Embedded Mobile Augmented Reality Trainer Within a Distributed HLA Simulation

David Armoza
Dennis G. Brown
Naval Research Laboratory
4555 Overlook Avenue SW
Washington, DC 20375-5320
202-767-3961, 202-404-7334
armoza@ait.nrl.navy.mil, dbrown@ait.nrl.navy.mil

Keywords:
Augmented Reality, BARS, HMD, JSAF, HLA, RTI, Semi-Automated Forces, Embedded Training.

ABSTRACT: The Battlefield Augmented Reality System (BARS) is a mobile augmented reality system that displays battlefield intelligence information to a dismounted warrior. The primary BARS components include a wearable computer, a wireless network, and a tracked, see-through Head Mounted Display (HMD). The computer is responsible for generating graphics to the HMD that appear, from the user's perspective, to exist in the surrounding environment. Thus, a building could be augmented to show, for example, its name or a plan of its interior. Simulated entities with which the BARS user might interact can also be displayed.

Joint Semi-Automated Forces (JSAF) is a software system capable of simulating entity level platforms and behaviors in a heterogeneous, distributed computing environment. This system implements communication between distributed components of the battle space with the High Level Architecture (HLA). A very important aspect of each is that interactions spawned by "live," "virtual," and "constructive" forces are seamlessly integrated.

In our experiment, we bridge the gap between a user wearing the BARS hardware and constructive entities. We use the Real-Time Infrastructure (RTI) to distribute and translate entity pose and appearance data between the BARS data distribution system and JSAF. This interface allows a BARS user to interact with the SAF entities in real time, including armed engagement. We will discuss the design and implementation of these two interfaces, including special considerations unique to each, and their applications with respect to embedded training.

1. Introduction

Mobile augmented reality has the potential to revolutionize live, embedded training. While live training provides the most realistic experience, it tends to expensive by virtue of its non-repeatability and consumed resources. Purely virtual training is useful but does not yet fully replicate the experience of live training [1]. However, with mobile Augmented Reality (AR), it is possible to insert synthetic forces into live training exercises in a realistic manner. This approach permits trainees to tactically engage synthetic forces.

In previous work, we discussed how to insert synthetic forces into the real world using AR [2]. Once it is possible for the AR user to interact with synthetic forces, how are the behaviors of those forces controlled? Existing Semi-Automated Forces (SAF) systems are the end result of many man-hours designing software algorithms capable of simulating the behavior of military and civilian forces. By connecting the Battlefield Augmented Reality System (BARS) to these systems, we have leveraged their capabilities for use in mobile AR training—the BARS user appears to the SAF system as another of its distributed entities, and the SAF entities appear in the BARS display as synthetic forces. Thus, it can be used as an embedded training system for multiple areas of interest. Our preliminary experiments have targeted simple military exercises for dismounted infantry, but can be transitioned for homeland defense scenarios, such as might be needed by first responders.

We used the Joint Semi-Automated Forces (JSAF) software system [3] to support research for the US Navy. The interface with JSAF uses a custom High Level Architecture (HLA) federate. This paper will explain how the interfaces were built, and what they can do.

In section 2, the problem is described in more detail. The existing BARS data distribution system is described in section 3. Section 4 describes how BARS was interfaced to JSAF. Section 5 wraps up with some ideas for future work.

2. Problem Statement

2.1 Mobile AR For Embedded Training

AR applies virtual reality techniques to the user's real world experience by generating, from the user's perspective, graphics and sounds that appear to exist in the surrounding environment. For example, it is possible to augment the view of a building to show its name, a plan of its interior, icons to represent reported hazard locations, and the names of adjacent streets. Research on BARS [4] has focused on the problem of developing information systems able to enhance the user's situation awareness by providing data about the environment and its contents.

