Adapting to Urban Warfare

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

Mark Torpey
Lockheed Martin
Burlington, MA
mark.torpey@lmco.com

Urban operations currently are of great concern to the defense community. J9, the Experimentation Directorate of USJFCOM, and the Joint Advanced Warfighting Program currently are conducting an experiment to investigate concepts for applying future technologies to joint urban operations. The first phase of the experiment focused on employing future sensors to remotely monitor and understand enemy operations in a foreign city. Characteristics of the urban environment include high building density, a large civilian population, and a cultural environment. These characteristics pose significant challenges for simulation designers. This paper describes the modifications required to adapt the simulations supporting the experiment, JSAF and SLAMEM, to the urban environment. A landscape with a large number of buildings had to be automatically generated and represented in a space efficient manner. Large concentrations of vehicles and pedestrians had to be modeled moving realistically through the city. This behavior had to be automatically generated since it would be impossible to individually control 100,000 entities. Embedding cultural features within the database in the form of building function codes allow civilian entities to automatically plan their movements based on generic daily schedules. Sensor models had to be modified to detect building properties, such as whether a building was fortified. The density of both entities and structures made both movement and intervisibility calculations significantly more expensive, requiring optimization combined with the application of large amounts of hardware. Computation and control was distributed between three CONUS sites and the High Performance Computing Centers at Maui and Wright-Patterson AFB. Limiting and balancing simulation traffic required a major effort. Source squelching was enabled by a distributed data collection system developed to collect data locally on each simulation node while still allowing analysts to perform real-time queries during the experiment.

Keywords: Urban warfare simulation, data distribution management, distributed logging, Urban Resolve, source squelching, situation awareness, sensor simulation, clutter, SLAMEM, JSAF, urban terrain databases

1. Background

Urban Resolve (UR) is a three-phase experiment designed to explore new approaches for urban combat in the 2018 time frame. The U.S. Joint Forces Command (USJFCOM) J9 Directorate and the Joint Advanced Warfighting Program (JAWP) at the Institute for Defense Analyses are conducting the experiment. The hypothesis is that future sensors will provide a significantly higher degree of situational awareness than we currently have, allowing us to conduct urban operations more effectively and efficiently. The first phase of Urban Resolve focused on exploring the level of situational awareness that could be achieved using sensor capabilities, which we believe will become possible in the next decade. A team of subjects used simulations of these sensor capabilities to monitor enemy activity in the urban Area of Interest (AOI) shown in Figure 1; building up knowledge of the enemy's positions, assets, and activities. Situational awareness is considered to have three levels: perception, understanding, and prediction. The experiment evaluated how well the test subjects were able to perform at each level and how they employed the sensor resources provided to them. Their capabilities were explored by varying the sophistication of enemy countermeasures and the sensors available over a set of seven trials.

Urban Resolve is a discovery experiment conducted at three levels of abstraction. Concepts are developed in workshops and wargames. Then they are refined via constructive simulation, in this case the SLAMEM[1] simulation running closed loop. Specific scenarios are tried out using a human-in-the-loop (HITL) simulation, composed of the federation of JSAF[2] and SLAMEM. Each level explores the key areas required to implement new urban warfare concepts in greater detail.

The HITL component of the experiment is being conducted in the Distributed Continuous Experimentation Environment (DCEE). The DCEE

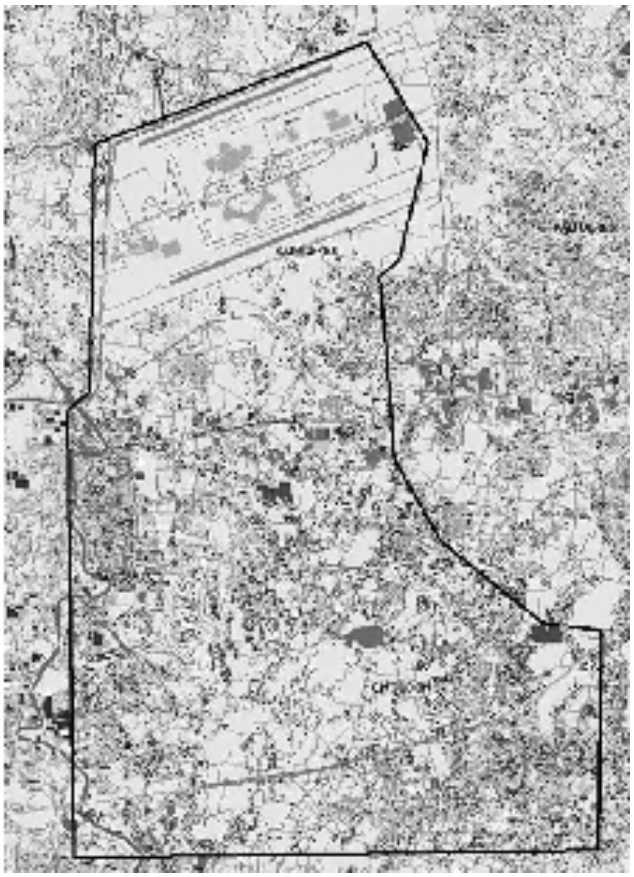

Figure 1. Urban resolve AOI

[1] was set up by J9 in 2003 to enable an increased pace of experimentation by providing dedicated space, hardware, and network facilities. Urban Resolve takes full advantage of these DCEE facilities. Urban Resolve was enabled by the Joint Experimentation on Scalable Parallel Processors (JESPP) project [2], which developed ways to use scalable parallel processors (SPPs) to perform large scale entity level simulations. Urban Resolve also takes advantage of the High Performance Modernization Program's resources. Initial development started on the Huinalu Linux cluster at Maui. For the execution of the trials, the Koa cluster at Maui and the Glenn cluster at Wright-Patterson AFB were used. The Urban Resolve experiment builds on the experience of previous experiments at J9, especially the J9901 Experiment [3] where JSAF and SLAMEM were used in a similar configuration to study the potential of future sensors and standoff weapons in a non-urban environment.

2. The Experiment

In Phase I, the experiment's subjects manned a Joint Intelligence and Fusion Cell (JIFC) as shown in Figure 2. The JIFC was responsible for monitoring activity in the AOI, which covered roughly 100 square kilometers. The area included an airport and a number of high-density urban centers as well as lower-density intervening areas. The JIFC had access to an array of high and medium altitude unmanned aerial vehicles (UAV), low altitude organic air vehicles (OAV), unattended ground sensors (UGS), and human intelligence (HUMINT) reports. Of these, the OAVs and the placement of the UGS were under the direct control of the subjects. The other sensors were controlled by the White cell and the subjects had to request sensor coverage from those assets. These platforms carried a wide variety of sensors that could be brought to bear on the city. All the sensors and sensor platforms were modeled in SLAMEM with the exception of the HUMINT agents who were represented by intelligent Soar agents in JSAF [4,5].

