Successful Joint Experimentation Starts at the Data Collection Trail—Part II

Robert J. Graebener, Gregory Rafuse, Robert Miller & Ke-Thia Yao
M&S Team, Experimentation Engineering Department, J9 USJFCOM
Suffolk, Virginia
rgraeben@ida.org, grafuse@alionscience.com, rmiller@alionscience.com & kyao@isi.edu

ABSTRACT

Last year Joint Forces Command's, Joint Experimentation Directorate (J9) initiated planning and development in technical support of the most complex experiment (URBAN RESOLVE) undertaken to date. The experiment trials (Summer 2004) will explore future concepts and technologies for achieving situational awareness and understanding when operating in a robust large-city urban environment. In addition, the need for generating quantifiable results took on a renewed level of interest. The Commander, Joint Forces Command directed that future experiments provide findings that can survive critical scrutiny, particularly if those transformational products and solutions are to be promulgated across the Department. The authors' add another chapter to last year's paper, as they craft a system for providing more creditable and quantifiable data to support experiment findings. This paper will cover: changes made in the initial plan for data collection and analysis as new challenges arose along the way; the technical issues related to the architectural choices; as well as the challenges awaiting the group of individuals charged with maintaining a nationwide, distributed federation and network whose ultimate goal is to provide cogent, traceable data generated from the federation and human-in-the-loop player inputs. In preparing for the experiment trials, initial data storage assumptions gave way to the realities of finding more robust methods of collection as bandwidth traffic increased as federation architectures were modified to support emerging user requirements. Innovative approaches on how near-real-time data would be collected were instantiated as attention turned towards the post-processing needs that would sustain the experiment analysis team in the months following the trials. Integrating scalable parallel processors and addressing issues dealing with the means for storing and retrieving extremely large quantities of data added to the challenges. Finally, major lessons learned will be addressed from a transformational perspective.

ABOUT THE AUTHORS

Bob Graebener, Colonel, United States Army (Retired), has been a member of the professional staff at the Institute for Defense Analyses (IDA) since March 1997. He is currently a Research Staff Member at IDA and has primarily involved in supporting JFCOM J7/J9 in Modeling and Simulation related matters for the past nine years. He is currently working towards a doctoral degree in Systems Engineering from GWU.

Gregory Rafuse is a data collection analyst and developer and is currently the lead developer for the data collection toolkit. He is a Software Engineer with Alion Science and Technology. Mr. Rafuse has previously served seven years with the US Army as a Field Artillery Crewman. He also possesses an AAS in Computer Information Systems (CIS) from McLennan Community College and is pursuing a BS in CIS from Strayer University.

Robert Miller is a Senior Software Engineer with Alion Science and Technology. He brings over 11 years of experience to the current effort of designing, coding, and testing software for the Future After Action Review System. He holds a Bachelors Degree in Engineering from The Cooper Union School of Engineering and a Masters Degree in Computer Science from the City University of New York.

Ke-Thia Yao is a research scientist in the Distributed Scalable Systems Division of the University of Southern California Information Sciences Institute. Currently, he is 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. For his Ph.D. thesis he implemented a spatial and physical reasoning system that automatically generated grids for novel geometries for computational fluid dynamics simulators.

Successful Joint Experimentation Starts at the Data Collection Trail—Part II

Robert J. Graebener, Gregory Rafuse, Robert Miller & Ke-Thia Yao
M&S Team, Experimentation Engineering Department, J9 USJFCOM
Suffolk, Virginia
rgraeben@ida.org, grafuse@alionscience.com, rmiller@alionscience.com & kyao@isi.edu

PRELUDE

The reader should be aware that the title of this paper ends with "—Part II". Those familiar with human-in-the-loop simulations like JSAF (Joint Semi-Automated Forces) and joint experiments set in the year 2018, such as URBAN RESOLVE, realize that when one pushes the boundaries of simulation-support-to-experimentation a discovery process, in its own right, is created as the bounds of "what can be done in simulation" is continually challenged and superseded. Over the course of the past year, what started as a concept for developing "a best approach for collecting and analyzing data" gave way to the practical experience gained through the number of integration events necessary to prepare for the formal trials. The authors' felt it necessary to add another chapter to last year's journey (Graebener, et. al., 2003).[1]

