My Simulation is from Mars; Yours is from Venus

Robert F. Richbourg, Ph.D
Institute for Defense Analyses
Alexandria, VA 22311
rrichbou@ida.org

Andy Ceranowicz, Ph.D
Alion Science and Technology
Alexandria, VA 22311
aceranowicz@alionscience.com

Robert R. Lutz
Johns Hopkins APL
Laurel, MD 20723
Robert.Lutz@jhuapl.edu

ABSTRACT

When two or more combat simulations are federated, differences in the ways they represent and reason about the world and how they communicate state changes can provide artificial advantages that lead to unfounded outcomes. Simulations are simplified representations of the real world that preserve detail and data where necessary to support their intended purpose and use abstractions where data is unavailable or in secondary areas. Such design decisions are valid when the simulations will be used as intended. However, when multiple systems, each of which have a unique purpose and supporting design, are combined in novel uses, their simplifying assumptions can overlap in ways that are not complementary and may result in invalid system interactions. Federated simulation events must minimize such occurrences to provide realistic results. However, this is much easier said than done; invalid interactions are typically caused by factors that can be deeply ingrained within each individual simulation system. Differences between these fundamental elements often only become obvious when the internals of the individual simulations are contrasted with each other. To achieve a fair fight, the simulations must be founded on compatible object models that preserve sufficient semantic equivalence between world models. The algorithms that form the basis for individual system reasoning must provide equivalent results across the interacting systems. The individual system environmental representations must be sufficiently correlated so that the potential for interaction between world objects is equivalent for all world objects. This paper considers these three fundamental characteristics of simulation systems: object models, reasoning algorithms, and environmental representations, from the perspective of the cross-system equivalence required to enable valid interactions. The general nature of the problem is defined, procedures to detect incompatibilities are developed, and strategies to prevent invalid interactions are proposed.

ABOUT THE AUTHORS

ROBERT RICHBOURG is a member of the Research Staff at the Institute for Defense Analyses. He is a retired Army officer who earned his Ph.D. in computer science in 1987. In his last active duty assignment, he was an Academy Professor and Director of the Artificial Intelligence Center at the United States Military Academy, West Point. He has been working in the area of simulation environments for the last fifteen years under sponsorship of DARPA, DMSO, M&SCO, JFCOM, STRICOM, and the NGA.

ANDY CERANOWICZ is the technical lead for federation and Joint Semi-Automated Forces (JSAF) development at JFCOM J9. He led the development of the Millennium Challenge 02 federation as well as the development of JSAF and its predecessors, ModSAF and SIMNET SAF. Andy is a Senior Science Advisor at Alion and holds a Ph.D. in Electrical Engineering from The Ohio State University.

ROBERT LUTZ is a Principal Staff Scientist at The Johns Hopkins University Applied Physics Laboratory (JHU/APL). He has over 27 years of experience in the design, implementation, and evaluation of computer modeling and simulation (M&S) systems for military customers. Since joining JHU/APL in 1992, Mr. Lutz has assumed leadership roles on a wide variety of M&S programs. Currently, he is leading several M&S standards initiatives (e.g., HLA OMT, DSEEP, SRML) within the Simulation Interoperability Standards Organization (SISO), supports the Live-Virtual-Constructive Architecture Roadmap (LVCAR) effort as both a Study and Expert Team member, and serves as the Navy representative to the Modeling and Simulation Activity center (MSAC) in support of the UAS Airspace Integration Joint IPT. He also serves as a guest lecturer in The Johns Hopkins University Whiting School of Engineering.

My Simulation is from Mars; Yours is from Venus

Robert F. Richbourg, Ph.D
Institute for Defense Analyses
Alexandria, VA 22311
rrichbou@ida.org

Andy Ceranowicz, Ph.D
Alion Science and Technology
Alexandria, VA 22311
aceranowicz@alionscience.com

Robert R. Lutz
Johns Hopkins APL
Laurel, MD 20723
Robert.Lutz@jhuapl.edu

INTRODUCTION

Large simulation federations, composed of multiple live, virtual, and constructive systems are important experimentation and training resources. Federating provides a unique mechanism that can combine disparate systems to create wide-spectrum environments that are useful in many application domains. But, what does it really mean when these different kinds of systems have been "composed" into a set of systems that can interact with each other during a simulation event? Typically, the kinds of interactions that are allowed to occur are constrained to the subset thought to produce valid results (and which often, unintentionally, may still include some that lead to invalid results as well). Experienced event designers understand that, even after a complete systems integration, the semantics of arbitrary system interactions may not be the same on the sender and receiver ends. As an example, we will use a federation composed of the "Mars" and "Venus" simulations. Consider when an entity in the Mars simulation is within the engagement range of another entity in the Venus simulation.

Several factors influence the potential for a Martian - Venusian engagement and its possible outcomes. First, the two entities have to be aware of each other. Once aware, each entity has to recognize the other as an adversary that should be engaged and that engagement is possible given the effective range of available weapons. Since the Martians and Venusians are the original Hatfields and McCoys, an engagement follows, requiring each entity to repeat the cycle of calculating their own damage and understanding the other's damage, until each determines that further engagement is no longer possible or necessary. Each step in this engagement process has some potential for system-to-system inconsistencies leading to an invalid engagement result.

