Agile Data Logging and Analysis

Ke-Thia Yao, Gene Wagenbreth and Craig Ward
Information Sciences Institute
University of Southern California
4676 Admiralty Way, Marina del Rey, CA 90292
{kyao, genew & cward}@isi.edu

ABSTRACT

The High Level Architecture Object Model Template (HLA OMT) supports simulation interoperability by providing a Federation Object Model (FOM) to formally describe the information interchange (objects, object attributes, interactions, and interaction parameters) within a simulation federation. Information used by a single federate within the federation is defined by the Simulation Object Model (SOM).

Often the federate SOMs are mutually incompatible, so standing up a federation typically requires a tedious process modifying the simulation federates to conform to the proposed FOM. A variety of agile FOM techniques have been proposed to facilitate this integration process.

From the simulation data logging and analysis perspective, there is an analogous problem of adapting the analysis tools to particular federations. Data analysis tools are designed in accordance with the analysts' notion of Measures of Effectiveness (MOE) and Measures of Performance (MOP). Often these measures are not directly compatible with respect to the underlying federation object model. This is especially troublesome for the lower-level MOP, which must have common characteristics with the logged FOM data.

This paper presents a two-layered framework that supports the agile adaptation of analysis tools to specific federations. The top semantic layer provides a modeling framework to capture concepts that analysts tend to use. The concepts include measurements and dimensions. Examples of dimension include are object classifications, time, and geographic containment. The lower syntactic layer describes how to map the particular federation object models to more abstract semantic concepts. In addition, we show how this approach supports reuse by taking advantage of the hierarchical nature of the object models. These concepts are now being successfully implemented and evaluated in the Joint Forces Command Urban Resolve 2015 experiment.

ABOUT THE AUTHORS

Ke-Thia Yao is a research scientist in the University of Southern California Information Sciences Institute, working on the JESPP project, which has the goal of supporting very large-scale distributed military simulation involving millions of entities. Within the JESPP project he is developing a suite of monitoring/logging/analysis tools to help users better understand the computational and behavioral properties of large-scale simulations. He received his B.S. degree in EECS from UC Berkeley, and his M.S. and Ph.D. degrees in Computer Science from Rutgers University.

GENE WAGENBRETH is a Systems Analyst for Parallel Processing at the Information Sciences Institute at the University of Southern California, doing research in the Computational Sciences Division. He specializes in tools for distributed and shared memory parallelization of Fortran programs and has been active in benchmarking, optimization and porting of software for private industry and government labs.

Craig E. Ward is a Parallel Computer Systems Analyst at the Information Sciences Institute of the University of Southern California. He is active in programming on the Joint Experimentation on Scalable Parallel Processor project, focusing on the data management issues. He has a B.A. in History from the University of California, Irvine, and an M.S. in Computer Science from Loyola Marymount University.

Agile Data Logging and Analysis

Ke-Thia Yao
Information Sciences Institute, University of Southern California
Marina del Rey, California
kyao@isi.edu

INTRODUCTION

The High Level Architecture Object Model Template (HLA OMT) supports simulation interoperability by providing a Federation Object Model (FOM) to formally describe the information interchange (objects, object attributes, interactions, and interaction parameters) within a simulation federation. Information used by a single federate within the federation is defined by the Simulation Object Model (SOM).

Often the federate SOMs are mutually incompatible, so standing up a federation typically requires a tedious process modifying the simulation federates to conform to the proposed FOM. A variety of agile FOM techniques have been proposed to facilitate this integration process.

From the simulation data logging and analysis perspective, there is an analogous problem of adapting the analysis tools to particular federations. Data analysis tools are designed in accordance with the analysts' notion of Measures of Effectiveness (MOE) and Measures of Performance (MOP). Often these measures are not directly compatible with respect to the underlying federation object model. This is especially troublesome for the lower-level MOP, which must have common characteristics with the logged FOM data.

This paper presents a two-layered framework that supports the agile adaptation of analysis tools to specific federations. The top semantic layer provides a modeling framework to capture concepts that analysts tend to use. The concepts include measurements and dimensions. Examples of dimension include are object classifications, time, and geographic containment.

The lower syntactic layer describes how to map the particular federation object models to more abstract semantic concepts. In addition, we show how this approach supports reuse by taking advantage of the hierarchical nature of the object models. These concepts are now being successfully implemented and evaluated in the Joint Forces Command Urban Resolve 2015 experiment.

AGILE DATA FRAMEWORK

The type of measure of effectiveness/measure of performance questions of interest to analysts are typically not directly captured by simulation loggers. In general analysts are interested in how well higher level mission tasks and objects are satisfied. An MOE is a question, or a measure, designed to illuminate how well particular mission tasks are satisfied with respect to a system (Gentner *et. al.*, 1996).

An MOP is typically a quantitative measure of a system characteristic used to support an MOE. For example, sample MOE questions are *"Can the red forces be pinned?"*, or *"Can the sensors detect red force movement within urban environment?"*. MOPs supporting these MOEs are typically statistical in nature. They may include percentage of red forces killed/damaged, percentage of blue forces killed/damaged, time taken for red forces to cross terrain, percentage of red forces detected within sensor footprint, percentage of red forces detected total, and percentage of detection by sensor type by terrain type by time of day.