INTRODUCTION

The initial concept of how to approach data collection and analysis when faced with a simulation federation that could generate data records in the terabyte range has evolved over the past year. Whereas PART I laid out the challenges associated with extremely large data generation conditions and the initial approach for meeting the experiment data collection requirements, and significant detail of the major changes will follow.[2]

This paper will cover:

- Changes made in the initial plan for data collection and analysis as new challenges arose along the way, as well as technical issues related to the architectural choices;
- Subsequent modifications in the data analysis tools to meet the changing user requirements,
- Challenges awaiting the group of individuals charged with maintaining a nationwide distributed federation and network whose ultimate goal is to provide cogent, traceable data generated from the federation and human-in-the-loop player inputs.
- Finally, lessons learned will be addressed from a transformational perspective.

In preparing for this year's experiment trials, initial data storage assumptions gave way to the realities of finding more robust methods of collection when network traffic increased as federation architectures were modified to support changing/emerging user requirements. Innovative approaches on how near-real-time data would be collected were instantiated as attention turned towards the post-processing needs that would sustain the experiment analysis team in the months following the trials. Integrating scalable parallel processors and addressing issues dealing with the means for storing and retrieving extremely large quantities of data added to the challenges. (Table 1)

Table 1. Data Collected During Dress Rehearsal Week.

# of Data Records of Interest (stored in 576 tables)	Percent of Total Data Logged	Size of Database
264 million	15-20	45 GB
Average size of each record: 180 bytes		

BACKGROUND

The original concept behind the FAARS (Future After Action Review System) toolkit was based on utilizing commercial-off-the-shelf (COTS) products for collecting simulation data. The original design specifications for the FAARS toolkit comprises three separate modules; a Data Collection Module utilizing hlaResults as the federation data interceptor and storage transport, a Near Real Time Module utilizing MySQL as the data storage medium along with a Apache web server with PHP scripts as the data presentation and analysis medium, and a Post Event Analysis Module using MySQL as the data storage medium and a custom written C++ user interface for accessing stored data for processing and analyses. Although this design works well and is being used in several joint experiments, it was not robust enough to support the URBAN RESOLVE series. Initial testing results using the complex urban terrain and tens of thousands of entities being detected by a large

[1] Last year's paper will be referred to as PART I for the remainder of this paper.
[2] The authors recommend a review of last year's paper to serve as a point of departure. Go to: http://www.alionscience.com/pdf/Data_Collection.pdf

constellation of sensors were adequately handled using Scalable Parallel Processor clusters, however the methodology of using hlaResults as the data collector no longer met the requirements. The reasons for replacing hlaResults were:

1) hlaResults only works with an NG-style RTI. For the UR effort, we are using an s-style RTI, which is a different implementation loosely based on DMSOs RTI-NG v1.3 standard.

2) When hlaResults subscribes to ALL entity traffic this overwhelms the physical network interface and causes packets to be dropped at the physical interface, effectively "missing" information.

3) Due to the nature of how cluster computers function, a significant amount of the simulation event information could not be effectively be logged.[3] Based on these factors, a different data logging architecture was needed.

INTERCEPTOR/LOGGER

The Interceptor/Logger application, an early version described in PART I, is an application process that resides on individual simulation nodes within the federation.[4] The determining factor on where to utilize the mechanism is determined by which federates are publishing information needed for data collection. The interceptor/logger, utilizing functionality in the RTI Application Programming Interface, inserts "hooks" into the published data streams by the RTI and then splits off two child processes; one process that writes and compresses the intercepted data into binary "log" files and a second process that decodes the data stream and inserts the decoded data into an embedded database application called SQLite. A separate daemon process called "sqlited" handles incoming socket-based connection attempts to query information that has been stored in the local database. Figure 1 is a diagram of the process.