Typically, the two entities become aware of each other when each applies internal line-of-sight (LOS) algorithms to their own internal representations of the physical environment. Different LOS algorithms applied to the same database can provide different results. The same LOS algorithm applied to different databases can provide different results. But the likely situation is that different LOS algorithms are applied to different environmental databases, clearly a case where inconsistent results are possible. Thus, even in cases where LOS should be reflexive, the Martian might see the Venusian while the Venusian is unaware of the Martian. Thus, the Martian entity has been provided with an unfounded "Stealth" capability, based on algorithmic differences, database differences, or both. This will provide an unfair advantage to the Martian and leads to an invalid engagement result. Further, the process of determining the need for engagement (a recognized adversary that is mission capable) requires each entity to correctly interpret the "entity state" information provided by the other. Here, object model agreements and interpretations have to be consistent if invalid results are to be avoided. (What does the DIS entity state "Slightly Damaged" mean to an HLA entity when determining the need for continued engagement?)

The example shows that invalid interactions can be caused by deeply ingrained factors within each individual simulation system. Differences between these foundational elements only become obvious when the internals of the individual simulations are contrasted with each other. To achieve interoperability, the simulations must be founded on compatible object models that preserve sufficient semantic equivalence between world models. The algorithms that form the basis for individual system reasoning must provide equivalent results across the interacting systems. The individual system environmental representations must be sufficiently correlated so that the potential for interaction between world objects is equivalent for all world objects, irrespective of the system simulating them. The following sections discuss these three fundamental characteristics of simulation systems: object models, environmental representations, and reasoning algorithms, from the perspective of the cross-system equivalence required to enable valid interactions. In some cases, procedures to detect incompatibilities are developed and strategies to prevent invalid interactions are proposed.

OBJECT MODELING

Incompatibilities among the object models of federation participants can be a fundamental source of both syntactic and semantic interoperability problems. Such incompatibilities can be felt at any of three levels. First, there are the object modeling issues that are introduced whenever different simulation architectures (e.g., DIS and HLA) are mixed within the same simulation environment. Such issues are relatively easy to detect, and are based on differences in the way different architectures define their object modeling concepts and supporting constructs. Aligning the names and structural relationships among all object model elements is the next level of object modeling issues. As an example, there may be an object that is being modeled by two or more federates using semantically equivalent internal representations, but having different names in their external interfaces (e.g., aircraft, air vehicle). Such issues are also relatively easy to detect, although verifying semantic equivalence can be rather difficult. The final level of object modeling issues is essentially the opposite of the previous issue. Here, there is a direct correspondence between the names and structure of elements in the object model, but the representations of these elements by the various federates are semantically inconsistent. These issues are the hardest to detect, and can have serious consequences for the overall validity of the simulation environment. The following sections detail each of these three levels individually, and identify potential strategies for resolving the inherent compatibility issues.

Mixed Architecture Object Modeling Issues

There are many different simulation architectures in use today. The architectures that currently dominate the Department of Defense (DoD) Modeling and Simulation (M&S) user community include the:

- Distributed Interactive Simulation (DIS)
- High Level Architecture (HLA)
- Test and Training Enabling Architecture (TENA)

In the DIS standard (IEEE 1278), there is no separate object model construct defined in the architecture. Rather, the DIS specifications define a set of data messages, called Protocol Data Units (PDU), that provide information concerning simulated entity states and the types of entity interactions that can take place in a DIS exercise. The collective set of PDUs defined in the IEEE 1278 standard, along with the format and syntax of the various data structures, are generally considered as the "DIS object model" in the sense that together they define the shared attribute data of objects and interactions and the way this data is exchanged at runtime.

In the HLA, (HLA v1.3 or IEEE 1516), object models define an agreement for runtime data exchange in a federation. Unlike the concept of a software object as commonly understood in the Object-Oriented (OO) software development community, HLA objects are encapsulations of state data for software objects that are being modeled within the federates. Runtime interplay among federates takes place through updates to such data (referred to as "attributes") and exchange of non-persistent action/event notifications (called "HLA interactions"). The specification of all such objects and interactions, with associated attributes and parameters, collectively defines the object model. A component of the HLA (IEEE 1516.2) provides a template for documenting object models.

In TENA, object models map much more closely to traditional software object models. TENA objects, referred to as "Stateful Distributed Objects" in the TENA specification, support the traditional features of OO programming, such as inheritance, composition, and remote and local methods. A standard object model is provided in the TENA specification, from which application-specific object models (referred to as "Logical Range Object Models") can be derived.

Gateways are the most common mechanism for reconciling object models across different architectures. Gateways are general purpose, non-simulation-specific software applications that provide a variety of translation services across dissimilar simulations. While some gateways are designed to support only a single, specific object model translation (e.g., Real-time Platform Reference Federation Object Model [RPR FOM] for DIS-HLA applications), others provide more "FOM agile" features for mapping between other object model representations.

Although gateways can be highly effective in reconciling object model representations across the different architectures, gateways can add complexity to the distributed simulation architecture, can be an additional source of error, and can increase latency across the simulation environment. For those reasons, the user communities associated with the various architectures have recently shown interest in reaching more generalized agreements on object model content. These agreements would reduce current dependence on gateways and reduce the time and effort necessary to implement future distributed simulation environments. Once such agreements are established, cross-community object model content agreements should

eventually lead to more formalized standards activities, which will provide an important means for maintaining the stated agreements in the long-term.