Simulation loggers do extremely well at capturing detailed operational data. Our distributed data loggers have captured terabytes of simulation data for the Urban Resolve Phase I exercises. Operational data include individual entity state changes and interactions among the entities. Depending on the type of entity, entity state changes may include *location*, *orientation*, and *velocity*. For vehicles, additional attributes may include *external lights on* and *engine on*. Interactions may include *collision*, *damage assessment*, *sensor detection*, and *contact report*. Only after appropriate processing are these types of operational data of potential use to the analysts.

The logging data collected from the simulation is at too low a level to be of direct use to the analysts. Information needs to be abstracted from the logged data by collation, aggregation and summarization. In order to perform this data transformation an *analysis data model (ADM)* has to be defined that is suitable for analysts and decision makers. We propose using a multidimensional data model as way representing the infor-

mation from their perspective, see **Section Defining the Analysis Data Model**. Next, a *logging data model (LDM)* of representing the collected data has to be defined, see **Section Logging Data Model**. This logging data model is defined in the context of HLA federations. In particular for the JSAF and SLAMEM simulation federates for the Urban Resolve exercises. Then, to bridge the gap and connect these two data models, we define abstraction relationship that maps the logging model to the analysis model, see **Section Mapping between LDM and ADM**. The **Implementation Section** describes our schema design and an initial graphical data abstraction mapping tool, as a part of the Scalable Data Grid toolkit (Yao and Wagenbreth, 2005). **Section Distributed Logging and Analysis** describes how these logging and analysis data fit into the process of distributed simulation environment used by US Joint Forces Command (USJFCOM).

DEFINING THE ANALYSIS DATA MODEL

One of the key focus areas of Urban Resolve exercises is to study the effectiveness of future Intelligence, Surveillance and Reconnaissance (ISR) sensors in helping soldiers operate in complex urban environments. The Sensor/Target Scoreboard provides a visual way of quickly comparing the relative effectiveness of individual sensor platforms and sensor modes against different types of targets (Graebener *et al.*, 2003; Graebener *et al.*, 2004). Sensor/Target Scoreboard is a specific instance of the more general multidimensional analysis (Kimball *et al.*, 1998). We use the Sensor/Target Scoreboard to motivate the discussion. In a 2005 I/ITSEC paper, we described the data management and analysis tool Scalable Data Grid that uses multidimensional analysis (Yao and Wagenbreth, 2005).

In the Urban Resolve exercise, simulated sensor entities lay down sensor footprints to delimit sensor coverage sweep. For each target entity within the footprint, a contact report is generated to hold the result of the sensor detection. The contact report includes information about the sensor entity, the platform the sensor entity is mounted on, the sensor mode, the target entity, the detection status, the perceived target type, the perceived target location, the perceived target velocity, and so on.

Sensor/Target scoreboards have the capability of providing summary views by aggregating individual sensor platforms into sensor platform types, such as *high altitude*, *medium altitude*, and *low altitude*. And, it aggregates individual target entity objects into target classes, which can range from the generic (like *Civilian Large Trucks*) to the specific (like Russian *MAZ-543 MEL*).

The Sensor/Target scoreboard is an example of two-dimensional of a multidimensional cube. Its two dimensions are the sensor dimension and target. One can imagine extending the scoreboard to take into account, say, weather conditions and time of day. This would add two more dimensions to form a four-dimensional cube.

We define our Analysis Data Model (ADM) in terms of multidimensional analysis. The ADM model consists of two key concepts *dimensions* of interest and *measures* of performance. Dimensions are used to define the coordinates of multidimensional data cubes. The cells within this data cube are the measure values.

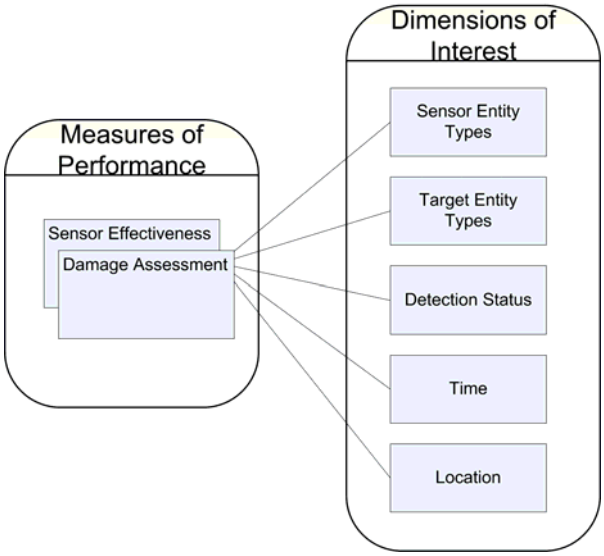

Figure 1. Dimensions of Interest and Measures of Performance

Dimensions of Interest

For large simulations, like the Urban Resolve exercises, the magnitude of data collected ranges in the terabytes. *Dimensions* categorize and partition the data along lines of interest to the analysts. Defining multiple crosscutting dimensions aids in breaking the data into smaller orthogonal subsets.