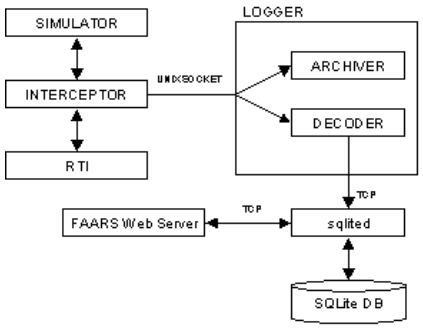


Figure 1. Interceptor/Logger Process

Because of the methodology of running interceptor/loggers on each simulator with data of interest, a separate mechanism was needed to retrieve information stored at each simulator location. A separate application process called "Aggregator" was developed that would handle the intercommunication between simulators logging data. The Aggregator is configured in a tree-like fashion, with a "Root Aggregator" at the head of the tree and "Child Aggregators" in branches from the root. The various branches reach out to the individual leaf instances of "sqlited" on each simulator. The interface to the Root Aggregator takes a Structured Query Language-formatted query and passes the query on to each branch Child Aggregators until the query finally reaches the individual instances of "sqlited". As each instance of "sqlited" responds with the requested data for the query, the Child Aggregators assemble the returned information in order of response and forwards the data on to the Root Aggregator which then assembles the complete returned information and forward it on to the original requestor. The Aggregator model works on Transmission Control Protocol (TCP) socket-based connections between the Root Aggregator and subsequent Children Aggregators.

Near Real Time Retrieval Of Data

With the utilization of the ISI interceptor/logger, the possibility of retrieving simulation information in a "near real time" manner became a reality. Typically, data collection efforts have had to wait until after collected logger files have been processed before any specific event information could be derived. This is a vast improvement in functionality and provides a wide range of uses that are still being realized as we move forward in the software development effort.

The Near Real Time data retrieval effort is based around the ability to query the ISI interceptor/logger application, retrieve the logged information from each node and store the retrieved information into a local

[3] The data could not be intercepted and logged by the hlaResults product because a significant amount of simulation traffic would be exchanged between SPP cluster nodes running the simulation and not transmitted outside of the cluster, a necessary prerequisite for hlaResults.

[4] Developed by the Information Sciences Institute (ISI) at the University of Southern California,

Relational Database Management System (RDBMS). The retrieved information is then used by the FAARS Near Real Time web server interface to allow users of the system to view various reports, charts and graphs based on the available information.

The process of retrieving intercepted information from each of the active ISI interceptor/loggers is handled by a series of BASH shell scripts on the FAARS web server. Each BASH shell script is targeted towards retrieving specific information, such as entity object states, and is used to process the retrieved information into the local RDBMS (aka cache). The data retrieval process is based on three steps. The first step is to send the request for information to the Root Aggregator. The methodology used by the retrieval process is based on making a TCP socket-based connection to the Root Aggregator and sending an SQL-formatted query. The second step is to wait for a response and process/validate the retrieved data and write this data to a temporary file. The expected response back from the Root Aggregator is a stream of plain ASCII text, which is tab-delimited for fields and is carriage return delimited for individual records. This information is then written to a temporary file in this same tab-delimited/carriage return delimited format. The third and final step is to then load the temporary file's data into the local cache.

The FAARS web server RDBMS cache uses MySQL v4.1.1. as the database engine. The database schema for the cache is based primarily on the schema used by the ISI interceptor/logger. This helps in facilitating compatibility with the information that is being utilized in near real time and data being reviewed post event. The main difference between near real time and post event processing is the different indexing schemas utilized on the local cache. The indices applied to the local cache database have been specifically tuned to support the types of queries that the FAARS web server uses for data displays.

Storage Space Requirements

When the overall design of the FAARS toolkit began to change to utilize the ISI interceptor/logger, physical storage space for collected data files and consolidated database became an issue. With the switch to using a larger-scale database engine than previously used and the need to analyze larger amounts of data than previously anticipated, the need for more physical media space became apparent. Where it was once thought that ten's of Gigabytes (GB) of storage space would be sufficient, it soon became apparent that this was not going to be acceptable. The central importance of disk space is its centrality to all three aspects of the process: storage of compressed logger files, storage space for staging uncompressed logger files while loading into consolidated database, and space needed for the final database tables and indices. What was finally settled on was a RAID 5 disk array totaling 1.7 Terabytes (TB) of disk space with a stand by disk array of 1.3 TB in size.