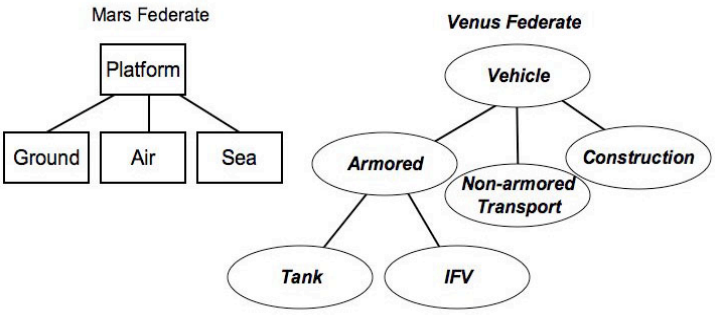

Figure 1. Federate External Interfaces

Aligning Names/Structure of Object Model Elements

Even within a single architecture community, there can be many different object model representations that must be reconciled when new distributed applications are built. This is particularly true for HLA applications. The separation of architecture from data is fundamental to the HLA paradigm. While this design feature provides considerable flexibility in the way object models are developed, such flexibility can (and has) resulted in a proliferation of object models across the various user domains. The impact of this situation is that some degree of object model reconciliation is required for most new HLA federations. Although static data/object model structures are formally defined in both the DIS and TENA specifications, both possess mechanisms to extend existing structures or even add whole new elements, and thus are subject to the same basic compatibility issues.

Some of the inherent problems commonly associated with object model reconciliation are introduced in Figure 1. Here, we show (partial) external interfaces of two different simulations. Mars is an aggregate-level multi-service application that decomposes a "Platform" base class into three basic subclasses; those of "Ground", "Air", and "Sea" entities. New objects are instantiated only at this second level, and are given the applicable characteristics (e.g., fighter jet, cruiser) via a defined initialization dataset. Venus is a higher-fidelity application (for Army vehicles) that also instantiates new objects only at the leaf nodes.

Suppose that these two simulations are both to be included in an HLA federation, and that the external interface of each federate is a Simulation Object Model (SOM). The issues associated with direct reconciliation of the two SOMs are rather obvious, as each structure has different class names and class relationships, but it is not as obvious if and where the various classes may be semantically different. The reconciliation of class names and structures across the various federates is normally performed as part of the Federation Object Model (FOM) development process. This involves developing a common object model representation for the full federation, and then having each federate map their native interface to the agreed upon FOM structure.

Since subscription needs tend to drive FOM class structures, assume that Mars wishes to subscribe to information about "Tank" and "IFV" (Infantry Fighting Vehicle) objects published by Venus, and Venus wishes to subscribe to "Ground" platforms published by Mars. Figure 2 illustrates a possible FOM based on these requirements. In this solution, platforms can be instantiated as either generic ground vehicles (by Mars) or as a "Tank"/"IFV" (by Venus). Thus, objects may now be instantiated at two different levels of the class structure. To implement this solution, each federate must apply the appropriate mapping between corresponding objects in the FOM and SOM. This translation may require the modification or addition of new attributes in the FOM, such as the inclusion of a "Type" attribute in the "Ground" class so Venus will know if new class instantiations should be considered a "Tank" or "IFV".

This is a rather simple example. In practice, the proper alignment of object names and structures can become quite difficult for large federations. While there are tools available to assist with such activities (e.g., FOM mapping tool in Mak's VR-Link software), there is no real substitute for the long and sometimes difficult negotiations that must occur at FOM development meetings. However, there are opportunities for improved object modeling techniques that may make such efforts less resource intensive in the future. As an example, since modern software development methodologies are frequently designed around

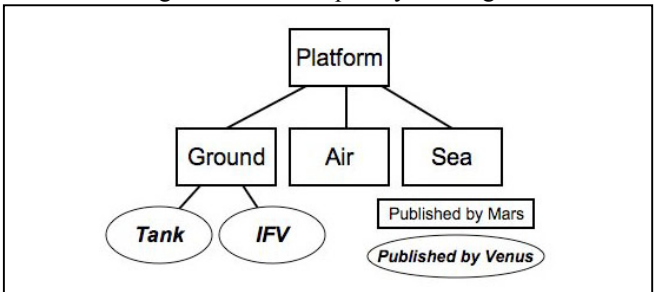

Figure 2. Reconciled Federation Object Model (FOM)

compositional approaches (i.e., building new software systems from small, reusable components), it seems reasonable that the same proven practices can be effectively applied to object modeling as well. The Joint Composable Object Model (JCOM) project at Joint Forces Command (JFCOM) is already implementing this concept. The current standard in this area is the Base Object Model (BOM) standard sponsored by the Simulation Interoperability Standards Organization (SISO). The BOM standard provides a template for describing object model components, along with a guidance document that describes how BOMs are developed and assembled into arbitrarily large object models. Although the infrastructure for more widespread application of the BOM concept is still relatively immature, some supporting tools do exist (e.g., Simvention's BOMWorks).