The target was an insurgent Red force that had taken control of the country from its elected government and was moving into the city to hide its forces and equipment in among the urban population and sensitive sites. Its infiltration into the city was peaceful as a significant minority of the population supported it. As the forces came into the city, they established fortified positions and antiaircraft sites. They also hid medium-range ballistic missiles and weapons of mass destruction. Life in the city proceeded normally during the infiltration, as the Red forces needed normal traffic to mask their movements and operations. The Red force was controlled by a team of 10 Red operators situated at the Topographic Engineering Center (TEC) at Ft. Belvoir, Virginia. Their operations were controlled by a Red leadership team, which was in turn controlled by the White cell. Although Red developed and proposed countermeasures to make Blue's job more difficult, the White cell decided which tactics Red was allowed to use. They were not considered experimental subjects. All the Red forces were simulated using JSAF taskframe entities.

A single operator located at the SPAWAR System Center in San Diego, California, controled civilian traffic. While the activities of the civilians were automated and guided by preplanned templates, at some points the operator was required to initiate the formation of crowds and the establishment of exclusion zones. The civilian population was modeled using Clutter, an offshoot of JSAF. Engineers monitoring the Linux clusters, data logging, and other simulation functions also were located at SPAWAR.

The experiment was supported by over 10 analysts and data collectors whose tasks included experimental design, monitoring activity in the Blue cell, logging and extracting data, and of course conducting the analysis. The analysts used the Future After Action Review System (FAARS) to extract the data required for analysis [6]. Figure 3 shows a result produced by the analysis team from Trial 3b. In each experiment, the Red forces

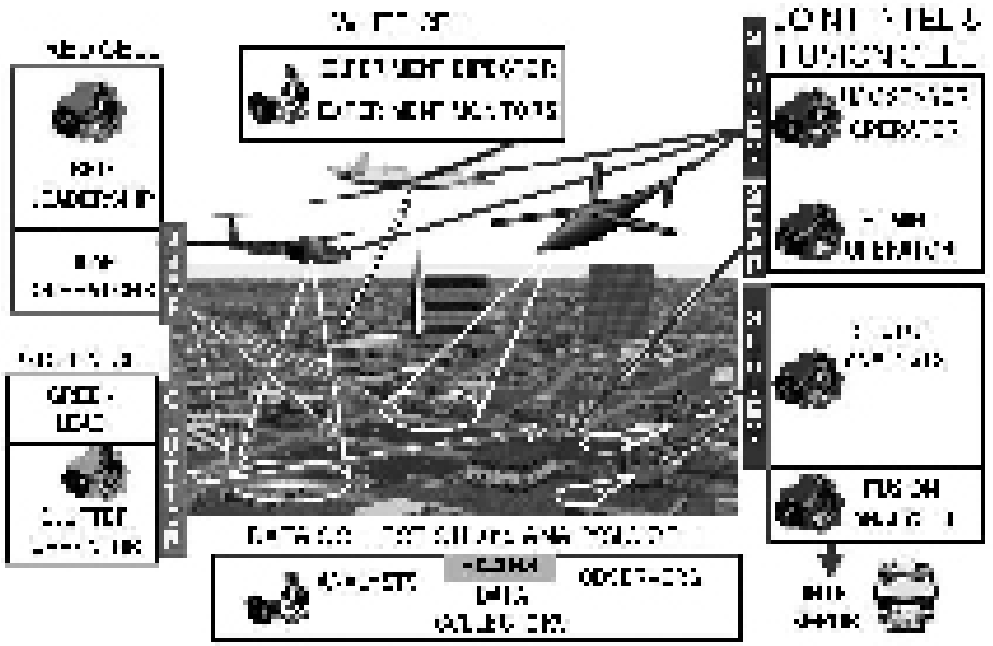

Figure 2. Urban resolve cell architecture

were required to perform a variety of activities, such as leadership meetings, positioning forces, and resupply. At the end of each shift the JIFC provided an intelligence report detailing the activities they had been able to discern and their estimate of why the enemy was engaging in those activities. In this trial the Red forces conducted a total of 177 distinct activities. The Blue cell was fully aware of 50% of those activities, partially aware of an additional 20%, and did not detect the remaining 30. The left-most column in Figure 3 shows the results for all activities while the remaining columns break down the activities by type of Red unit.

The Blue, White, and Analysis cells were located at the J9 DCEE facility in Suffolk, Virginia, and technical control was located in the adjacent J9 Simulation and Analysis Center. Coordination between all sites has been significantly simplified by the use of the Marconi ViPr video conferencing system.

Preparation for the experiment started in September 2003, with execution starting in June 2004. Four two-week trial periods were held from June to October. The first trial period was used for a single trial while the remaining periods were divided into two trials each. Preparation for the HITL experiment included a requirements conference, seven one-week developer testing events, two preliminary trials, and a three-week player training session. As is typical in this type of experiment, requirements evolved continuously throughout the development period as JAWP personnel assembled the sensor characteristics and experimented with the C4I displays. The preliminary trials allowed White cell personnel to man the Blue cell and learn how to perform the tasks the players would be asked to carry out. These trials produced many improvements in the system design.

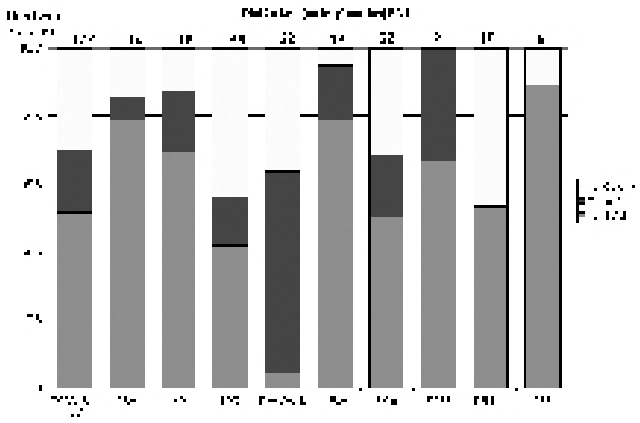

Figure 3. Trial 3b — level one situation awareness

3. Modeling

The key features of the urban environment for the first phase of the experiment were the dense building environments that occluded sensors and made it difficult to track enemy movements and the civilian population, which Red used to mask its movements. Incorporating these features into the federation caused significant computational loading requiring both optimization of the simulation software and distribution of the computation over many computers including scalable parallel processors.