Because of the distributed nature of logging data that has begun to be utilized, it has become necessary to develop means to: retrieve all of the saved binary data logs on each simulator where the ISI interceptor/logger was instantiated; prepare and decode the binary data files, and then; insert the decoded data into a consolidated database representing the complete accumulation of data for a particular event. A process called "Data Staging" has been developed that accomplishes these tasks in an organized, efficient manner, making the best usage of available bandwidth, processing cycles and disk space. (See Figure 2) The Data Staging process begins with retrieving the binary log files at the end of each day's simulation run from each simulator logging data. The data is moved and stored on the local storage point in a hierarchical format based on the event name, day of the event and the simulator where the log file was retrieved. Once the data has been moved, Perl-based scripts are run against the individual binary log file to decode and format the binary data into plain-text, comma-separated value (CSV) flat files. The translation of the data and the creation of the storage database schema are based on utilizing definitions found in the Federation Object Model (FOM) and Federation Execution Document (FED) for the federation in use. Each CSV-formatted file represents a section of data to be inserted into the consolidated database for the event. A final Perl-based script takes the CSV-format files and inserts the decoded data into the appropriate table within the consolidated database.

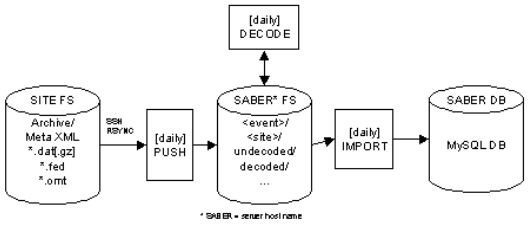


Figure 2. Data Staging Process

Database Engine Configuration Issues

Previously, the MySQL v4.0.18 RDBMS was selected for storing the decoded logger data for Post Event analysis operations. As a database engine, MySQL is

both an open-source and a commercial product line with a significant amount of engine performance tuning available for the end user to adjust based on specific needs. Through trial and observation, several adjustments to the database engine were decided on that would afford us the best performance for both the loading and the retrieval of data.

Database Table Configuration Issues

The MySQL database server engine supports several different table types, the default being MyISAM but provides support for BerkeleyDB, InnoDB, MERGE and MEMORY table types. It was decided to stick with the default type of MyISAM mainly for the fact that once data is loaded into a table, the data in the table becomes static and read only. Both BerkeleyDB and InnoDB table types are transaction safe, which for our purposes are not necessary.

One of the primary concerns for the table definitions revolved around the number of rows the tables will contain. During initial testing, it was discovered that the default number of rows that a MyISAM table could hold was less than the number of rows to be loaded. The overall data size for the table is determined by the types of fields used for the table and the data size for each type. Examples of this would be an INTEGER field type, which can have a data size up to 4 bytes and VARCHAR field type, which can have a data size up to the length of the text value + 1 byte. For the purposes of this experiment, there are three table settings that had to be set in order for the tables to scale to the number of rows anticipated. By adjusting the AVG_ROW_LENGTH, MAX_ROWS and ROW_FORMAT variables for MyISAM tables, it was possible to adjust the number of rows of data that the table can have. The ROW_FORMAT variable defines how the table rows should be stored. The option value can be FIXED or DYNAMIC for static or variable length row formats. When a table is defined that does not have BLOB or TEXT type columns, you can force the table format to FIXED or DYNAMIC with the ROW_FORMAT table option. This causes CHAR and VARCHAR columns to become CHAR for FIXED format or VARCHAR for DYNAMIC format. The AVG_ROW_LENGTH variable defines an approximation of the average row length for a table. This should be set only for large tables with variable size records. With a MyISAM table type, MySQL uses the product of MAX_ROWS times AVG_ROW_ LENGTH to decide how big the resulting table will be. If neither of these variables is specified, the default maximum size for a table will be 4GB. Overall, adjusting these variables are an absolute must to support the number of rows of data that have been observed for both the Near Real Time cache database and the Post Event consolidated database.

Because of the number of rows of data being stored into tables, it is imperative that efficient indexing be applied based on a thorough analysis of how data is extracted from the databases.

All columns used are not necessarily indexed, but only the columns that would enhance a typical query. As an example, within most tables there are VARCHAR fields that hold RTI-determined object name values. The object name in question, for the most part, uniquely identifies a specific entity within the simulation. Any column in a table that contains this type of data has an index applied to it because most of the queries posed utilize this column type as part of the qualifier of an SQL statement. Other indices are applied on a table-by-table basis within both the Near Real Time cache and Post Event databases geared towards their unique needs, but painstaking research went into selecting the most efficient usage of indices as part of each database schema creation effort.