Semantic Compatibility Issues

Successfully translating between different object model representations is necessary but not sufficient to assure full object model compatibility. Even if all local simulation interfaces can be mapped to a common federation-wide representation, it does not guarantee that all runtime data will be interpreted by federates in a consistent fashion. This is a major source of "fair fight" issues in distributed simulation applications. As an example, Mars and Venus may both recognize an "Aircraft" class and subscribe to its "Altitude" attribute, but if Mars expects the data in feet while the Venus expects the data in meters, the simulations will perceive the aircraft altitude differently. Such issues can also influence characteristics such as accuracy, update rate, or in some cases, the actual meaning of the data item. Sometimes, these inconsistencies will simply produce a small bias, favoring one simulation. In other cases, the overall validity of the federation execution may be compromised.

Unfortunately, these "deeper" interoperability issues are frequently both harder to detect and harder to resolve. Some architectures provide automated utilities to reduce the chances for inaccurate interpretation of runtime data, such as the standard method implementations provided by TENA (e.g., reference frame conversions). However, such capabilities do not eliminate the need for structured Verification, Validation, and Accreditation (VV&A) processes to overlay federation development. Careful attention to semantic equivalence in federation object model development is essential to avoid fair fight issues. Although such activities can be resource intensive, the impact of false conclusions drawn from invalid federation executions is much a higher price to pay.

ENVIRONMENTAL REPRESENTATION

Typically, whenever two or more systems are prepared to interoperate in a single simulation event, one of the first requirements is correlation of the environmental databases those systems rely on. But what does it mean when the event manager states that the environmental databases have been correlated? Often, the implication is simply that an entity placed on the Martian landscape at a specific coordinate appears at an acceptably equivalent location when observed from the Venus system. This is certainly the minimum requirement, but such "correlation" does little to ensure the validity of the subsequent system-to-system interactions.

Different types of environmental databases (atmospheric, open water, acoustic, etc.) support different types of operations. In this discussion, we focus on the land domain as a representative case. Aside from supporting visual presentation, the most important services supported by the environmental representation are line-of-sight (LOS) and mobility calculations. To guarantee valid interactions, these calculations must produce the same results for the same mobility or LOS analyses irrespective of the database supporting the calculations; the databases should be functionally equivalent. However, such equivalence may be harder to achieve than is commonly realized.

From the LOS point of view, we claim functional equivalence can only occur when the elevation surfaces of the databases are identical. To see this, consider the terrain cross section illustrated in Figure 3. Wherever different databases have different elevation surfaces at corresponding (x,y) locations, it is possible to position an observer so that different fields of view occur. In Figure 3, the dashed and solid lines depict two different elevation surfaces. The shaded regions show areas that are visible based on one surface and blocked according to the other. While this is a trivial case, a single example establishes the validity of our claim. It also provides an illustration of the situation that should be avoided. Positioning a second opposing-force entity in

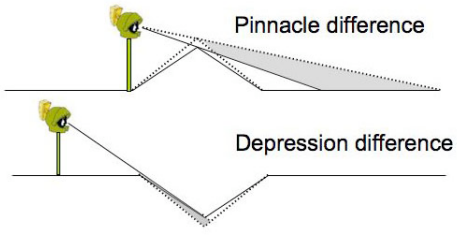

Figure 3. Different Elevation Surfaces Yield Different LOS Results

the gray shaded area of Figure 3 allows the potential for an invalid interaction because one database would indicate that LOS existed and the opposing entities would be aware of each other, so an engagement could occur. However, obstructions in the other database prevent the entities from observing each other, precluding engagement. The Martian can acquire and engage the Venusian without ever being detected.

The fact that databases having different elevation surfaces can lead to invalid interactions does not mean that entities using those databases should be prevented from interacting at all. Figure 3 also illustrates that there are a great many pairs of locations where consistent LOS results would result, despite the database differences. That is, positioning one entity at the illustrated observer location and locating the second entity anywhere that is not shaded will result in consistent LOS analyses using the different databases. In fact, there is a bounded set of locations for the second entity that would lead to different LOS solutions while the set that gives equivalent results is unbounded (assuming entity height is not bounded). Thus, in the case illustrated, the probability of an invalid interaction based on different LOS results is comparatively low. So, ruling out any interactions between entities using these databases is inappropriate. A better strategy is to actively manage the interaction opportunities in a way that maximizes the potential for valid interaction.

Established management strategies are designed to preclude situations where interacting simulation systems could develop inconsistent world views. One strategy separates incompatible simulations in time or space (or both) so that, although the simulations all operate within the same federation, they do not directly interact with each other during any specific simulation event. A second management method allows different systems to interact, but requires that one of the interacting systems adjudicate all decisions about each interaction. Using the above entity-to-entity engagement example, one simulation would determine the existence of LOS and provide the other simulation with its result, thus assuming the role of "line-of-sight server" and ensuring that both simulations reach the same conclusion. (Note that another similar option would place one federate in the role of "terrain server".) Finally, another strategy requires that all interacting systems use the same terrain. Different run-time formats restrict the applicability of this strategy and, even when it is viable, the systems must all use algorithms that provide the same results as well.

There are also other problems with these strategies. The first approach prohibits interaction of systems, so the overall simulation event would more appropriately be described as a collection of separate systems, not as an integrated federation. The second approach is more acceptable, but clearly has shortcomings. Client systems may well be stripped of their essential character when their decision-making authority is removed. Further, server systems might incur greatly increased computational load and communicating their decisions would increase network load and add latency into the federation. Finally, none of these solutions are fully applicable when live systems are involved.