Dimensions have associated measurement units, or *coordinates*. Choosing the granularity of these units aids in determining the size of the subsets. For example, depending of the dynamic nature of the phenomenon the analysts are trying to study, they may choose to

define time dimension units in term of minutes, days, weeks or years.

Another dimension example is in terms of simulation entity groups. For the sensor dimension in the sensor/target scoreboard, the analyst may want to group together sensors by the type of platform: high-flying UAVs (unmanned aerial vehicles), mid-altitude UAVs, OAVs (organic air vehicles) and UGSs (unattended ground sensors). The targets may be group together, for example, by transportation mode: air, ground and sea.

Hierarchical dimensions define how a coordinate relates to other coordinates as its subset. It serves to group together similar units from the analyst's perspective. By defining hierarchical dimensions, analysts inform the system on how to aggregate and summarize information. For example, the analysts may want to subdivide the sensor platform category into the sensor modes: MTI (moving target indicators), SAR (synthetic aperture radar), images, video, and acoustic.

Hierarchical dimensions can be viewed as a partial ordering relationship. At the top node of the partial ordering is the set containing all the coordinates. The bottom nodes of the partial ordering are singleton sets containing just one coordinate. Edges between nodes indicate superset/subset relationship. A node's parent is its superset. A node's child is its subset.

Nodes in hierarchical dimensions are also given coordinates. We call coordinates for non-singleton nodes *abstract coordinates*. Coordinates for singleton nodes are the *concrete coordinate* of the single element in set.

Multiple decompositions of the same dimension are also useful. For example, there may be many different simulation federate types playing in the federation. The analysts may want to verify how each federate response to the sensor contacts, so they define the category to be the type of federates.

Measures of Performance

After the data have been partitioned along lines of interest, the data subsets may still be large. *Measures* provide quantitative ways of characterizing the data subsets. A key characteristic that measures should have is that they are can be *aggregated*. The hierarchical crosscutting dimensions partition the data into a hierarchy of subsets. The measure must be able to provide a meaningful summarization. To be computational efficient the measure aggregation operator must satisfy the *associative* and *commutative* properties—the measure of a set must be computable from the measures its subsets.

In the case of the sensor/target scoreboard the measure of performance is simply an integer count of the number of times a sensor has detected a target. The aggregation operator is the addition operator.

Sometimes mean and variance performance measures are of interest to analysts. For example, instead of integer detection counts, the sensor/target could be extended to maintain a floating-point number indicating degree of uncertainty. Then, in this case it makes sense to measure the mean and the variance of uncertainty. As defined the mean and variance operators do not satisfy the associative and commutative properties. For example, given only the means of two subsets, it is not possible to compute the mean of the union of the two subsets. However, the mean and variance measures are computable from other measures that are efficiently computable. Mean can be computable from two associative measures: the count of number of detections (or uncertainties), and the sum of uncertainties. Variance requires an additional sum of squares of uncertainty. Let X be the uncertainty, and n be the count of number of detections:

$$Mean = \frac{\sum X}{n}; \quad Var = \frac{n\sum X^2 - \left(\sum X\right)^2)}{n^2}$$

Correlating Measures

Typically MOE decomposes into multiple performance measures. If it is possible to decompose these measures along the same dimensions, or sometimes called conforming dimensions, then it is possible to compare these measures. Figure 1 depicts an additional *damage assessment* measure. Potentially target entity, time and location dimensions are applicable to both damage assessment and sensor effectiveness. For example, an analyst may ask how more likely are detected enemy target entities to be damaged.

Let X be sensor effectiveness and Y be damage assessment. If define an additional measure that is the sum of X times Y, then we can determine covariance between damage and detection by using the "mean" operator:

$$Cov(X,Y) = Mean(XY) - Mean(X)Mean(Y)$$

LOGGING DATA MODEL

The Logging Data Model (LDM) describes the content and format of data being logged by SDG. SDG uses relational databases to store the logged data. In this case, the LDM is a basically a relational schema.

Logging Data Model Generation

HLA rules state that the FOM shall document the agreement among the federates on data to be exchanged at runtime. Our LDM is automatically generated from the FOM description. FOM describes objection classes and interactions. Classes have attributes, and interactions have parameters. The attributes and interactions are defined in terms of primitive data types and complex data types.

For each object class we create one top-level table within the relational schema. Simple primitive class attributes mapped to columns in the table. Attributes with complex data types many be either mapped to multiple columns, or to rows in a sub-table. Complex data types with fixed length and with low cardinality are mapped to multiple columns in the top-level table. For example, a location attribute with x,y,z coordinates are mapped three columns. Complex data types with high or unbounded cardinality, such as arrays, are mapped to sub-tables. The sub-table contains keys referencing the parent table, sequence column indicating the order in the array and data columns representing the actual data. Depending on the data type, it is possible to have sub-table of sub-table tables. Interactions and their corresponding parameters are handled similarly to the object classes and their attributes.