NEAR REAL TIME PROCESSING

An example of one of the tools used for near-real-time analysis is the Track Matrix. The track matrix table provides a tabular snapshot of the current (based on the last 30-35 minutes of simulation time before the query is submitted) number of tracks associated with each type of entity. The row labels of this table are the actual truth types of entities being tracked at the current time. The column labels represent the perception of the entities being tracked. The column headings are exactly the same as the row headings because the set of possible perceptions is the same as the set of possible track types; Perceptions are determined by the SLAMEM simulation federate's sensor fusion center utilizing algorithms based on Bayes Rule. The resulting target type with the highest probability is the type associated with the track. The table entry in a given row and column is the number of tracks belonging to the corresponding row type that are perceived to be the given column type. A column labeled 'Ambiguous' indicates that those tracks are not resolvable. This means that the sensor fusion process determined that two or more target types were equally probable as the type of target being tracked.

Subsequent details associated with the Track Matrix are a series of additional tables and graphs segregating tracks into their ages, which is defined as the length of time between when the track was created and the last time it was updated. By segregating tracks by their

age, it is possible to get a sense on how well sensors and, in some cases players, are aware of the entities being played within the simulation. Older tracks can be perceived as having a higher probability of positive identification as opposed to tracks that persist for a shorter amount of time.

POST EVENT PROCESSING

In accordance with numerous authorities, the highest-level decomposition of the Post Event Processing system was into a single control class, entity classes, and an interface class (inasmuch as the interface class was a straightforward application of Microsoft Foundation Class (MFC), it will not be discussed). There are three general entity classes, called Database, Processor, and Final_Results. These roughly correspond to a traditional functional breakdown into input, transformation, and output. To promote the greatest possible generality, interactions between the Database class and the other classes were performed using Open DataBase Connectivity (ODBC). The Final_Results class encapsulates Microsoft Excel or a commercial graphical package called ChartDirector. Communications to and from that class uses either Microsoft's OLE Automation or ChartDirector's API. The Processor class, as well as most of the infrastructure of the system, was written using C++.

The post processing system comprises eight overall functional areas, all invoked by the user. These functions are as follows:

1. A Killer/Victim (K/V) Scoreboard,
2. A Killer/Victim details display,
3. An Entity Life Cycle summary screen,
4. An Entity Details Display,
5. A Sensor/Target (S/T) Scoreboard,
6. A Sensor/Target details display,
7. A Track Perception Matrix, and
8. A timeline (String) depiction that displays, graphically, the events in the lifecycle of any specific entity.

K/V Scoreboard
As currently coded, the K/V Scoreboard is produced by querying for the number and enumerations of all killers and victims are obtained via simple SQL queries. For each possible combination of killer and victim, the Damage Assessment interaction is queried to obtain the relevant victim's state. This is recorded, along with the entity causing the damage. Summations are performed by type (as indicated by enumeration values). The final results are presented in the form of an Excel spreadsheet or comma-separated value flat file.

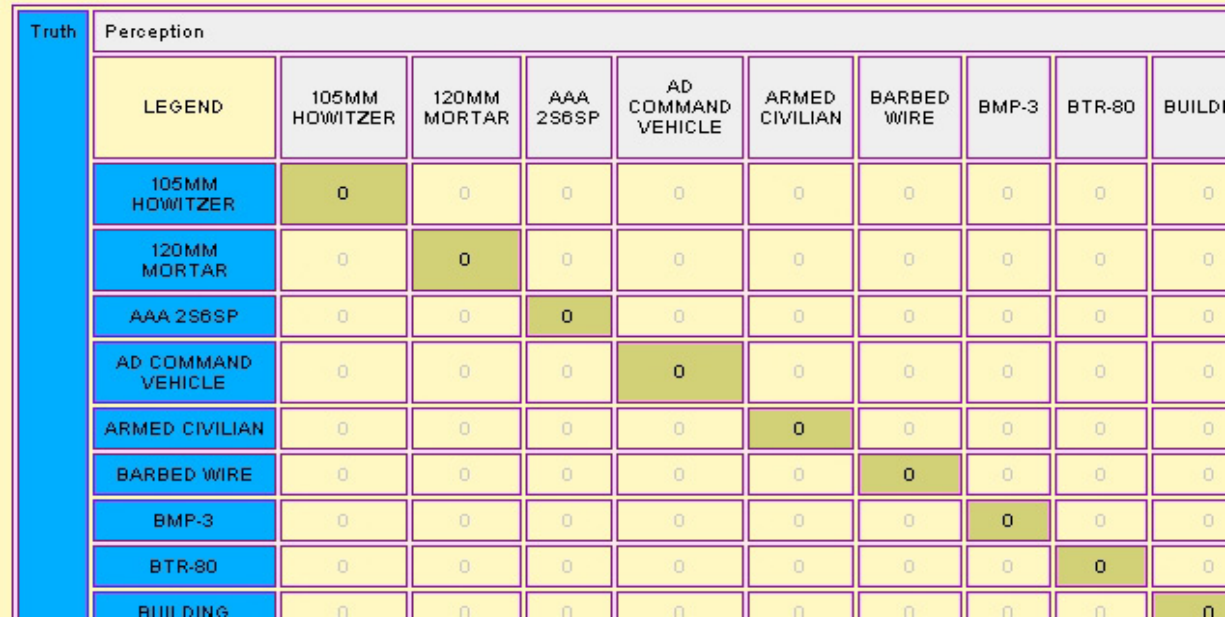
K/V Details Display
To obtain the details of any Killer/Victim interaction, the user is first presented with a screen enabling him to choose a particular killer and victim. Queries are performed against a lookup table to transform these English names into enumerations. An SQL statement is then constructed and executed that extracts the relevant fields from the Damage Assessment interaction.