3.1 Urban Terrain

The data for the buildings, roads, and rivers was obtained from a commercial source. It was combined with additional coastline, feature, and elevation data [7]. The building data included footprints for approximately 1 million buildings. However, it did not cover the entire city. This geospecific data was used to create additional geotypical data to fill in the the missing areas and produce a final database of over 1.8 million buildings. Figure 4 shows the building coverage of the original source data while Figures 5-7 show the resulting terrain database, both from the perspective of an operator at a JSAF plan view display (PVD) and from the perspective of a 3-D viewer. The outline of the AOI is shown in the upper left of Figure 5. Manual terrain intensification was used to add foliage and smaller objects such as dumpsters and jersey barriers to the AOI. The target databases were the JSAF Compact Terrain Database (CTDB), the 3-D Visual Database, the dynamic terrain database, and the SLAMEM terrain database. The first three are all constructed via a common process that ensures that they are fully correlated. ArcView Shape files of the linear and areal features were imported into SLAMEM to provide correlated building footprints, roads, and rivers. All the intervisibility calculations are performed in JSAF mitigating issues of terrain correlation between the two simulations.

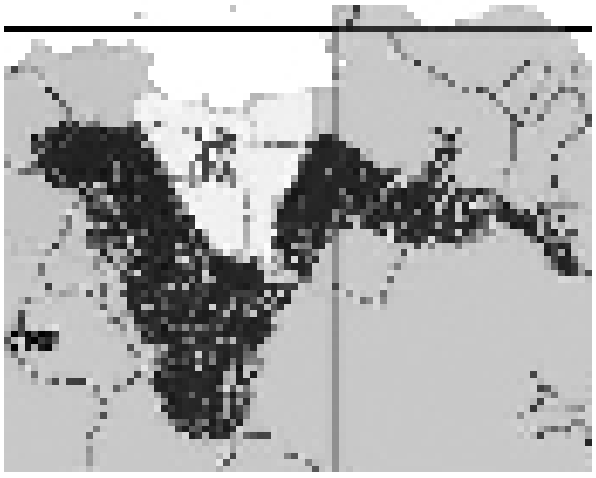

Figure 4. Original building data

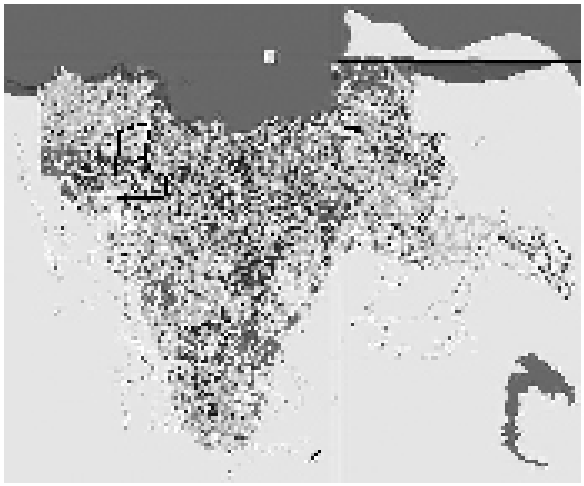

Figure 5. Filled in urban area

In addition to the building footprints, the source data contained some building names and attributes, but only for major buildings. To fill in the attributes for the remaining buildings, statistical models were developed based on norms for different land use areas. More systemized versions of these techniques are described in Adelson [8]. Although the focus of the experiment was only a small AOI, the database was built prior to this experiment's design as part of our research on using SPPs for Joint Experimentation. Furthermore the urban area is embedded in a low-resolution worldwide terrain database that includes the entire world between 75 degrees north and 75 degrees south latitude. To represent this terrain we had to modify the CTDB format; the experiment uses CTDB version 8.7. While there were changes to many areas of CTDB [9], changes to reduce the size of multiple elevation structures (MES) were especially important.

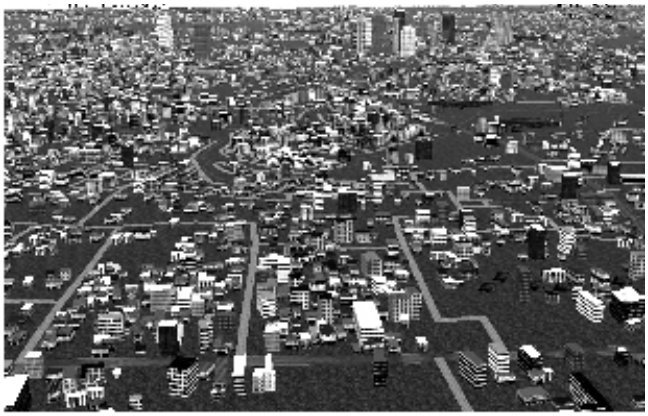

Figure 6. Aerial view of AOI

There are two different building representations used in the database. First, there are buildings without interiors, whose walls are extruded from the footprint data. These are extremely space efficient. The second are multiple elevation structures that include interiors, floors, doors, and other features. The MES format uses a great deal of space and the experiment's AOI contains around 47,000 MES buildings while a second area contains 18,000. The terrain database requires approximately half a gigabyte of disk space for the urban area and one gigabyte for the remainder of the world. JSAF memory maps the terrain into its address space allowing Linux to do the actual paging of the terrain data into memory. Since 1.5 gigabytes is a large fraction of a 32-bit address space, a meta-paging scheme in JSAF dynamically maps and unmaps terrain cells from the address space.

Figure 7. Street level view of AOI

3.2 Civilian Population

The civilian population provides the environment for hiding Red movements. In previous experiments we modeled civilian traffic with JSAF entities called clutter. These low resolution entities drove around the road network along randomly generated routes and occasionally parked. However, for this experiment, we needed traffic whose density would vary throughout the day and would move around the city in a purposeful manner so that it would be hard to distinguish from the Red traffic trying to blend in with it. We created a new class of clutter called commuter clutter for this purpose [10]. These clutter entities use timetables to control their movement. Each clutter template (scenario) is provided with a timetable of activities. These activities include going to work, going out to eat, going home, and so on. For each activity listed, a time range is specified and clutter entities randomly pick a time within the range to perform the corresponding activity.

Although modern terrain databases provide highly detailed models of buildings, interiors, trees, vegetation, and weather effects, they are behaviorally barren. In the "The Sciences of the Artificial," Simon [11] observes that the complexity of an ant's path arises from its environment. So it is with most everyday activities. Our previous buildings provided cover, concealment, and obstacles for combat operations, but did not serve any civilian purpose. If an activity called for a clutter entity to go and eat, there was nowhere to eat. To deal with this problem, we attributed all of the buildings in the simulation with building function codes (BFC), including homes, restaurants, houses of worship, offices, stores, etc. (see Figure 8). Although this scheme is simple, it provides the cultural environment necessary to drive commuter clutter. When each clutter entity is created, it selects a nearby building with an appropriate building function code as its home; it selects another building with an appropriate function code as its workplace. When the timetable calls for the entity to go to work, it plans out a route to go to its chosen workplace. If some of the streets are blocked with exclusion zones, the entity finds a detour. When the entity's timetable says it is time to go to a restaurant, it randomly selects a nearby restaurant and drives there. Upon arriving, the vehicle parks and the occupant gets out of the car and walks into the building. This simple timetable approach allows us to realistically model the ebb and flow of traffic as the day progresses. The experiment used approximately 100,000 clutter entities in or near the AOI. This included roughly 25,000 pedestrians and 75,000 vehicles.