The purpose of the relational schema is for the efficient storage of log data intercepted during the federation execution. It is not intended to capture all the information contained within the FOM. Uschold (2004) points out that the primary purpose of relational schema is *data integrity*. For example, database foreign key constraints can be used to enforce that a row in a sub-table must have a corresponding row in the parent table. When deleting the parent row all the corresponding sub-table rows must also be deleted. However, such data integrity constraints are expensive to enforce. In practice many applications such as the SDG identifies these constraints during but does explicitly enforce them on the collected data. However, we do use foreign key constraints on the Analysis Data Model.

Intercepting/Decoding HLA Messages

HLA rules state that all exchange of FOM data among federates shall occur via the RTI, and that federates shall interact with the RTI in accordance with the HLA interface specification. RTI-s, a highly-scalable implementation of the RTI (Helfinstine *et. al.*, 2003), provides an interceptor plug-in framework that exposes calls to this HLA interface to registered plug-ins, see Figure 2. SDG exploit this plug-in to intercept and log messages federate attribute updates and interaction sends. With respect to the RTI, the contents of these messages are raw binary strings. RTI provides publish/subscribe facilities to exchanges these messages, but not decode their contents. To provide query and analysis capabilities, these messages must be decoded with respect to FOM.

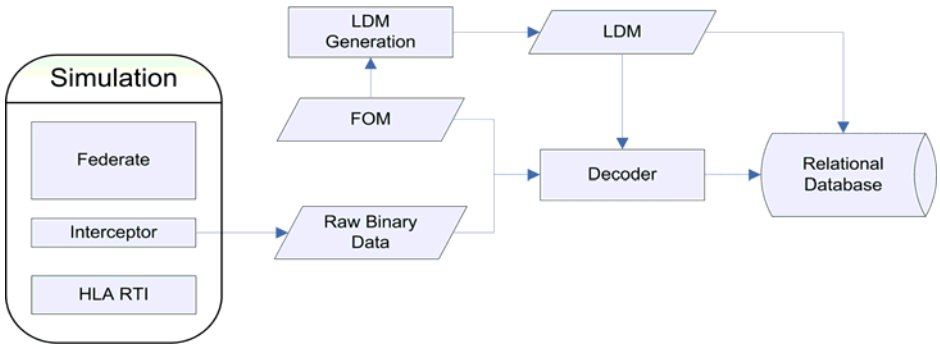

Figure 2. Logging data flow

MAPPING BETWEEN LDM AND ADM

The data intercepted from the simulation federates is stored according to the Logging Data Model (LDM). To provide useful information to the analysts the data has to be abstracted and aggregated by translated it into the Analysis Data Model (ADM). Figure 3 depicts the data flow for this translation process.

Message Extraction

The purpose of the message source extraction is to provide an interface layer that hides FOM variations from the rest of mapping process. Message extraction performs include: filtering to keep on the columns needed by the analysis data model, and partitioning values. Currently, message extraction is implemented using SQL select statements.

As described in the previous section log data is stored within a relational database using a LDM schema that is automatically generated from the FOM. In practice, the information within the LDM schema is a super set of the information needed by analysis data model. Message extraction prunes away columns that are not needed.

The rationale for partitioning is two-folds. One is the numerical precision that simulators are capable of is often not needed by the analysts who are interested in summary and aggregate information. Message extraction take the initial processing step needed to map the logged toward analyst's notion of dimension of interest. Second is such high precision generates large cubes that takes huge storage space. The analysts have to decide the minimum granularity he wishes to examine the data.

Partitioning can be performed on single numerical values, or vector values. The prototypical example of single numerical value is time. JSAF typically keeps time at the resolution of seconds. SDG logs message on the order of milliseconds. An example of vector values is geographical location. JSAF is capable of sub-meter location resolution.

The partitions do not necessarily have to be uniform and linear. To emphasize particular time periods and locales where actions are taking place, analysts should be able to define smaller grain partitions. For example, in urban center where the battles occur the grid size can be defined relatively small, say one hundred meter squared or ten meter squared. Then the grid size can progressively increased the locale move towards the outskirts of the city.

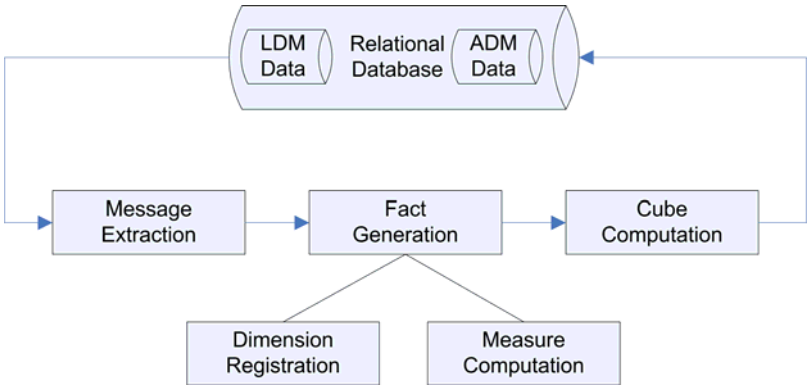

Figure 3. Data flow for mapping data stored in the Logging Data Model to the Analysis Data Model

Fact Generation

The inputs to the fact generator are messages, and the outputs are coordinates into the cube along with the measures associated with the cube. Two operations that the fact generator has to perform are: *dimension registration* and *measure computation*.