Entity Life Cycle
The Entity life cycle summary output is derived largely from the entity state objects. All the entities used in the execution are gathered together into a vector. For each entity thus obtained, its entity state object is queried to obtain the fields necessary to compute its final state. This state is then determined and added to a running total.

Entity Details Display
Entity life cycle details are obtained from numerous objects. The user first chooses an entity via a series of drop-downs. To obtain the entity's state changes, its entity state objects are scanned for all records indicating a change of state, whose details are then recorded. The appropriate objects are then queried to ascertain the entity's creation and deletion details; data on sensor hits and weapon fire events, and detonations occurring on or near the entity.

S/T Scoreboard
The Sensor/Target Scoreboard summary output is similar in structure and layout to the Killer/Victim Scoreboard with the exception of the data obtained for the matrix display. For each possible combination of sensor platform and detected target, the Contact Report interaction is queried to obtain the relevant information concerning the detected target and the functional mode the sensor used to interrogate the target. Summations are performed by sensor platform and by sensor mode (as indicated by enumeration values) with the final results being presented in the form of an Excel spreadsheet or comma-separated value flat file.

S/T Details Display
To obtain the details of any Sensor/Target interaction, the user is first presented with a screen enabling him to choose a particular sensor platform and detected target. Queries are performed against a lookup table to transform these English names into enumerations. An SQL statement is then constructed and executed that extracts the relevant fields from the Contact Report interaction.



Truth	Perception								
	LEGEND	105MM HOWITZER	120MM MORTAR	AAA 2S6SP	AD COMMAND VEHICLE	ARMED CIVILIAN	BARBED WIRE	BMP-3	BTR-80	BUILDI
	105MM HOWITZER	0	0	0	0	0	0	0	0	0
	120MM MORTAR	0	0	0	0	0	0	0	0	0
	AAA 2S6SP	0	0	0	0	0	0	0	0	0
	AD COMMAND VEHICLE	0	0	0	0	0	0	0	0	0
	ARMED CIVILIAN	0	0	0	0	0	0	0	0	0
	BARBED WIRE	0	0	0	0	0	0	0	0	0
	BMP-3	0	0	0	0	0	0	0	0	0
	BTR-80	0	0	0	0	0	0	0	0	0
	BUILDING	0	0	0	0	0	0	0	0	0

Figure 3. Track Perception Matrix

Track Perception Matrix

The Track Perception Matrix summary output is designed to show information concerning simulation Tracks and how they are being perceived by the sensor model being used by the federation. For each possible combination of true entity types (truth guise) and perceived entity types (perceived guise), the Track and Track_probabilities interactions are queried to obtain the relevant information used to generate the display. The display consists of column headings representing the perceived guise possibilities, row headings representing the truth guise possibilities, and a diagonal across the table which represents where the truth guise and perceived guise intersect. See Figure 3 for an example section from a Track Matrix. The row and column heading extents are determined in advance by aggregating entity types together via a lookup table. The information displayed is then available for export to either an Excel spreadsheet or comma-separated value flat file.