3.3 Red Forces

The Red forces were represented primarily by JSAF taskframe controlled entities [4,12]. There were over 1,000 Red entities in the Urban Resolve scenarios. A key requirement was to have the Red forces blend in with the civilian clutter. To do this, the movement algorithms of the Red forces were modified to use the same logic and route planning as clutter when driving down roads.

Figure 8. Buildings colored by BFC classes

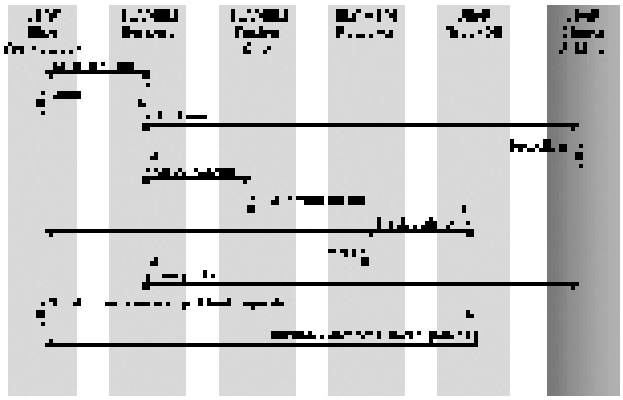

Figure 9. Sensor control, interactions, and data flow

When driving off road they revert to the taskframe JSAF movement algorithms. Thus Red road movement is indistinguishable from that of the civilian vehicles. The complexity of planning off-road movement in urban areas overwhelmed the route planner traditionally used by JSAF. To deal with all the building obstacles we changed from using a vector-based planner to a grid based planner based on the A* planner used by DISAF [13,14]. This planner also more realistically spreads out planning over the course of multiple entity updates. The planner was optimized to reduce its memory and processing requirements so that it would perform adequately in the complex urban environment.

The Red forces needed to modify buildings to create fortified positions. External barriers were modeled as entities and placed around fortified areas. To avoid making these prepared positions too obvious, jersey barriers and other construction obstacles also were placed around the AOI in unfortified areas. To represent internal fortifications, we used building properties. An editor was developed that works with the dynamic terrain federate to modify building properties on each federate's copy of the terrain database. This editor allows the Red cell to dynamically mark buildings as modified, fortified, and occupied. Using appropriate sensors, the Blue players can detect the state of these building properties via a modified version of the distributed sensor proxy. This federate responds to sensor footprints by returning information about buildings instead of entities.

The Red Operations Cell was designed to provide physical separation between the Red leadership and the Red subordinate unit commanders. While this arrangement was not critical for this Phase of Urban Resolve, it will be for Phases II and III where a major objective of the Blue force will be to cut off and isolate different units of the Red force. Phase I provided an opportunity to experiment with the implementation of this concept. Red was not allowed to have a shared electronic common operational picture and the leadership's operational picture had to be based on reports from subordinates. Xchat was used as a low-cost solution to modeling communications between the Red forces. Chat channels were set up to replicate all the communications mechanisms available to the Red forces. For radio messages, operators used JSAF to send out the appropriate radio signals to the federation for interception by ELINT[3] sensors and then used Xchat to send the text message to the appropriate Red leader. The text messages were available to the White cell for generating intelligence reports to the Blue cell and were recorded for use by the analysis team.

3.4 Sensors

Sensor modeling is performed by SLAMEM using confusion tables to model the classification and identification process. A confusion table is a matrix (see Table 1) whose rows correspond to the ground truth types of the entities being sensed and columns correspond to the possible identifications that can be assigned to the detection. Each cell of the matrix gives the probability that a sensor analyst will classify or identify the target as the type at the top of the column given its ground truth identity. For example, the center cell in Table 1 is the probability that a sensor analyst will recognize a truck correctly. Each row of a confusion table gives the probability mass distribution for all the possible things that a ground truth entity could be identified as, including unknown. The ground truth row categories must be mutually exclusive but the column headings can overlap allowing a Ford truck to be classified either as generic truck or more specifically as a Ford truck.

The operators control SLAMEM platforms via an editor in JSAF. SLAMEM (see Figure 9) receives commands from the editor and models the positioning of the platforms and the orientation of their sensors. It translates the field of view of the sensors into footprints and sends those footprints to JSAF and clutter. Their entities respond by calculating whether they are within the footprint and whether they have a clear line of sight to the sensor. If so, they respond with a detection report to SLAMEM. These reports are misnamed as they indicate that the entities can be detected, not that they have been detected. SLAMEM rolls the dice to determine which of the entities returning detections are actually detected. Entities that are detected are then

	SA-15	Truck	Fuel Truck
SA-15	.7	.05	.25
Truck	.1	.6	.3
Fuel Truck	.2	.3	.5

Table 1. A simple confusion table

fed through the confusion tables to determine their perceived classification. There are confusion tables for different types of sensors, different sensor ranges, and different times of day.

Once a detection has been classified, the SLAMEM fusion center associates the detection with an existing track or decides to create a new track. At this point SLAMEM uses Bayes Theorem [15] to compute a posterior probability distribution for the target's ground truth identity. Each track maintains an associated vector containing these posterior probabilities. For example, for the confusion matrix in Table 1, the vector would contain the probability that the track is an SA-15, the probability that it is a fuel truck, and the probability that it is a truck. This probability vector is updated as each new detection is associated with the track and usually converges toward the true identity of the track. The tracks are held by the JSAF track database, which sends track updates out to the Blue workstations for monitoring. The tracks also are used by SLAMEM to perform automatic retasking of sensors to maintain existing tracks on interesting targets.

3.5 Blue C41

Each of the Blue players is given the suite of equipment shown in Figure 10. The suite includes a JSAF workstation that serves as their primary C2 system. On one screen there is a 2-D tactical map of the area, including a display of the positions of their sensor assets, the fields of view of their sensors, the resulting tracks and their histories. A second screen provides editors for examining tracks, commanding assets, and controlling what is displayed on the map. The Blue players translate the sensor data presented on these screens into a picture of the situation by creating situational awareness objects that indicate what they believe the enemy is doing. A third screen provides access to the InfoWorkspace (IWS), ForceView, and a survey application. IWS is a collaboration tool for interplayer communications. ForceView is an experimental C4I system that displays a 3-D view of the AOI. The survey application is used to collect subjective analysis data from the players.

4. Computational Architecture

Due to the large numbers and high densities of buildings and civilian entities, significantly more computer power was required to run this experiment than previous ones. We attacked the problem through both the optimization and distribution of the computations. Some of the optimization highlights include modifying our intervisibility calculations to avoid premature sorting, adding a maximum structure height in the terrain patch headers to avoid unnecessary checking of building obstructions, and ignoring building interiors for external intervisibility calculations. These and other changes made a significant difference in the number of entities we could simulate. Prior to the changes, the 933 MHz nodes on the Huinalu cluster were being overwhelmed by the flood of intervisibility calculations required to process SLAMEM sensor footprints and could only simulate several hundred entities without overloading. After these optimizations, they could simulate several thousand.