The purpose of dimension registration is to assign the facts to their proper concrete coordinates. In the case of sensor/target scoreboard, given a fact that states sensor UAV-347 detected M1A1-tank-14, determine the cell in the scoreboard that needs to be updated. The dimension registration involves two steps. The first is a precomputation step. All the partitions within a dimension have to be assigned unique numbers or coordinates. Second step is, given a fact, determine which partition it belongs. For entity-based dimension, determining the partition may just be a simple table lookup. For numerical valued dimensions, algebraic or piecewise linear mapping functions may be used. For geo-location dimensions, simple computational geometry containment may be used.

Once the dimension coordinates have been determined, measure computation determines what value to put into the cube cell. As we have seen in the analysis model section many common measure can be computed using simple polynomials. For counts the measure simply returns the number one. For computing mean, the value itself has to be returned. For variance, an additional

square of the value is needed. For covariance, multiplying the two individual measures is needed.

Cube Computation

Facts are concrete observations from the simulation. The facts have to be aggregated according to how hierarchical dimensions are defined. The cells corresponding to the abstract coordinates of the cube have to be computed. Here we assume the aggregation operator satisfies the associative and commutative properties. Given these assumptions we can efficiently compute measures for all the abstract coordinates by doing a bottom up traversal of the partial ordering hierarchy.

IMPLEMENTATION

As part of the Scalable Data Grid (SDG) toolkit, we are developing a prototype implementation of this agile data analysis framework. This framework is currently being tested within the Urban Resolve 2015 exercises.

Meta-level Analysis Data Schema

In designing this implementation, we have strived to maintain flexibility. Figure 4 depicts a relational schema that implements the Analysis Data Model. Approximately the top third of the figure describes the dimensions, and the bottom two-thirds describes the measures and facts.

For efficiency reasons, data warehouse representation of multidimensional cubes typically use what is known as the star schema (Kimball *et. al.*, 1998). The star schema explicitly defines one relational table for each dimension. For the sensor/target scoreboard, the star schema would use two dimension tables to define the sensor dimension and the target dimension. One interpretation of the star schema is that it hard codes the two dimensions into the relational schema.

Instead of hard coding, our approach is to define a meta-level schema that is capable defining and expressing multiple dimensions, see Figure 4. In our formulation, the sensor and target dimensions are defined as data, *i.e.* rows in the meta-level relational table.

The `sdg_cube` table represents multiple dimensional cube definitions. Each cube is defined by an ordered list of dimensions (`sdg_dimensionDesc` table). Each dimension has a name and an English description (`sdg_dimension` table). Each dimension is defined by a set for concrete and abstract coordinates (`sdg_dimensionValue` table). These hierarchical coordinates form a partial ordering (`sdg_valuePartialOrdering` table). Similar types of coordinates are grouped together and given a name (`sdg_dimensionNode` table). For example, Figure 5 shows that concrete coordinates are grouped into the Entity node. Then, the Country node is used to group together coordinates for US, USSR, Iraq and so on. The partial ordering of the coordinates also induces a partial ordering of the nodes (`sdg_PartialOrdering` table).

In a similar fashion we do not hard code measures, measure aggregation operators, and facts into fixed tables. We define meta-level tables to store these data models as data, see Figure 4.

Analysis Data Model Editor

The advantage of using a meta-model is that we can easily and quickly design analysis data models tailored to the needs of the analysts.

Figure 5 shows a screen dump of our prototype SDG cube dimension editor. The editor presents to the user a tabbed view of the dimensions. Each tab corresponds to one dimension. The figure depicts a three-dimensional sensor/target/detection status scoreboard. The detection status dimension breaks sensor contact reports down into four types: detected, not detected due to line of sight, not detected due to velocity, and not detected due to concealment.

The top-half of each tab depicts the dimension as a tree-table. Each node/row in the tree-table represents a dimension coordinate. For example, the entity `vehicle_Swenden_CIV_Bus` has coordinate 37, and the entity `vehicle_Sweden_CIV_sm_car` has coordinate 38. Both of these concrete coordinates belong to the abstract coordinate -44, called `Sweden`. `Sweden` in turn belongs to abstract coordinate -224, called `Land`.

The tree-table has editing features that allows users to quickly define new coordinates or modify existing partial orderings. Editing operations include adding a new node/row; editing the content of a table cell; promote a node up the hierarchy; and demoting a node. Cut and paste operations are also defined.

The editor uses Java JDBC to directly connect to relational databases in order to load and to store the dimension definitions.