The centerpiece of BARS is a mobile augmented reality system that displays head-up battlefield intelligence information to a dismounted warrior, similar to the head-up display (HUD) systems designed for fighter pilot cockpits. The system consists of a wearable computer, a wireless network, and a tracked see-through Head Mounted Display (HMD). Three-dimensional (3D) data about the environment is collected (through surveying, sensors, or reports by other users) and made available to the system. By using a Global Positioning System (GPS) unit and an orientation tracker it is possible to know where the user is located and the direction in which he is looking. Based on this data, the desired 3D data is rendered to appear as if it were in the real world. Figure 2.1 shows the BARS wearable system.

In addition to providing real-time situation awareness data, BARS can render synthetic forces into its real-world view. Compared to providing situation awareness data, rendering synthetic forces in BARS has additional needs because even though the basic AR functionality is the same, the paradigms required to solve these problems are very different. In the situation awareness mode, BARS adds information to

Figure 2.1 The BARS Wearable System.

the real-world view that the user would not normally see. It is necessary for this information to stand out and appear artificial. In the embedded training mode, BARS inserts cues into the real-world view that, ideally, the user could not distinguish from reality. For example, a team of trainees in a Military Operations for Urban Terrain (MOUT) training facility could work together against a simulated enemy force.

Special needs dictated the changes necessary to accommodate embedded training scenarios in BARS. These changes were described in more detail previously [2], but we will summarize them here. First, the graphics renderer had to be updated to handle realistic entities, helped by the use of a third-party human animation system. Second, occlusion models had to be created to represent the locale in which the training scenario occurred. These models are used to properly occlude the synthetic forces as they walk behind walls, over hills, and so on. Without these models, the synthetic forces would be visible in unrealistic ways. Figure 2.2 shows an example of what a user sees when using BARS for embedded training. Finally, a tracked weapon surrogate had to be added to the system. This weapon has the same type of tracker

Figure 2.2 Computer-generated soldier drawn in the real world.

that tracks the user's head. When the user pulls the trigger, the system knows where the weapon is aimed and can model the effects of firing the weapon appropriately.

2.2 The SAF-based entity

In our experiment, we used a real-time, time-stepped simulation built upon the design principles of Modular Semi-Automated Forces (ModSAF) [5], the predecessor of JSAF. JSAF is designed to seamlessly integrate three types of entities. An *entity* is intended to represent, within the simulation, a specific instance of a real-world, physical asset. The entity types that a SAF simulation is designed to simultaneously coordinate within a single distributed world are referred to as "live", "virtual", or "constructive". A "live" asset represents updates from a real-world object (e.g., an actual destroyer participating in a naval exercise). A "virtual" entity is an entity that reflects the actions of a man-in-the-loop tool. A "constructive" entity is an entity whose behavior and state are determined purely from the internal code modules of the SAF system. The simulation system maintains a database of SAF-based entities. The entities are stored in the database fall into two basic categories. The first category represents the constructive entities that originate on the local SAF processor. The second category tracks the state updates of all the other entities in the simulation, whether they are constructive entities from other parts of the distributed simulation, man-in-the-loop simulators, or updates from actual forces. For the purposes of our discussion, the JSAF dismounted infantry (DI) entity is an example of a constructive entity, while the BARS user is an example of a virtual entity. We shall discuss this database in more depth below (see section 2.3).

The SAF-based entity represented in the simulation database is derived from an object-based architecture. Each entity is a unique instance of a parameterized set of object models that are joined to create a descriptive model for an entity type, whether it is a human infantryman, a tank, or any of the real-world objects that a simulation tool might need to represent. The entity is created by joining a set of generic component models that may or may not be specified at a higher level of detail through object-oriented inheritance. For instance, the SAF DI we utilized is created by using a derived instance of the generic Hull model (upon which all entities are based). The Hull model contains the functional interface for the movement of an entity within the simulation, and provides the frame upon which other models hang. One of the primary models that hang upon the Hull is the Turret. The Turret is typically configured with an arbitrary number of components (e.g., weapons, sensors, etc.). In the case of a SAF DI, there is a derived Turret that models the torso and head of a human through the additional components "mounted" upon or "carried" by this Turret:

Aural sensor model ←→ Ears
Visual sensor model ←→ Eyes
Ballistic Gun model ←→ Rifle (M16A)