4.1 Distributing SLAMEM

We first used the distributed sensor protocol to distribute some of SLAMEM's calculations in 1999. It allows JSAF processors to do the intervisibility calculations while SLAMEM performs the detection and identification portions of target acquisition [16]. However, due to the higher density of entities in this experiment, a single SLAMEM processor was unable to keep up with the load. The sensor entities were distributed to multiple SLAMEM federates, which modeled their movement, sensors, and detection processing. The resulting classifications are sent to another SLAMEM federate (see Figure 9), the fusion center, which produces tracks and track updates. A separate JSAF track database holds the track state and publishes it to the rest of the federation. Yet another SLAMEM federate, the retasker, controls the automated retasking of sensors to maintain the current tracks. A total of 11 computers were used to run SLAMEM.

4.2 Distributing Clutter

Initially, we required over 100 SPP nodes to simulate clutter. After optimization, the number of nodes required was reduced significantly, but we kept using all the available nodes as a safety factor. It also allowed us to experiment with the use of SPPs at multiple widely-distributed sites. The key scalability concept for clutter has been to limit its subscriptions to the minimum amount of information possible, such as detonations and sensor footprints. In particular, clutter did not subscribe to entity updates, so clutter entities have been ignorant of traffic from other simulations and simply drove through each other. For this experiment, we sought to create more realistic clutter movement without the loss of efficiency that listening to other entities would entail. Because each clutter simulator has thousands of entities that can each decide to travel to any part of the AOI, using geographic filtering to limit subscriptions is not effective. The approach developed to control clutter movement is based on deterministic algorithms replicated on every simulator modeling entity road movement [10]. Each intersection has a

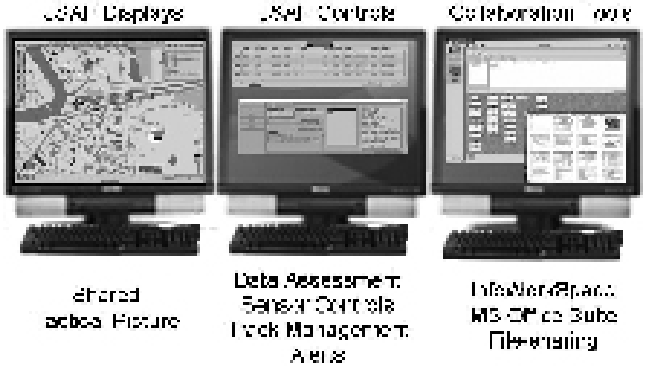

Figure 10. Blue cell C41 workstation

traffic controller for all movement into that intersection. Each vehicle pulling onto a road connected to that intersection sends a message to the traffic controller to register with it. The controller then sends commands to the entity to control its movement as it approaches the intersection and goes through it. The controller takes care of multiple traffic lanes and pedestrian traffic. Because this control algorithm is replicated on each node, each entity receives its commands from the local copy of the intersection controller. However, each copy of the controller listens to all the registration messages for that intersection from the entire federation. The magnitude of this control traffic is approximately 10% of the total entity updates. An alternative approach would be to migrate entities so that all entities controlled by a particular intersection would be on the same computer. However, implementing migration would have been significantly more complicated and potentially could cause computational hot spots if many clutter entities and sensors converge in the same location.

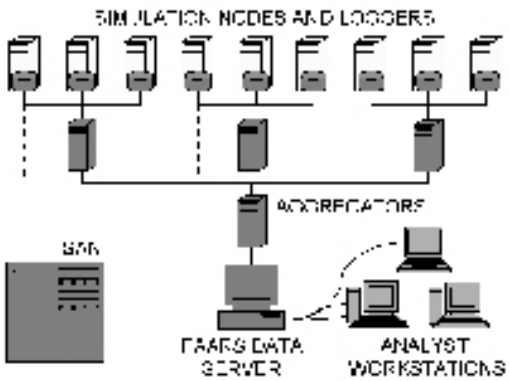

Figure 11. Distributed data collection and analysis

4.3 Source Squelching

Because of intersection control, entities now are constantly starting and stopping as they make their way through the city. In addition, the road network is much denser and full of turns and intersections than previous databases. This makes entities break their error thresholds much more frequently, producing significantly higher update rates and bandwidth requirements than in previous experiments and exercises. In addition to entity updates, our bandwidth budget also must support the intersection control logic interactions, the constant sweeping of sensor footprints, the returning detections, as well as track updates. With widely distributed sites, the challenge was to balance the network traffic with the available bandwidth. Early network measurements showed that 5,000 clutter entities generated approximately 7.5 Megabits/sec (Mbps) of network traffic. Extrapolating to 100,000 clutter entities the network bandwidth required would have been 150 Mbps just for clutter updates alone. Since our wide area network links only were capable of supporting around 50 Mbps, we clearly could not put all the traffic out on the network.

Runtime infrastructures (RTIs) have long had the capability to perform source squelching. That is to turn off the publication of simulation data that no one has subscribed to. However, it has been very hard to take advantage of this capability in practice. Most simulations are not significantly distributed, so they do not benefit from data distribution management (DDM) and simply use class based subscriptions; listening to all the instance updates for each class of federation object. The worst offenders are loggers that typically try to record everything and white cell displays used by simulation controllers who want to see everything happening in the simulation. These practices defeat source squelching. To make it work for Urban Resolve, we had to implement technical and cultural changes.

First we replaced our centralized logger approach with a distributed design. A distributed logger process runs on each node that has a federate producing simulation state updates. The logger process records any publications the federate produces directly to the local disk drive of the computer the federate is running on. It is implemented using an RTI intercept that allows the logger to catch any publications the application sends to the RTI irrespective of whether the RTI actually sends them out or not. Thus the logger is not a federate and does not generate any subscriptions. One of the advantages of a centralized logger is that the data is collected in a single database that can be queried while the experiment is running. To retain this capability, the distributed logger supports simple SQL queries while it is recording. A tree of aggregators (see Figure 11) takes