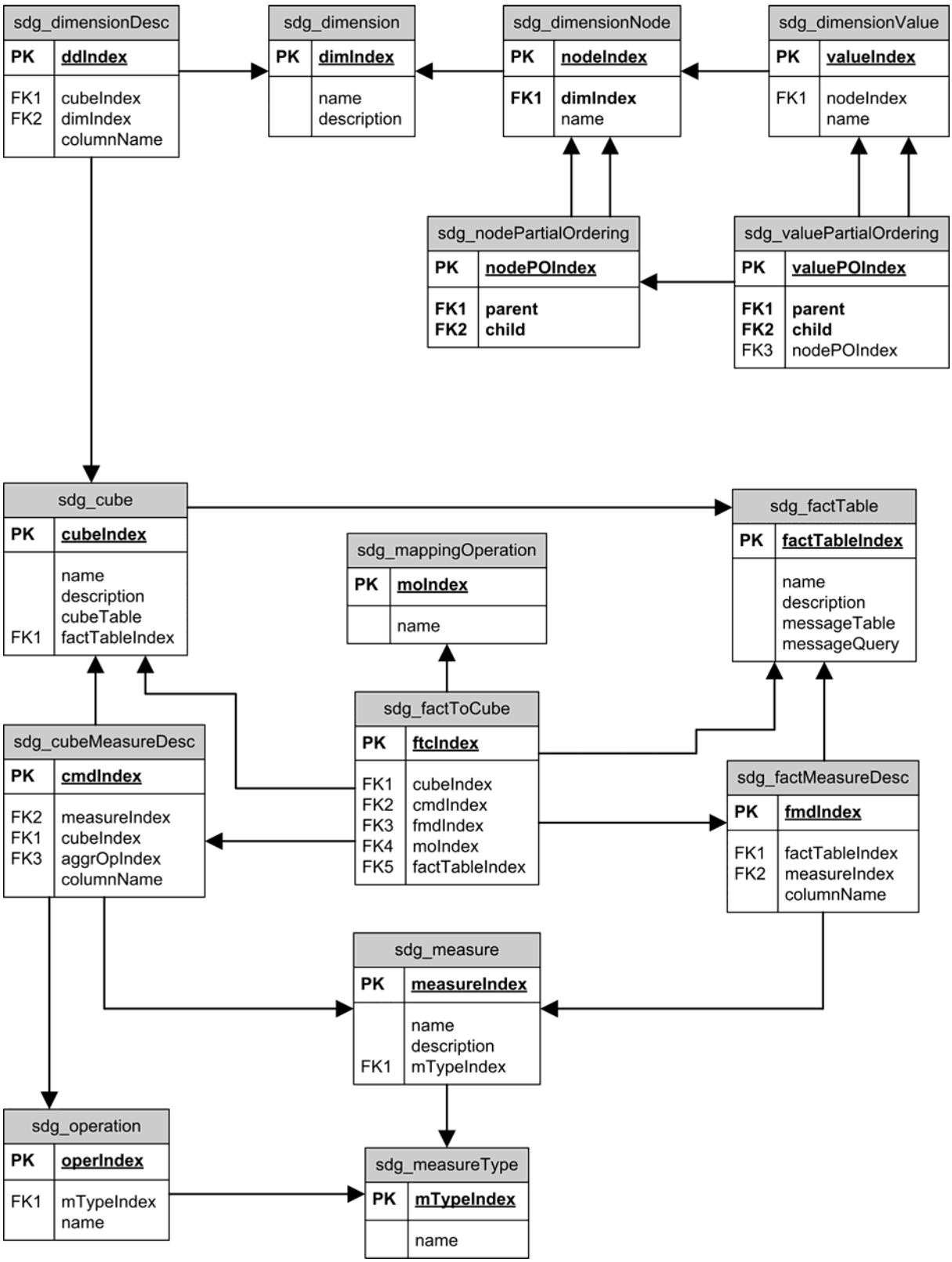

Figure 4. Meta-level Analysis Data Model Schema

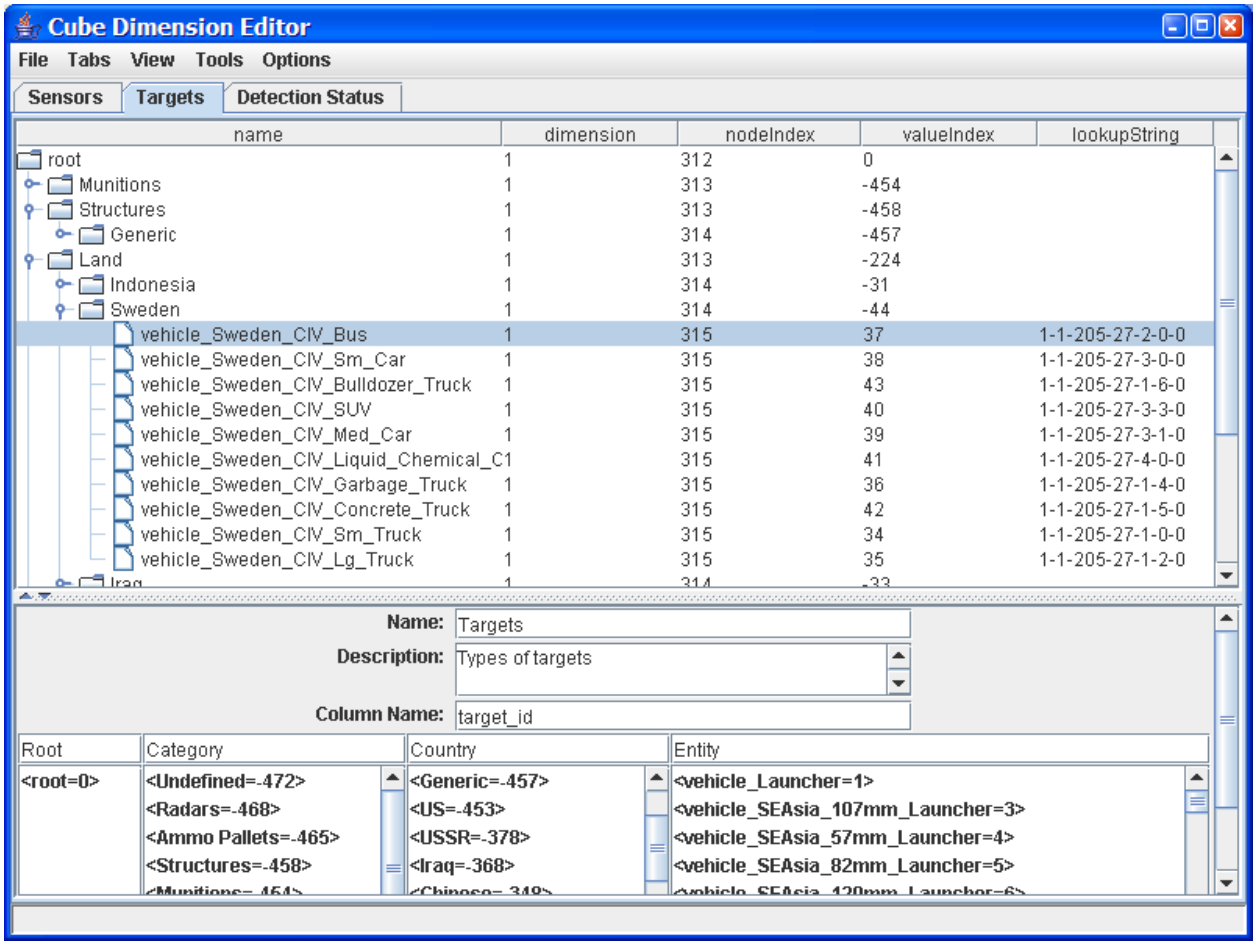

Figure 5. Cube Dimension Editor

DISTRIBUTED LOGGING AND ANALYSIS

The goal of the Urban Resolve exercises is to evaluate how sensor capabilities can aid the war fighter in urban environments. Urban terrains are complex, and for realism they must be populated with many civilian entities. The sensors should be able to detect enemy forces trying to blend into the hustle and bustle of city life. Simulating such urban environments requires tremendous amount of distributed computer resources (Lucas & Davis, 2003). Urban Resolve exercises have been simultaneously running on multiple Linux clusters from Maui High Performance Computer Center, ASC and USJFCOM J9.

So far the paper has focused on creating a multidimensional cube on a single computer. To work in distributed environments we need to define an additional layer on top to aggregate multidimensional cubes distributed across different machines.

The left-hand side of Figure 6 depicts a single three-dimensional sensor/target/detection status score-cube. This cube is generated from data logged from the local simulation federate. It represents only a partial, incomplete view. To generate a complete view, cubes from other simulation federates have to be aggregated.

The right-hand side of Figure 6 depicts a tree summing together all the distributed cubes. Again we can use of the associative and commutative properties of the aggregation operator. We do not have to send the raw facts, which can be potentially bandwidth intensive. We only have to send the cubes themselves.

