *R*

The input parameters *ll_corner, ur_corner, sample_dist_x sample_dist_y* are the same as before for the MaskedAreaPlot() function. The first two lines compute an ordered set of coordinates over which to iterate. Predicates $P_X(x_0, x_1, \ldots, x_{n-1})$ and $P_Y(y_0, y_1, \ldots, y_{m-1})$ are the same as defined for the MaskedAreaPlot() algorithm. In fact, the algorithm is almost identical to MaskedAreaPlot(). The differences are in the parameter list and the body of the inner loop. In the inner loop body, a test point is passed as the vehicle's location to the SimpleMobility() function. The algorithm samples mobility of the given vehicle at each test point and assigns the result to an array element. The array is then returned as the function value.

Figure 8 shows the position of a storm in a C2 client map window. Figure 9 shows the storm's effects on mobility. Only environmental effects are shown in the overlay. Terrain effects are not shown in this example overlay.

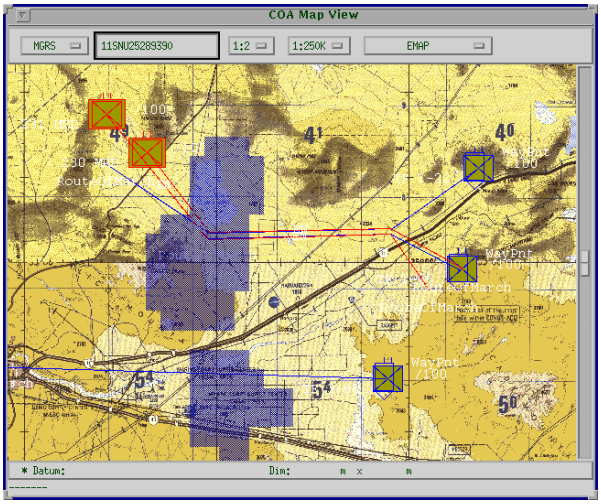

Figure 8: A storm is shown as the darkly shaded region to the left of center.

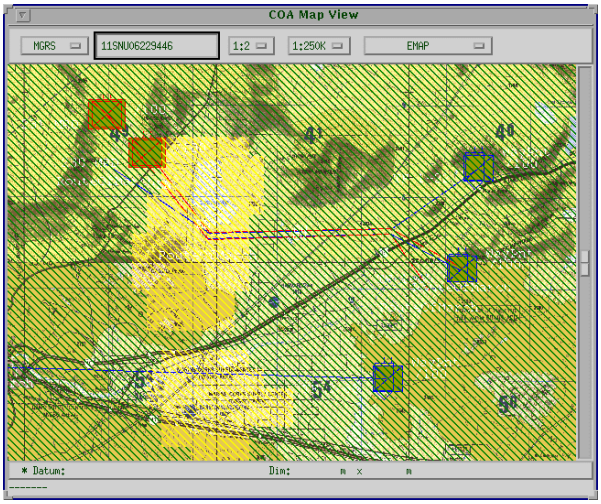

Figure 9: A mobility overlay shows slow-go areas as lightly shaded regions. Terrain effects not shown.

Now, if the user requests a time change the EMS updates itself using data from the EDS, then the C2 client can resubmit its Mobility() query in the same manner as described before for MaskedAreaPlot().

4. **Future Work**

We intend to develop an example C2 client using the Joint Mapping Tool Kit (JMTK). This is to show the architecture is not dependent on the client software used so far. Use of these techniques with a web based C2 client will also be explored. It would also be desirable to identify other modeling software packages, from either M&S of C2 programs, to use as an alternate EMS.

5.**Conclusion**

A C2 client can pose queries about the effects of the environment on mobility and visibility to the EMS. The EMS in turn receives data from the EDS to instance its models (see figure 5). The EDS stores gridded weather data indexed by time. A system clock keeps track of the current time of interest defined by the user. When the clock changes, the EDS feeds data to the EMS to update environmental models and keep them consistent with the new time of interest. The system hides the details from the user. The user need not be an expert in meteorology or have any special training in the use of models.

The approach presented here for extending C2 decision-support systems with existing M&S models of the natural environment can be used to enhance capabilities of existing C2 tactical decision aids, or for developing entirely new ones. Many models exist. It's only a matter of

integrating them. This work is a step in that direction. Our efforts to develop an infrastructure, by which models from other systems can be used to perform environmental reasoning in a C2 decision-support system, have been successful.

6. **References**

[Djokic, 1996] D. Djokic, *Toward a General-Purpose Decision-support System Using Existing Technologies*, GIS and Environmental Modeling: Progress and Research, Pages 353 – 356, GIS World Books, Fort Collins, CO, 1996.

[Esperanca and Samet, 1996] C. Esperanca and H. Samet, *Spatial database programming using SAND*, Proceedings of the Seventh International Symposium on Spatial Data Handling, volume 2, Pages A29 - A42, Delft, The Netherlands, August 1996.

[Fedra, 1996] K. Fedra, *Distributed Models and Embedded GIS*, GIS and Environmental Modeling: Progress and Research Issues, Pages 413 – 417, GIS World Books, Fort Collins, CO, 1996.

[Gunther, 1998] O. Gunther, Environmental Information Systems, Springer-Verlag, Belin Germany, 1998.

[Iwerks and Samet, 1999a] G. Iwerks and H. Samet, *Integrating the Natural Environment into a GIS for Decision Support*, Proceedings of the 7th International Symposium on Geographic Information Systems, Pages 73 – 78, 1999.

[Iwerks and Samet, 1999b] G. Iwerks and H. Samet, *The Spatial Spreadsheet*, Visual Information and Information Systems: Third International Conference, VISUAL'99, Pages 317 – 324, Amsterdam, The Netherlands, June 1999 Proceedings, Springer-Verlag,

[Kinkead and Roberts, 1998] M. Kinkead and J. Roberts, *Software Architecture for Military Planning and Battlefield Visualization*, Proceedings from the Summer Computer Simulation Conference (SCSC'98), July 1998.

[McGee, 1999] S. McGee, *The Integrated Weather Effects Decision Aid: A Common Software Tool to Assist in Command and Control Decision Making*, Proceedings from the C4ISR Cooperative Research Program Command and Control Research and Technology Symposium, United States Naval War College, Newport, RI, June 29 – June 1, 1999.

[OMG, 1998] OMG, *CORBA/IIOP 2.2 Specification*, available at www.omg.org, February 1998.

[Tolk, 1999] A. Tolk, *Requirements for Simulation Systems When Being Used as Decision Support Systems*, Paper 99F-SIW-002, Proceedings of the 1999 Fall Simulation Interoperability Workshop, University of Central Florida, Orlando, FL 32826, 1999.

[Tolk and Schiefenbusch, 1999] A. Tolk and K. Schiefenbusch, *Decision Support Systems – Concepts and Effects*, Proceedings from AORS XVIII, Special Session 2: C4ISR and Information Warfare, Fort Lee, Virginia, October 1999.

[Whitney *et al*., 1998] D. Whitney, R. Reynolds, D. Sherer, P. Dailey, M. Driscoll, M. Zettlemoyer, R. Schultz and I. Watkins, *Impacts of the Environment on Warfighter Training: STOW 97 Experiences with TAOS*, Paper 98S-SIW-224, Proceedings of the 1998 Spring Simulation Interoperability Workshop, University of Central Florida, Orlando, FL 32826, 1998.