Furthermore, there are models that represent the interaction of the SAF entity with the environment and instantaneous effects. For the SAF DI, this includes but is not limited to the following models:

- Line of Sight Models: visual occlusion by buildings and terrain, temporary environmental effects ranging from time-of-day based light levels to obscurement from smoke.
- Damage Models: damage effects upon entities or terrain from indirect and direct fire weapons.
- Weapon Models: determine the chances for indirect and direct weapon fire to hit a particular target, whether that be an environmental feature such as a road or as in our case an enemy combatant ("red" forces).
- Terrain Models: provide the environment wherein a DI moves (includes algorithms to deal with obstacle avoidance, navigating multi-elevation structures (MES), etc.).

By using these models along with many others, it is now possible to track entities and allow them to interact. The interactions can be as simple as those found in physics-based models (e.g., ballistic trajectories, etc.) to task-based state behaviors (e.g., detection and response to enemy units, etc.).

2.3 Distributed Entities

State information for distributed entities is shared within the SAF system by using a distributed database, called the PO (Persistent Object) database [6]. It is an object-oriented database that provides a localized interface for the command and behavioral information that must be shared between the distributed simulation elements. This database was designed to address several issues. It allows an arbitrary number of processing engines to work together in a seamless fashion. It provides a reliable method for the thousands of objects that must be tracked in a simulation to communicate both the "ground truth" of a simulation along with the command and control (C^2)

behavior necessary to organize complex multi-entity tasks (e.g., unit organization, missions, etc.).

By leveraging the capabilities of the SAF system to distribute entity information and behavior it is possible to design scenarios for the constructive entities that can execute arbitrarily complex behaviors. The complexity of the tasks supported by the SAF systems at the time of our prototype development was fairly basic for a DI. Ongoing efforts in the area of Joint Urban Operations (JUO) are expected to enhance the capabilities of the SAF systems to provide better options for embedded training.

In our case, we were primarily interested in communicating the existence of a BARS user to the SAF, thereby permitting SAF entities to properly interact with the BARS "entity." To this end, we used the existing communication protocols that signal the creation, update and destruction of a remote entity to the PO database. Thus, the SAF system in question would have sufficient information to be aware of this new entity. Conversely, the SAF systems would transmit entity creation, update and destruction data to external applications as dictated by their data distribution models.

Now that we have described the two major pieces (the mobile AR system and the SAF entities), we will describe how they were connected, beginning with a description of the BARS data distribution system.

3. BARS Event Distribution System

The BARS distribution system is based entirely on the concept of events, which are discrete packets of information sent between program modules within a BARS application and across applications. Events are used to instantiate objects (in effect, to transmit a view of a database between systems), update existing objects, and to provide other non-database status information such as protocol management data. The BARS event distribution system is described in detail in [7], so only the relevant features will be described in this section.

3.1. Replicated Object Repositories

All of the data for a scenario is stored in a distributed database called the object repository. The data consists of the mostly static models of the physical surroundings (buildings, streets, points of interest, etc), dynamic avatars that represent users and other entities, and objects created to communicate ideas, such as reports of enemy locations, routes for users to follow, and digital ink. This repository is replicated in whole or in part for each application by sending and receiving events. When an object changes, only the changed information is distributed, rather than the entire object state. A database object is assumed to be "live" until an explicit "destroy" event is distributed.

When a BARS application starts, it loads an initial set of objects from a number of sources, including saved databases, other applications already running on the network, and files specified on the command line. The initial set of objects typically consists of street labels, landmarks, building information, and other terrain-like information, as well as an initial set of objectives, routes, and phase markers for the current task. Since the user is initially given a database to start, and everything else in the wearable system is self-contained, the user will have a working AR system even if all network connectivity is lost during an operation.