We can build on results from earlier work (Richbourg, 2001) describing alternative methods that do not suffer these limitations. While there are many causes for conflicting LOS results between environmental databases (different source data, different transformations to create the run-time databases, errors in one database, different features or feature placement, use of incompatible algorithms, etc.), a comparative analysis focused on the final databases and the calculations they support can provide the information necessary to manage interactions. Applying context-sensitive viewshed analyses (Ray, 1994) to the databases and comparing the results can identify areas that provide the best potential for valid interactions.

Consider the case if Mars used Level 1 Digital Terrain Elevation Data (DTED L1) and Venus used a Level 2 DTED (DTED L2). Figure 4 is a shaded relief depiction of the Level 1 data (which includes elevation posts at a horizontal spacing of approximately 83 meters). Figure 5 is a shaded relief depiction using Level 2 DTED (approximately 28 meters horizontal post spacing) for exactly the same area. Even at this zoom level (approximately 30 KM square area is shown), the DTED 2 shows a crisper representation of the terrain.

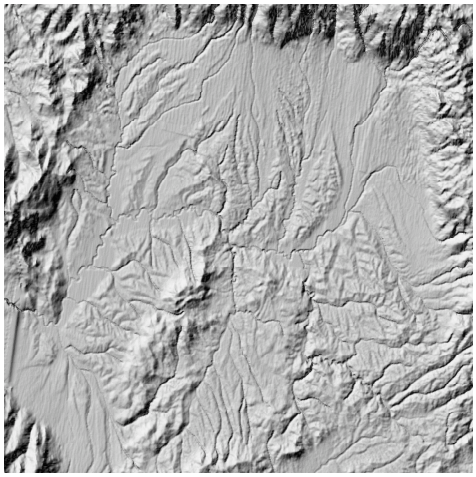

Figure 4. DTED Level 1 Shaded Relief

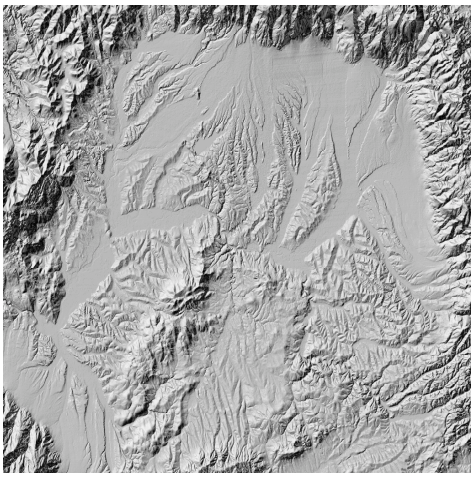

Figure 5. DTED Level 2 Shaded Relief

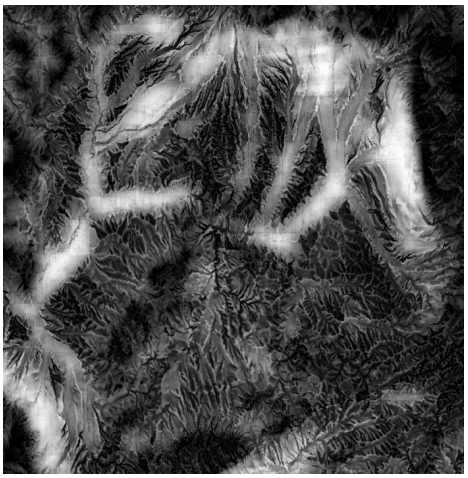

Figure 7. DTED 2 – Tank-to-Tank Viewshed

Figures 6 and 7 show the viewsheds for these representations. Both viewsheds are derived from tank – to - tank LOS approximations for all locations in the 30KM square area. Each elevation post (pixel) in the viewshed is scored according to an estimate of the number of other elevation posts, within engagement range, where LOS would exist between two tank entities. (We applied bi-level interpolation to the lower-resolution DTED L1 so that both databases have the same scoring opportunities.) These scores have been mapped into the gray scale for illustration. Light areas indicate locations where an observer (tank platform) has relatively open LOS and fields of fire regarding targets (other tank platforms) within the effective range of their main weapon. Dark areas correspond to areas of poor LOS for that type of observer (but relatively good concealment potential). Comparing the viewshed information allows identification of those areas where LOS analyses applied to either the DTED L1 or DTED L2 elevation data are likely to produce similar results and therefore valid interactions between Martian and Venusian tanks.

Figure 8 illustrates the correlation of the two viewsheds. Areas that appear white correlate to at least 90%. This correlation map provides a tool for event managers to use in event planning. For example, one could plan force interactions in locations that correlated to at least 90%, accepting some risk (e.g., 10% or less) of invalid interaction. However, note that these types of viewshed comparisons are closely tied to the entity types that are anticipated to engage one another. The illustrations in Figures 6, 7, and 8 all assume that target and observer are located about 3 meters above the ground surface and that they are separated by 3,000 meters or less. Creating the same type of analyses for other potential entity-to-entity engagements would require both height AGL and range parameters appropriate for those entities. That is, a viewshed