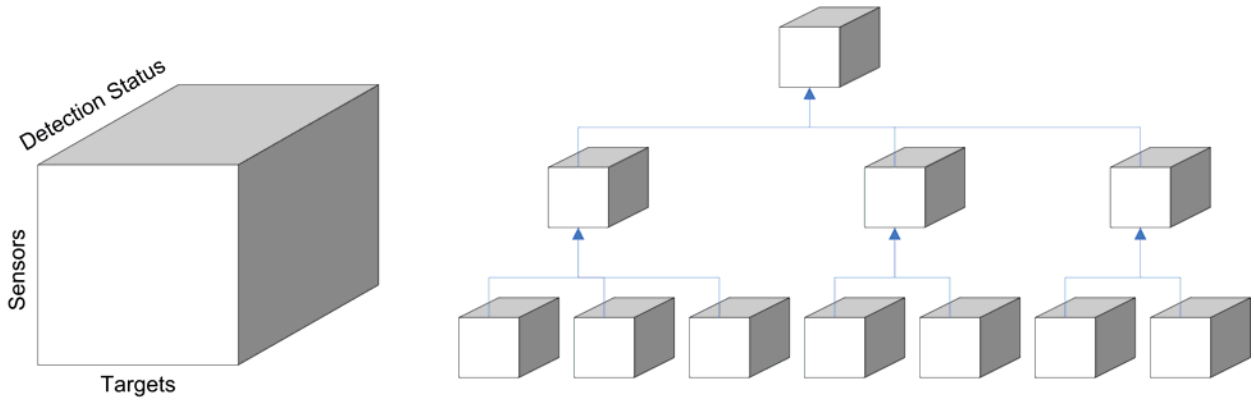

Figure 6. Distribute data analysis

RELATED WORK

Previous research in the simulation has focused on creating data frameworks to facilitate simulation federates participating in federations using different FOMs.

Agile FOM Framework (AFF) describes a mapping methodology for translating back and forth between the internal federation SOM representation and the external FOM representation (Macannuco *et al.*, 1998; Wilbert *et al.*, 1999). Types of AFF mappings include naming convention mappings (*i.e.*, `Position` to `Location`), attribute group mappings (three attributes representing location to one vector attribute), unit conversions, byte swapping and so on. In the work describe in this paper, we do not address these types of mappings. Potentially, in our framework we could define an additional AFF layer. Instead storing the logging data in the FOM representation, we would store the data after AFF mapping by means of the Agile FOM Interface. The advantage of storing AFF mapped data is two-folds. One is that AFF provides unit conversions. Analysts may prefer one unit measurement representation over another. The other is that some of our low-level processing tools, such SQL scripts, do not have to be rewritten to adapt different naming and grouping conventions.

Another interoperability approach is to provide a common Object Model Data Dictionary from which SOM and FOM developers can choose to incorporate into their models (Hammond *et. al.*, 1998). This approach can potentially work in tightly knit communities where developers can agree upon a common representation, and in domains where the modeled elements are relatively static and do not change frequently.

Conceptually similar approach to the AFF is to use more formal knowledge presentations to describe the FOM and SOM models (Rathnam and Paredis, 2004). This work defines a world ontology in which FOM and SOM ontologies can be expressed. Then, a formal mapping is defined between the SOM and FOM ontologies.

CONCLUSION

The ability to capture and log detail message traffic from very large scale simulations is exceeding our ability to analyze and comprehend that data. This paper describes a framework for quickly translating these operational-level log data into analyst-level data that are capable of supporting decision makers. The framework explicitly defines a two-level data model that separates the operational logging data model from the analysis data model. The agility of the framework results from being able to isolate changes to the logging data model as a result of changes to the federation object model, and from being able to quickly define analysis data model that match analyst notion of measure of effectiveness and of performance.

ACKNOWLEDGEMENTS

I would like to thank Dan Davis for his careful review and thoughtful comments on this paper.

This material is based on research sponsored by the Air Force Research Laboratory under agreement number FA8750-05-2-0204. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon.

The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force Research Laboratory or the U.S. Government

REFERENCES

Graebener, R., Rafuse, G., Miller, R., & Yao, K.-T. (2003). The Road to Successful Joint Experimentation Starts at the Data Collection Trail. *Interservice/Industry Training, Simulation, and Education Conference*.

Graebener, R., Rafuse, G., Miller, R., & Yao, K.-T. (2003). The Road to Successful Joint Experimentation Starts at the Data Collection Trail—Part II. *Interservice/Industry Training, Simulation, and Education Conference*.

Hammond, J., Dey, M., Scrudder, R., & Merkin, F., (1998). Populating the HLA Object Model Data Dictionary—A Bottom-Up Approach. *Simulation Interoperability Workshop*. 98S-SIW-075.

Helfinstine, B., Torpey, M., & Wagenbreth, G. (2003). Experimental Interest Management Architecture for DCEE. *Interservice/Industry Training, Simulation, and Education Conference*.

Kimbal, R., Reeves, L., Ross, M. & Thornwaite, W. (1998) *The Data Warehouse Lifecycle Toolkit*. Hoboken, New Jersey: Wiley.

Macannuco, D., Dufault, B., & Ingraham, L. (1998). An Agile FOM Framework. *Simulation Interoperability Workshop*.

Lucas, R., & Davis, D. (2003). Joint Experimentation in Scalable Parallel Processors. *Interservice / Industry Training, Simulation, and Education Conference*.

Rathnam, T., & Paredis, C.J.J. (2004) Developing Federation Object Models Using Ontologies. *Proceedings of the 2004 Winter Simulation Conference*.

Tan, G., Hu Y., & Moradi, F. (2003). Automatic SOM Compatibility Check & FOM Development. *Distributed Simulation and Real-Time Applications*.

Wilbert, D., Macannuco, D. & Civinskas, W. (1999). A Tool for Configuration FOM Agility. *Simulation Interoperability Workshop*. *99F-SIW-116*.

Yao, K.-T., & Wagenbreth, G. (2005). Simulation Data Grid: Joint Experimentation Data Management and Analysis. *Interservice / Industry Training, Simulation, and Education Conference*.