3.2. Event Transportation

The heart of the event transportation system is the Object and Event Manager. The Object and Event Manager is responsible for dispatching events within an application and distributing those events to remote applications. When the Object and Event Manager receives an event, it places that event on an asynchronous event queue. An event dispatching thread delivers the event to all the listeners that are subscribed to receive the specified event type. It is possible to dynamically extend the set of events and listeners handled by the dispatcher at runtime. Figure 3.1 shows the flow of events within an application.

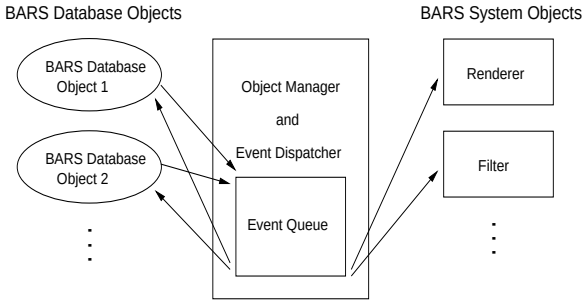

Figure 3.1 The flow of events within an application.

This event mechanism was extended to allow many separate applications to trade events by creating Event Transporters. Event Transporters allow Object and Event Managers in different application instances to send and receive events over Internet Protocol (IP).

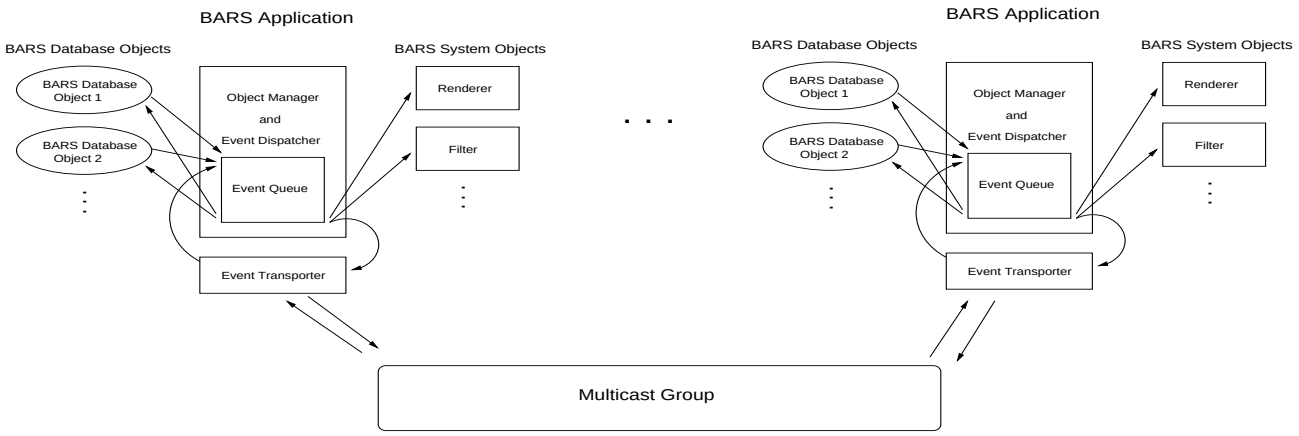

Figure 3.2 The flow of events between applications.

Figure 3.2 shows the flow of events between applications. If an event is tagged as distributed, an Event Transporter serializes the event and broadcasts it to other applications. The Event Transporters in remote applications synthesize the event object and dispatch it on those applications' event queues. The system uses transporters based on IP multicast and TCP/IP.

As events are created, they are tagged "reliable" or "unreliable" designating how they should be transported. Object creation and deletion events are always sent reliably. Object changes are sent reliably or unreliably based first on whether or not the modification is relative to other changes. Relative changes have an ordering and each one is important, so those are sent reliably. Non-relative changes, such as the constant updates of a user's position, are mostly sent unreliably since if one were missed, the next would overwrite it anyway. Periodically, these non-relative changes are sent reliably. This policy makes the assumption that the implementation of IP networking in a real operation may drop IP packets often, making reliable multicast expensive, and so events are not sent reliably unless they are thought to be truly necessary.