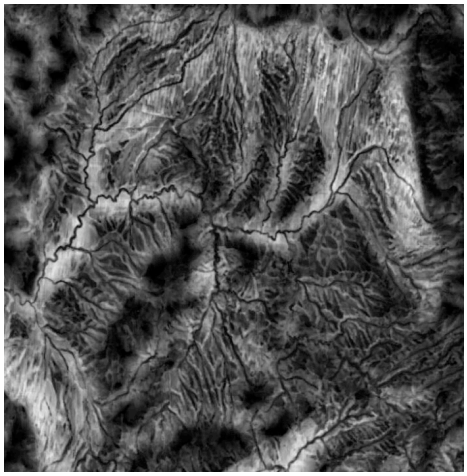

Figure 6. DTED 1- Tank-to-Tank Viewshed

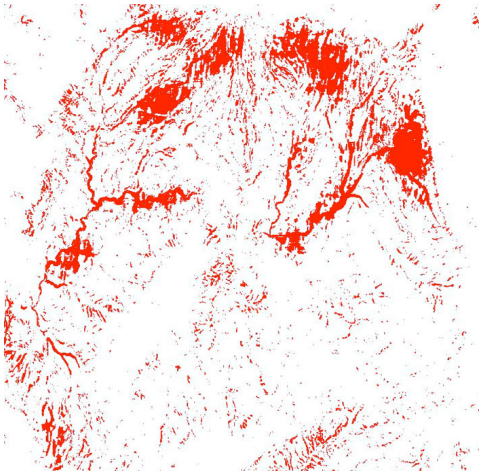

Figure 8. Viewshed Correlation at 90%

analysis has to be performed in the context of likely system interactions.

There are other aspects necessary to fully correlate environmental databases and ensure valid system-to-system interactions. Here we have focused on terrain data to provide an example of the type of analyses that are necessary. Even though elevation data analyses are less important when working with open water or atmospheric data, the main point should still hold. Correlation of databases to ensure valid interactions must be performed in the context of the functions that will apply during potential engagements. While one could directly compare the individual data elements of each database (elevation post points in this terrain-centric example), such a comparison is completely context-free and is thus not very informative from an exercise management point of view. As an example, translating root-mean squared error (RMSE, derived from directly comparing surfaces) into something indicative of the potential for a fair fight seems very unlikely and perhaps impossible. Simply characterizing differences in the surface elevation does not provide the type of information necessary to make appropriate decisions. Further, we assert without proof that there are cases where very large differences in surface elevation have little impact on LOS results and there are some cases where small differences can have a large impact on calculated LOS. Conversely, the function-based, context-sensitive method described here allows event planners to make informed decisions about risk of invalid interactions in their events.

ALGORITHMIC INTEROPERABILITY

Even if you have managed to agree on and express the same shared objects and interactions and you use identical supporting data for them and their environments, there is still no guarantee that the federation will be a valid simulation. The "whole is greater than the sum of its parts" cuts both ways. Even if the individual parts are validated, their sum does not necessarily yield valid federation. Since V&V for a single simulation is fairly difficult, it is not good news that additional V&V is required for the federation.

To see why simulation validity is so fragile, we need to examine what a model is as we hold that a computer simulation is a collection of executable models engineered to work together. The concept of a model is at best fuzzy; what is the difference between a model and a theory for instance [Morgan]. But, for the purpose of this discussion please accept the following definition: a model is a collection of abstractions of the world called variables together with a mechanism to predict the values of some of the variables based on the values of the rest. This is very close to the definition of a mathematical function, with the added restriction that the model includes principled mechanisms to obtain the values of all the variables from the real world. Thus a model is a function that predicts the state of one part of the world from that of another part and that prediction can be checked by comparing it with values obtained directly from the world. The prediction is often self referential where future values of a variable are a function of past values, but that fits the definition since each point in time can be considered a separate variable. Consider a familiar example, the plastic models of airplanes available at hobby shops. Given a coordinate relative to a point on the nose of the real aircraft, the model can predict whether that point is inside or outside the real aircraft.

So why do we build models? To capture aspects of the world in simplified forms that make it easier for us to manipulate them and predict what would happen if we manipulated the real thing [Woodward]. For example, if we were planning to buy a small hanger to house our real airplane and we wanted make sure the plane would fit in the hanger, we could take our model and try to place it in a same-scale model of the hanger; essentially performing a simulation. In fact, for that purpose we could use far simpler models of the airplane and hanger; ones consisting only length, width, and height. However if we wanted to see what the air flow over the plane's wings looked like, the simpler models would be of little help. But building a scale model requires a great deal of measurement while the length, width, and height model requires only three. Even worse, the more complicated and complete a model is, the harder it is to figure out what manipulations will achieve our goals. Models are simplifications of reality and they get their power from the fact that they are easier to manipulate than the real thing but are still able to predict how the real thing will respond to a certain set of manipulations. So the ideal model will be just complicated enough to predict the results of the manipulations we are interested in and no more. Thus any object or process can be represented by an unlimited number of models representing different properties at different resolutions for different purposes.

Another factor in the huge variability between computer models is the fact that they are mathematical models. Physical models automatically bring with them various properties of the world. Mass, drag, gravity, and the reflection and absorption of electromagnetic energy all come with the physical model; perhaps not in the right proportions, but for free. With mathematical models on the other hand, nothing comes for free. The mathematical model

salesman gives the used car dealer a good name. "You didn't say you wanted tires on that model car, that's an expensive contract extension." Thus computer models differ not only because of purpose but because everything that is represented has to be explicitly added (we will ignore emergence as, unless it is very carefully cultivated, it is more likely to be weed than a flower). As there are an infinite number of ways to abstract, having different model builders abstract the same way is unlikely.