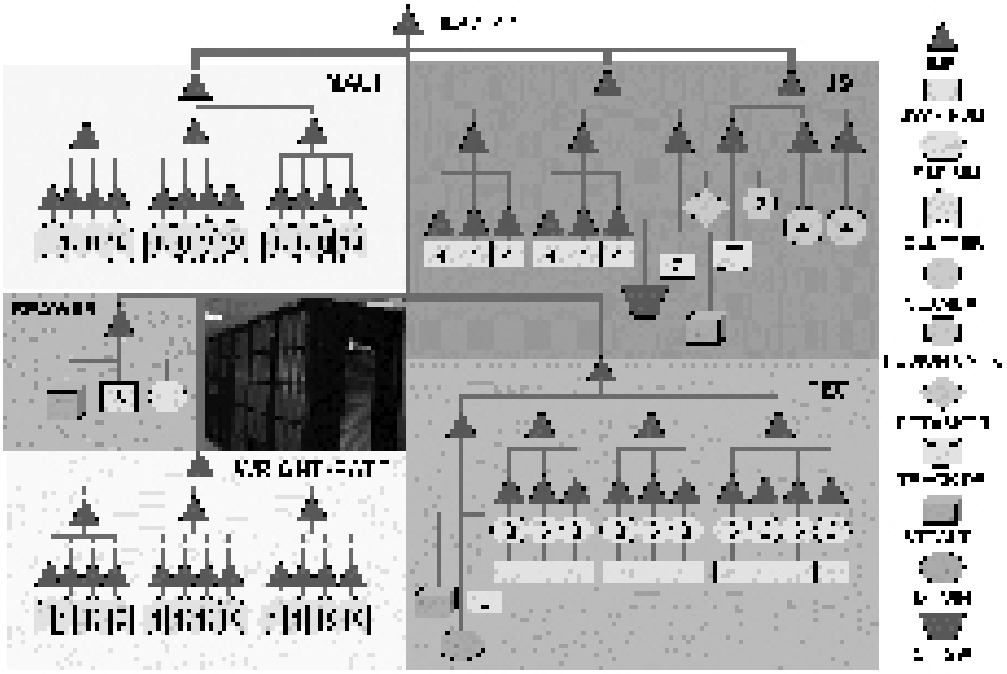

Figure 12. Urban resolve federation

a query and delivers it to all the loggers in the federation and assembles all the responses back into a single query response. Naturally, the volume of such queries needs to be limited or the advantages of source squelching will be lost. Each evening, after the conclusion of the experiment day, data from the loggers is downloaded to a Storage Area Network (SAN) data storage facility at J9 and is then inserted into a relational database [17].

The other major problem for source squelching is subscribers that can expand their view and subscribe to too much data resulting in a network overload. If a map display or PVD is subscribed to clutter entities, the operator can zoom out causing subscriptions for all the clutter entities to be turned on, forcing 150 Mbps of traffic to attempt to go through 50 Mbps of network bandwidth. To prevent this, we limited the scale at which operators could view individual ground entities to 1:5000 — a view covering several square kilometers. This was sufficient to maintain network stability while still allowing operators to perform their tasks.

Even with these changes, source squelching remains a fragile concept requiring the federation developers to carefully balance their bandwidth constraints with their federates' consumption. Occasional overloads can be handled by using rate limiting with prioritized message dropping. However, we found that once we had balanced the federation it was reliable and required no additional attention during the UR Phase I experiment.

4.4 RTI Transports

Just as we had to optimize our simulation algorithms, we also worked to optimize the transmission of simulation data. This was made easier by the RTI we used, RTI-s, which implements all the transport types internally rather than making each application implement them independently. This arrangement allows more flexibility in tuning the transports for a particular experiment. A variety of transports were used along with rules that would switch between transport types depending on the level of traffic generated. For example, an object using a minimum rate transport with a heartbeat of 30 seconds would automatically change to using a state consistent transport if its state was not changing for a while. Similarly the object could change to a faster minimum rate transport if required. Substantial effort went into tuning the transports for different objects and interactions.

4.5 Data Distribution Management

In our previous distributed simulation architectures, we implemented data distribution management using multicast addressing. Simulation state was divided into interest regions that the RTI associated to multicast addresses. Updates for each interest region were published to the corresponding multicast address.

Simulations requiring data from particular interest regions subscribed to the appropriate multicast addresses. All the simulation state was published to the network but individual simulations could listen to as much or as little of it as they needed. For Urban Resolve, we switched to a DDM implementation based on tagging simulation state updates with an interest attribute and using a network of interest management processors (IMPs) to route the information between federates [18]. Each federate is connected to an IMP and the IMPs are connected into a tree as shown in Figure 12. A federate's interests are expressed in a sparse bit vector. This interest vector is propagated around the tree so that each federate's output link knows what the listeners on the other side of the link have subscribed to and only those data items matching the subscriptions are sent forward. This implements source squelching both for simulation federates and the IMPs. So if Maui does not have any federates subscribing to tracks, no track data is sent to Maui. We located our federates so that track updates were confined to the J9 LAN while all Red forces control information was confined to TEC to minimize the traffic going over the WAN.

One advantage of this method of implementing interest management is the ability to have far more interest regions than are available using multicast addressing on an IP switch. High-end CISCO switches like the 6509 have a theoretical maximum of 7,000 multicast groups and the rate at which subscriptions can be changed is globally limited. We used 175,000 interest regions for Urban Resolve and a federate can change as many subscriptions as it wants at any time.

Another advantage of this approach is that in practice running multicast over WANs tends to be difficult. Since there often are multiple remote routers involved in the network path, it is difficult to get all of them configured properly and maintain that configuration over the course of the experiment. Additionally, since multicast routing is dynamically recomputed, it is not unusual to occasionally drop multicast routes leaving some simulation nodes without state updates. The best reliability we have achieved in multicast based events has been through the use of the MRouteD multicast tunneling program. We used it to connect together sites in the US, Canada, Germany, and Australia for the Multi-National III Experiment. That only worked reliably when the remote MRouteDs were connected in a star configuration to a central MRouteD in the U.S.

The IMPs allow you to set the transport protocol on each link to TCP, UDP, or rate limited UDP; point-to-point protocols that are easy to route around the world. The rate limited UDP transport allows you to prioritize which interest regions are dropped first. You also can enable compression on the links to reduce bandwidth. It took us about three test events to balance the traffic on the network. The biggest problem we ran into was overloading the output ethernet connections of the IMPs. Most of our connections were initially 100 Mbps full duplex and we would get bursts of traffic that exceeded this if an IMP had too many links. So we limited the number of links to around five unless the IMP had a gigabit connection. We also reduced the bursting problem by using TCP whenever the latency on a link was low. All links on LANs were TCP. For long distance links with high latencies, such as those to Maui and California, we used UDP because the TCP throughput was too low. We experimented unsuccessfully with different TCP window sizes to increase the throughput. When TCP is used on the WAN it is important to monitor network latency between sites because the route used can change at any time. At one point, our connection between J9 and TEC was rerouted causing a very high latency between the two sites. This reduced the TCP bandwidth severely and caused packet loss between the sites.

Another reason to use IMPs instead of multicast is that not all SPPs have IP connections between their nodes. Instead they may use another interconnect technology such as the message passing interface (MPI). An MPI transport layer was added to RTI-s to allow it to run on non-IP based SPPs. We developed two different types of IMPs: tree-connected and mesh-connected. For this experiment we used the tree-connected IMPs, because they where ready first and although the mesh-structured IMPs can provide more bandwidth with a favorable network topology, our underlying network architecture was tree structured, precluding those benefits.