3.3. Bridges

Some applications communicate with external information systems. These applications are called *bridges*. They join both the BARS distribution system and an outside system and translate object creation and change events between BARS and the outside system. By maintaining associations between BARS objects and these outside objects, it is possible to represent those objects in BARS and vice-versa. Figure 3.3 shows how a bridge application fits into a session.

It is through bridge applications that we connect BARS to JSAF. We will describe this application next.

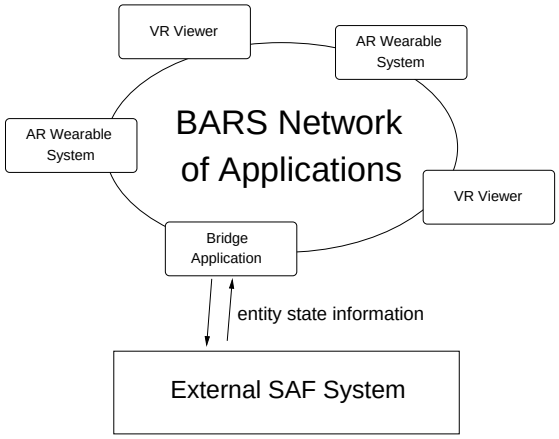

Figure 3.3 A bridge application and other applications.

4. Connecting JSAF and BARS

4.1 Overview of HLA

The High Level Architecture (HLA) is an IEEE standard [8] that was developed as a means by which the Modeling and Simulation (M&S) community could share a common architecture for distributed modeling and simulation. There are three underlying portions of this framework; the rules, the federation interface specification, and the object model template. The HLA federation interface defines the common framework for the interconnection of interacting simulations, and is of particular interest to our understanding JSAF. The HLA Runtime Infrastructure (RTI) and a set of services implement this interface. The RTI and these services allow interacting simulations to efficiently

exchange information in a coordinated fashion, when they participate in a distributed federation.

As mentioned above, a *federation* is a set of associated applications that use a unifying object model, agreed upon prior to execution, to achieve a specific objective. These applications are called *federates*. JSAF is a federate, as is the SAF Bridge mentioned in section 3. A federation is managed by a "FedExec" process that is instantiated by the first federate that successfully invokes the service for federation creation. Conversely, a federation ends when the last federate registered with the FedExec invokes the service for federation destruction. It is typical for all federates to possess the capability to "create" and "destroy" a federation, thus avoiding an explicit invocation order for federates in a federation. Federates use the return values of these RTI services (and many others like them) to recognize when various calls may be appropriately ignored. The unifying object model, called the Federation Object Model (FOM), defines the attributes of objects and/or interactions that federates are permitted to communicate to other federates.

Within this framework, an object refers to an instance of a simulation entity whose state changes are shared with other federates. Similarly, an interaction is typically used to communicate an event to other federates. This contrasts somewhat with the BARS data distribution system's concept of everything being an event. The "publication" and "subscription" of the object updates and interactions are an additional aspect of the ability of the RTI to communicate the requested information. Even if the FOM permits the transmission of data between federates, it is the responsibility of the individual federates to publish (announce that they send out data) and subscribe (announce interest in receiving data) for the objects and interactions in which they are interested. We leave further details of the HLA model of distributed computing to the interested reader.

4.2 Description of JSAF

As mentioned earlier, JSAF evolved from the work on ModSAF for the Defense Advanced Research Program Agency (DARPA). It is a collection of libraries and programs that are oriented toward real-time large-scale distributed simulation. JSAF is actively used as a tool that validates the applicability of integrating transition technologies into the modern warfighter's inventory of capabilities and tools. Its current development is sponsored by US Joint Forces Command, Joint Experimentation Directorate (J9), and has been integral in the US Navy's Fleet Battle Experiments.

JSAF primarily uses the HLA as its communication layer. For any object or interaction, JSAF determines the locality of related entities for this communication and sends the necessary information via the RTI if they are known to be upon a different processor. For example, if a DI fired upon another DI on the same JSAF process, the simulation would handle this interaction internally. But if the second DI was known to be upon another federate (in our case a BARS bridge), then the interaction would go out via the RTI. The only limitation to remember is that JSAF must publish this interaction (a default setting). To complete this communication, the federate expecting to receive this interaction must correspondingly subscribe to this interaction.