Now back to simulations; simulations are complete in the sense that they contain all the models required to represent both the objects and processes of interest and the aspects of the environment that we want to examine them interacting with. They would be useless otherwise. So whenever we bring together two simulations they usually have different models of the same phenomena. If the phenomena are objects such as an F16, we resolve the problem by saying, "OK, the Venus simulation will model all the F16s and Mars will model F15s." But because we and our interoperability architectures are object focused, when it comes to processes we don't do the same thing. Each simulation implements its own model of the processes of target detection, damage calculation, driving, collision avoidance, etcetera. So, when we run our federation we will have different models for the same processes running simultaneously. Essentially the F16 is flying over Venus and the F15 is flying over Mars and through the very clever trick of the RTI they think they are both on the same planet and they can interoperate. But their gravity and atmospheres are very different and so the F16 in Venus doesn't quite behave the way an F16 in Mars would and the F15 may be confused by this behavior difference. But wait you say, "Can't I change the Martian data files to make them close enough to the Venus data and solve the problem?" Well maybe, but unless the data files have exactly the same variables and use them in exactly the same way, i.e., have the same algorithms, which is unlikely, there will always be some difference. Venus may calculate LOS from the center of mass of a flight of aircraft while Mars may calculate LOS from each of its sensors and crew. Venus may use a ray tracing routine to calculate intervisibility while Mars uses just range. It is important to note that each algorithm choice has its pluses and minuses. It is not just a matter of selecting the model with the most detail; otherwise we would only test in the real world since it is by definition its own best model. Venus may have a much more detailed detection algorithm but it can only handle a scenario where it has no more than three opponents. It won't be much good for a Red Flag scenario, but it is certainly more accurate in a one - on - one scenario.

So how do we detect these problems and correct them? In general there are no easy answers. In a single simulation there is a long engineering and testing process to harmonize the models and make them work together. The uninitiated may get the false impression that federations allow us to cheat that process and get a free lunch. But the experienced practitioner assumes that models from different simulations will be incompatible, guilty until proved otherwise. However, simulations often contain hundreds of process models so exhaustive testing is not practical and resolving identified problems is not easy. Federation builders have developed some techniques to deal with the problems:

- The first is to limit testing to the scenario or class of scenarios that we intend to run and our purpose; the more specific the better.
- We divide up the simulation of objects so that objects that operate in similar environments are all simulated together. Aircraft in one simulation, surface vessels in another. That way, similar entities have the same environment.
- We use persistent federations, where a history of testing and interoperability modifications gives us confidence that the models are compatible, at least for previous purposes.
- We limit the scenario to regimes where the simulation models give consistent results. If one simulation doesn't model night operations well, don't run a night scenario.
- We dumb down the over achievers. The nail that sticks out gets hammered down. Raising the least common denominator is usually much harder.
- We provide common process models for all the simulations to use. This is the equivalent of what we do for objects. The problem is generally latency, so the TENA approach of allowing external methods to be compiled into each federate has a lot of promise here.
- We add noise into the models or handicap some in such a way as to level the playing field.

In the long run, the best solution is to teach models to our simulation engineers. We need to classify and name different process models so we can talk about them at a higher level. There should be taxonomy of standard models and their variations. The vocabulary of the modeling and simulation engineer needs to include the Acquire model [Friedman] much more than it needs to include JSAF, JCATS, F16, DIS, or HLA. That way, when we bring our simulations together to federate, we can talk at a much more meaningful level than we do now and communicate much more effectively and efficiently.

The second benefit of developing the simulation model ontology called for above is that it is needed to make the automated integration of models possible. The service oriented architecture (SOA) strategy calls for the dynamic integration of services via a web architecture. SOA is a development of the business world [Carter]. Under the SOA strategy, an organization decomposes its IT architecture into business processes and services. Services are reusable information processing tasks such as charging a credit card or reordering a stock item. The key innovation is that, the services are not hardwired into the application. Instead, services are catalogued and applications discover them in the catalogue each time they go to call them. This decoupling makes it much easier to reuse services and swap out old implementations for new versions. This can be applied to simulation by making our process models into services and our simulations, like a business process, would primarily consist of a top-level loop and specifications of the types of models that are required to run the simulation. Upon starting up, the simulation would search online catalogs to find models that it needed to execute. In business scenarios you generally call the credit card authorization service once for each transaction. In simulation, we know we are going to call each model repeatedly at high frequency. So it doesn't make much sense to rediscover the model each time. The discover process should only be conducted at the start of each event rather than repeatedly during the execution. Latency will probably be a critical issue for many models, so it may be necessary to dynamically download the models before execution. However, for this approach to solve our interoperability problems we need to end up using the same process models. It is not enough to automatically discover F16, F15, and missile models. If these models don't share a common environment, common sensor physics, and common target detection and acquisition models (i.e., the underlying world processes and data), then they will have the same interoperability problems as our current federations without the people in the loop to solve them. Thus the federation has to choose the process model services and tell the simulations and their object models to use them. The key to allowing that to happen is a common ontology of process models that allows us to write the specifications needed to identify the models required.