String Chart

The String chart requires much of the same data as contained in the Entity Life Cycle details screen, and therefore uses the similar algorithms to gather data. However, instead of sending the results to an Excel spreadsheet, the data is fed to a commercial graphing product (ChartDirector). This product produces a timeline whereby events are depicted as color-coded icons. Placement of the icons at different y-axis values depicts the different events. These are placed in proper time order, with the x-axis showing wall clock time.

The String chart allows the user the choice of displaying the requested information in three possible ways with the data segregated into four sections:

- Entity Event,
- Blue Activities,
- Track Events, and
- Sensor/Target Events.

The Track Events and Sensor/Target Events sections are also segregated to show all instances of individual track numbers and sensor name/mode combinations or actual target type and bumper number. Each section or subsection is then sorted by time.

The Entity event section graphically depicts all changes that are derivable from an examination of objects received via the High Level Architecture (HLA) federation. These include entity creation, entry into damage states (firepower, mobility, firepower and mobility, or total destruction), moves and stops, and entry into or exit from camouflage. This section also records instances of the entity firing its weapon, receiving incoming fire, being deleted, or being recreated after a deletion.

The Sensor/Target section can depict two types of information. If the entity in question is being sensed, it will contain a comprehensive display of all the sensors that "saw" the entity, the sensor's mode, the highest

acquisition, the highest correct perception, the perceived type, and the time of detection. If the entity was within the footprint of the sensor but was not detected, a brief explanation of why is given.

If the entity in question is itself a sensor, the chart displays a listing of all entities that were detected, their actual type and bumper number, the highest acquisition, the highest correct perception, the perceived type, the sensor mode employed, and the time of detection. As before, if the entity was within the footprint of the sensor but was not detected, a brief explanation of why is given.

The track events sections show all tracks associated with the entity. For each track, a listing of the top three most probable entity types is given, along with the computed probability of the entity being of that type and the number of sensor hits used to determine it.

The blue activities section is reserved for human actions, such as planning a mission, assigning it a priority, initiating an attack or mission abort, requesting a bomb damage assessment, etc. All such user actions are sorted by time.

In addition to the pictorial generated by ChartDirector, a file containing the corresponding raw data (in either CSV flat file form or as an Access database) is also generated to allow the analyst to examine the information used to generate the picture directly.

CHALLENGES

General Gordon Sullivan (Chief of Staff of the United States Army, 1991-95) once said, "You don't know what you don't know," a statement that accurately describes today's challenges in mining extremely large databases.

Some of the challenges under initial assessment by the FAARS team:

- Policies and procedures for allowing interested government agencies access to the data generated by the URBAN RESOLVE experiment.
    - The data storage server is connected to the DREN. *What is the best approach for allowing others on-line access while minimizing the impact on the UR data collection and analysis effort?*
    - *Will the answer be purely policy driven or can software and hardware solutions enable the simultaneous utilization of the database?*
- Less than twenty percent of the collected data is of primary importance to the data analysts, at present.
    - *What impact will occur when the remaining eighty percent of the data is transformed through the data staging process and available on the terabyte storage device?*
    - *Will new techniques be required?*

CONCLUSION

The discovery process is not solely a characteristic of the joint experiment, but touches many aspects associated with the experimentation effort. As Joint Forces Command and the Joint Advanced Warfighting Program at IDA demand more from the simulation community the ripple affect moves throughout the various federates providing support.

In this specific case, the data collection and analysis effort has met the near term challenges brought about by an experiment scenario that requires over one-hundred-thousand entities; 1.8 million buildings and man-made urban structures to set the stage for achieving situational awareness in the 2018 timeframe. Use of hundreds of scalable parallel processors each logging the data generated during run-time was the impetus for the FAARS effort.

New challenges that have arisen since PART I was published will pale as we contemplate the challenges that we face for the upcoming year's trials. The good news story is the FAARS team, in fact the whole M&S team in J9, will continue to meet and overcome whatever challenges arise, and who knows, there might be another chapter awaiting...

REFERENCES

Dehncke, Rae W., Graebener, Robert J. 2004. *Urban Resolve: Joint Experimentation Raises the Bar for M&S.* Orlando: IITSEC 2004 Paper.

Graebener, Robert J., Rafuse, Gregory, Miller, Robert, and Ke-Thia Yao. 2003. *The Road to Successful Joint Experimentation Starts at the Data Collection Trail*. Orlando: IITSEC 2003 Paper.