4.6 Queries

A side benefit of having large number of interest regions was the number of RTI objects in any particular region is relatively small. This made it possible to use interest region-based queries frequently. Conventional class-based queries can be severely disruptive if the number of objects in a class is very large; a large network spike is created when they are used. Object queries are not particularly useful because you need to have discovered the object prior to querying. Interest region-based queries are much more practical because they can be used to rapidly discover objects when subscribing to a new interest region. Thus there is no need for the objects to continue publishing if they are not changing. It solves the late joiner problem as long as the late joiner does not subscribe to too much at once. It also reduces the latency between joining a new interest region and finding out about the objects in it.

The workhorse query that allowed our simulation to function was the footprint query produced by the SLAMEM sensors. Figure 13 shows rectangular footprints in the southern half of the AOI. Each

Figure 13. Footprint queries on JSAF PVD

footprint produces a virtual snapshot of the area under surveillance. This allows SLAMEM to avoid subscribing directly to entities, which would be disastrous given the wide coverage of all the sensors. The footprint query effectively is a one-time subscription with very high resolution. The disadvantage of using footprint queries for sharing all entity information is the additional latency of having to send out the query and wait for it to be processed. It is not appropriate for situations requiring fast responses. Also, if many sensors are viewing the same entities, footprint queries could cause an increase in network traffic.

4.7 Federation Configuration

Figure 12 shows the Urban Resolve federation configuration roughly as it was during the trials. The federation was constantly changing as the experiment developed and computation was redistributed. The head IMP was located at J9. It exceeded the five connection rule since it had a gigabit link. Each IMP was primarily limited by its connection to the switch rather than its processing load. The availability of more gigabit connections would have significantly reduced the number of IMPs required. Clutter was primarily simulated on the MAUI and Wright-Patt clusters. All Red entities were simulated at TEC to confine the control traffic to that site. The TEC machines were organized by operator with one IMP connecting the operator's JSAF PVD with his three JSAF simulators. All of the SLAMEM assets were located at J9 together with the track database and the Blue player workstations. This allowed us to keep the track updates local to J9. Figure 12 only shows the simulation machines. It does not show the aggregator hierarchy for the loggers, the data analysis hardware, the non-JSAF C4I equipment, or the networking hardware.

We made several attempts to run the Red simulator federates on MAUI. First, we simply ran the simulators on MAUI and the PVDs at TEC. The Persistent Object (PO) control protocol proved to be too verbose and fragile to support high latency links. PO currently is being replaced by another protocol, which we hope will be 90% lighter and more fault tolerant. It should also make it easier for other applications to control JSAF entities. We also tried placing both the PVD and the simulators at MAUI and using VNC[4] technology to allow operators at TEC to control them. Unfortunately, for rapidly changing displays VNC breaks down and reverts to a slow linear update.

It is not practical to configure and start up 200 to 300 federates manually, especially when attempting to run production trials. The chances for error are too high even if there was adequate time for a manual start. The addition of trees of IMPs and aggregators that need to be configured makes the problem even harder. To make experimentation practical, the UR federation relies on a system called MARCI[5], which allows a single operator to start, stop, and monitor the federation. It also distributes software updates to the federates. MARCI is capable of automatically generating connection maps for the applications on the SPPs, which makes it easy to deal with changes in the availability of the SPPs and their nodes [19].

5. Preparation for Phases II and III

The requirements for Urban Resolve Phases II and III continue to push the evolution of the Urban Resolve federation. While many of these changes are experiment specific, like new entities and weapons, others like the previously mentioned replacement of the PO protocol are significant advancements for the UR Federation. The following are some of the major changes occurring in the federation.

The ability to represent and display large-scale urban environments has led to collaboration between J9 and the National Geospatial Agency (NGA) to produce additional high-resolution urban databases for experimentation. J9 continues to evolve CTDB, to support better models of building damage, road and other linear networks, and building properties.

The intersection control logic has been shown to be an effective and low bandwidth approach to modeling city traffic. However, due to different latencies between clutter simulators, messages do not reach all the controllers in the same order and some messages, especially those going over the WAN may be lost. This causes the controllers to produce different results, which leads to anomalies in traffic behavior. One controller may have a vehicle move through the intersection while another assumes it will stop and wait. As a result, the use of replicated distributed controllers is at best an 80% solution. While this could be improved by the use of time management and timestamp order delivery, these solutions come with a price. Their efficiency is highest

in federations with few federates and large time steps. The UR federation has hundreds of federates and over 100,000 entities updated at up to 2 hertz. We currently are investigating an alternative approach of only assigning control of an intersection to a single clutter simulator. This eliminates conflicts, but adds latency to vehicle control commands. To overcome this latency we are using a new dead reckoning algorithm designed for road traffic. We expect the result will yield better road movement behavior and be more tolerant of latency.

While the use of clutter profiles and building function codes has significantly expanded our capability to represent the behavior of urban populations, we are working to federate several models of PMESII[6] phenomena with JSAF and clutter to improve the representation of human factors to our federation. This will allow us to use the Urban Resolve federation to explore effects based operations. One potential problem in incorporating some of these phenomena is that their time scales are longer than those typically played in a real time human in the loop simulation.

Since Urban Resolve Phases II and III will involve combat operations, we are enhancing our combat models to support more variables. Indirect fire projectiles are now flown out to determine whether they will hit intervening buildings prior to reaching the ground. The JSAF indirect fire damage model has been modified to take intervening buildings into account. For humans, posture now affects the likelihood of taking damage (e.g., prone soldiers are less likely to be injured from a ground impact explosion than those standing upright).

Our data collection and analysis system is being re-engineered to support distributed processing of queries. Queries will be evaluated as close to the data as possible. The goal is to reduce the network traffic required to query the simulation while it is running and make post-processing faster. This will free up more bandwidth for the simulation.

For Phase I of UR, a 2-D map display of the common operational picture was the Blue cell's primary window on the AOI. For future phases we are investigating providing integrated 3-D views of the AOI to the players, both to replicate sensor feeds and to provide a 3-D common operational picture.

An important practical factor for federation development and debugging is the time it takes to distribute new software out to the federation. MARCI has been rewritten to use an adaptive software distribution technique in which each federate that has already received the software update becomes a distributor for the remaining federates. The technique automatically adapts to new configurations by testing the round trip time for messages to reach other federates before selecting where a federate should get its update from. It has reduced software distribution times from on the order of an hour to minutes.

6. Conclusions

The dense environment characteristic of urban areas challenges the capabilities of our current combat simulations. It requires significantly more computational power and new and more efficient algorithms. The incorporation of civilians into the simulation underscores how barren our simulations are and the need to incorporate the cultural environment. While this federation was able to satisfy a few of these requirements much more work is needed to represent urban combat accurately.