CONCLUSION

Ensuring that arbitrary simulations will interact in valid ways is difficult because there are so many moving parts that have to be aligned properly. Object models define the lexicon for interactions and alignment of different object models, as with any lexicon, requires resolving both syntactic and semantic differences. We have processes to attack the syntactic differences, although the existing methods are often mostly manual. These can be labor-intensive, tedious undertakings that typically require several iterations to complete. The semantic issues are far more difficult to resolve because we do not have processes that are guaranteed to: 1) identify all the differences and; 2) provide any help in modifications that can resolve them. The difference here is that the syntactic issues stem from visible constructs while the semantic problems are usually based on internal processes that are difficult to isolate or expose. Probably, the best solution here is to develop and utilize component-level object models that can be shared and used across the community. In some respects, this is the object model corollary to the service-oriented approach; the idea is to have everyone use the same object model "service."

Like the object model semantic issues, the related environmental issues are also difficult to resolve. The causes behind potential invalid interactions are only apparent when placed in the correct context of use. The same representational difference can lead to invalid results for one type of interaction and have no adverse impact on others. Even a small difference in elevation surfaces, as an example, can lead to inconsistent line of sight analyses that, in turn, can lead directly to invalid outcomes derived from unfair advantage. However, the unfair advantage is fundamentally dependent on the exact context of use, and may not be a factor in many outcomes. As a result, even exhaustive and detailed comparisons of different environmental representations can be fruitless. Thus, the notion of having "correlated databases" between interacting federates is likely meaningless, unless the correlation was synonymous with having ensured equality of the surfaces. A more meaningful characterization of the potential for valid interactions is based on functional comparison. These kinds of characterizations, where comparison is functional and within context, can be used as management tools that allow event designers to construct scenarios that the different environmental representations will support or, at least, to accept a bounded risk that the scenario will result in valid outcomes.

Managing potential interactions becomes far less important if all of the interacting systems are using the same environmental database, as might be the case if a terrain server could be used. This is an attractive solution, but is also rarely a viable approach. Run time environmental databases are typically heavily optimized for specific systems and are thus very difficult to share across systems without some form of translation or reformatting. Preserving functional equivalence throughout the translation process is also

problematic. Given that most simulation systems start with the same (or nearly so) source data for their environmental data, the problem of maintaining functional equivalence during translation is at least partly responsible for the current situation. Also, the environmental data is usually quite voluminous, making it difficult to serve over a network. These kinds of problems usually imply that the server approach is not as viable as a management solution. Informed management, as we suggest above, can be used to preserve interaction validity.

A service-oriented approach may be more attractive in solving problems of model inconsistencies. After all, if every system used the same model (algorithm) then there would be little chance for inconsistency. This approach fundamentally depends on the existence of a model ontology, which we do not have today. As a result, the current approaches to dealing with model inconsistencies also depend on management. In fact, the viewshed analyses described above are as applicable to line of sight algorithm differences as they are to environmental representation differences; they are useful in placing the environment within its functional context of use.

Viewshed type analysis is tractable for the algorithms that use environmental data directly, because they are sufficiently limited in number. It is difficult to make the same claim for the general class of algorithms and models that form the basis for all system – to - system interactions in simulation events. Thus, the current management approaches usually include some type of circumscription, limiting the potential interaction types to the minimum required. The best solution here is not unlike that for ensuring object model compatibility. Devising an ontology that can describe a core set of accepted models would serve the community well. It would be particularly useful as an enabling resource to provide modelers with a common vocabulary for describing widely used and accepted models.

In sum, the community has not made appreciable progress towards ensuring valid interactions derived from semantic interoperability. However, there is a way forward and progress can be made. The necessary ingredients are common components of object models, informed management of environmental data use, and development of an educated community that uses a common modeling ontology.

REFERENCES

Base Object Model (BOM) Template Specification (2006), Simulation Interoperability Standards Organization (SISO), www.sisostds.org.

Carter, S. (2007). *The New Language of Business, SOA and Web 2.0,* Upper Saddle River, NJ, IBM Press.

Friedman, M. H., Tomkinson, D. M., Scott, L. B., O'Kane, B. L., and D'Agostino, J. (1989). Standard Night Vision Thermal Modeling Parameters, *Proceedings of the SPIE Annual Meeting*.

IEEE Standard for Distributed Interactive Simulation – Application Protocols (1995), IEEE Standards Board, www.ieee.org.

IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) – Object Model Template (OMT) Specification (2001), IEEE Computer Society, www.ieee.org.

Morgan, M. and Morrison, M. (1999). *Models as Mediators,* Cambridge, Cambridge University Press.

Ray, C. (1994). *Representing Visibility for Sighting Problems.* PhD. Thesis, Rensselaer Polytechnic Institute.

Richbourg, R., Graebener, R., Stone, T., & Green, K. (2001). Verification and Validation of Federation Synthetic Natural Environments. *Proceedings, Interservice / Industry Training, Simulation, and Education Conference*, November 29, 2001.

The Test and Training Enabling Architecture (TENA) Architecture Reference Document (2005), Central Test and Evaluation Investment Program (CTEIP).

Woodward, J. (2003). *Making Things Happen, A Theory of Causal Explanation,* Oxford, Oxford University Press.