Significant progress was made in federation scalability during this experiment. Distributed logging showed that it was possible to collect all the data without creating a central bottleneck for the federation. This enabled source squelching. The practical application of very large numbers of interest management regions was demonstrated. This enabled the use of queries to reduce the network requirements of the simulation. A large highly detailed urban area was represented embedded in a worldwide terrain and a large population of civilian entities was demonstrated responding to a simple cultural environment. To a large extent, this experiment was made possible by the use of an experimental RTI, RTI-s. Our ability to customize the RTI to support new requirements was essential to making this federation work.

As we address the challenges of urban simulation, we adopt solutions like intersection logic controllers, road-based dead reckoning and footprint queries that are not directly compatible with standards like the Real Time Platform Reference (RPR) FOM [20]. This will make incorporating other federates into the UR Federation more difficult in the future. However, it is a necessary price for enabling experimentation in the urban environment.

7. Acknowledgements

The authors thank J9 and the JAWP for supporting this project, and the many people who contributed to its success. Bill Helfinstine of Lockheed Martin developed the IMPs and the RTI-s enhancements. Dan Speicher of Lockheed Martin developed the clutter and the intersection control algorithms. Ke-Thia Yao, Dennis Allard, Brian Barrett, and Gene Wagenbreth of ISI developed the distributed loggers. Phil Colon of Toyon distributed SLAMEM. Dave Bakeman developed CTBD Format 8. Greg Rafuse of Alion developed FAARS and Rich Williams of BMH extended MARCI to support the UR Federation and SPPs. We thank the J9 Experiment Engineering Team for their dedication and the many innovations they produced to make this experiment work.

8. References

[1] Ceranowicz, A., Dehncke, R. and Cerri, A. (2003) Moving toward a Distributed Continuous Experimentation Environment, *Proceedings of the 2003 Interservice/Industry Training Simulation and Education Conference.*

[2] Lucas, R. and Davis, D. (2003) Joint Experimentation in Scalable Parallel Processors, *Proceedings of the 2003 Interservice/Industry Training Simulation and Education Conference.*

[3] Ceranowicz, A., Torpey, M., Helfinstine, B., Bakeman, D., McCarthy, J., Messerschmidt, L., McGarry, S. and Moore, S. (1999) J9901: Federation Development for Joint Experimentation. *Proceedings of the Fall 1999 Simulation Interoperability Workshop,* Paper 99F-SIW-120.

[4] Ceranowicz, A., Nielson, P. and Koss, F. (2000) Behavioral Representation in JSAF. *Proceedings of the Ninth Conference on Computer Generated Forces*, Paper 9TH-CGF-058, University of Central Florida.

[5] Rosenbloom, P., Laird, J., McDermott, J., Newell, A. and Orciuch, E. (1985) R1-Soar: An Experiment in Knowledge-Intensive Programming in a Problem Solving Architecture, *The Soar Papers, Volume One,* Rosenbloom, Laird, Newell Eds., MIT Press 1993.

[6] Graebener, R., Rafuse, G., Miller, R. and Yao, K. (2004) Successful Joint Experimentation Starts at the Data Collection Trail – Part II, *Proceedings of the 2004 Interservice/Industry Training Simulation and Education Conference.*

[7] Prager, S., Bakeman, D., Haes, S. and Goodman, G. (2004) Malls, Sprawl, and Clutter: Realistic Terrain for Simulation of JUO, *Proceedings of the 2004 Interservice/Industry Training Simulation and Education Conference.*

[8] Adelson, S., Salemann, L., Farsai, S., Miller, D., Miller, T. and Nakanishi, M. (2004) Complex Terrain Databases for Urban Operations. *Proceedings of the Spring 2004 Simulation Interoperability Workshop,* Paper 04S-SIW-007.

[9] Miller, D., Cauble, K., Bakeman, D., Torpey, M., Helfinstine, B. and Ceranowicz, A.. (2003) Extensions to the CTDB Format to Support Joint Experimentation., *Proceedings of the 2003 Spring Simulation Interoperability Workshop*, Paper 03S-SIW-133.

[10] Speicher, D. (2004) Simulating Urban Traffic in Support of the Joint Urban Operations Experiment. *Proceedings of the 2004 Interservice/Industry Training, Simulation, and Education Conference.*

[11] Simon, H. (1969) *Sciences of the Artificial*, Cambridge: MIT Press.

[12] Haskell, E., Volkert, J. and Dufault, B. (2004) Red Force Modeling in JFCOM Experiment Urban Resolve. *Proceedings of the 2004 Interservice/Industry Training Simulation and Education Conference.*

[13] Smith, J. (1994) Near-term Movement Control in ModSAF, *Proceedings of the Fourth Conference on Computer Generated Forces and Behavioral Representation*, Orlando, pp 249-260.

[14] Reece, D., Kraus, M., and Dumanoir, P. (2000) Tactical Movement Planner for Individual Combatants, *Proceedings of the Ninth Conference on Computer Generated Forces*, Paper 9TH-CGF-045, University of Central Florida.

[15] Sivia, D. (1996) *Data Analysis A Bayesian Tutorial*, Oxford: Oxford University Press.

[16] McGarry, S. and Torpey, M. (1999) Back to Basics: Balancing Computation and Bandwidth. *Proceedings of the Fall 1999 Simulation Interoperability Workshop,* Paper 99F-SIW-188.

[17] Williams, R., Tran, J. and Helfinstine, B. (2004) Creating a Communication Infrastructure for Simulating Urban Operations, *Proceedings of the 2004 Interservice/Industry Training Simulation and Education Conference.*

[18] Helfinstine, B., Torpey, M. and Wagenbreth, G. (2003) Experimental Interest Management Architecture for DCEE. *Proceedings of the 2003 Interservice/Industry Training Simulation and Education Conference.*

[19]Williams, R. and Tran J. (2003) Supporting Distributed Simulations on Scalable Parallel Processors, Proceedings of the 2003 Interservice/ Industry Training Simulation and Education Conference.

[20] SISO (1999) *Guidance, Rationale, and Interoperability Modalities for the Real-time Platform Reference Federation Object Model (RPR FOM),* Version 1.0, 10 September 1999, www.sisostds.org.

9. Author Biographies

Andy Ceranowicz is the technical lead for federation and Joint Semi-Automated Forces (JSAF) development at 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.

Mark Torpey is a J9 federation developer and is the lead developer and integrator of JSAF. He is a staff software engineer at Lockheed Martin Simulation Training and Support — Advanced Simulation Center (LMSTS-ASC) in Burlington, Mass. Mark holds an M.S. in computer science from the University of Massachusetts.

Endnotes

[1] Simulation of the Location and Attack of Mobile Enemy Missiles — Toyon Corporation

[2] Joint Semi-Automated Forces — JFCOM

[3] Electromagnetic Intelligence

[4] Virtual Network Computing

[5] Multiple Application Remote Control and Instrumentation

[6] Political, Military, Economic, Social, Infrastructure, and Information.