We built a SAF Bridge application, as described in section 3.3, to connect BARS and JSAF. This application implements the BARS networking paradigm as well as implementing an RTI interface to communicate with JSAF. JSAF has a set of libraries supporting the Agile FOM Interface (AFI). By using this interface, JSAF creates a mapping of its internal data representations into an external representation. This mapping is necessary for JSAF to participate in different federations without modification. The mappings are stored in special files, called reader files that are interpreted at run-time. The unexpected advantage to this approach is that these libraries can be used by other applications, such as our SAF Bridge, to create a pseudo-JSAF federate. By including a subset of the JSAF support libraries, it is possible to create SAF representations of BARS objects in the bridge. This has many advantages:

- The transparent communication between the bridge and JSAF by RTI object updates and interactions (i.e., calls and formatting issues handled by the internal JSAF libraries).
- Using the same terrain databases and JSAF's terrain libraries to ease position translations between BARS and JSAF.
- Leverage physical models in JSAF to handle weapon behavior (ballistics, damage, etc.).

4.3 Issues

In BARS and JSAF there are corresponding "events" that relate to the creation, change and destruction of an entity. In BARS, these specific events trigger the dissemination of the salient object state changes to other applications, such as the SAF Bridge. When it receives this information, it is necessary to translate the data into the appropriate object updates such that it may be sent via the RTI to JSAF. BARS typically

updates positions at a much higher rate than a system such as JSAF desires. So we track the updates for our BARS user in a lookup table, and use the simulation time libraries we inherit from the JSAF interface code to limit the update rate to a more reasonable one (1 Hz). An exception to this rule is when there are significant orientation and movement changes in the BARS user (i.e., begin walking, change facing, etc.).

JSAF has a corresponding mechanism for object updates. The SAF Bridge catches incoming updates at the rate they arrive and store the information in another lookup table (STL map). In this case, we also store a pointer to the JSAF platform object that relates to the object update. This allows the SAF Bridge application to use the built-in dead reckoning code from JSAF to interpolate the current position of a moving entity without having to receive constant updates. The designers of ModSAF recognized that if constant position updates were going out from all entities, then the network bandwidth would quickly be consumed. They implemented a dead-reckoning algorithm that cut down on network traffic. In essence, each entity would have a set frequency to update its position. Remote machines would calculate a new position based on dead reckoning. The originating machine on the other hand would simultaneously calculate its ground truth position and its new dead-reckoning position, and if they differed by some delta, a new update would be broadcast to the network. This aspect of dead reckoning has a side effect due to the normal effects of network latency. For example, the SAF Bridge sends position updates to BARS at 10 Hz and a new JSAF update arrives indicating that at some point in the past the entity being tracked had turned. This leads to a visual artifact in the BARS environment display when the SAF entity "jumps" to its new location.

In a similar fashion, the interaction between an "armed" BARS user and JSAF DIs requires additional management. The SAF system needed to be informed whenever the BARS user was firing. The tracked weapon uses a simple momentary contact button that the user presses to indicate a firing. The tracker data is used to call the JSAF provided ballistics library and determine if any of the synthetic forces have been hit. If the target is hit, a "fire" interaction is sent by the RTI to JSAF. Once received, JSAF can compute the damage and if necessary change the status of the SAF entity (damaged, dead, etc.). By using the corresponding libraries inherited from JSAF, the BARS user can also be targeted and damaged by SAF entities. We have yet to implement a mechanism to indicate incoming weapon fire and damage to the BARS user.

The translation of entity parameters was less straightforward. BARS uses a simple coordinate system based in meters in three dimensions from an arbitrary origin. JSAF uses a tiled, multi-cell terrain database that depends upon a Geocentric Coordinate System (GCC) to unify position-related data. These databases are stored in Compact Terrain Database (CTDB) format. The position data is easily converted into the more commonly recognized Global Coordinate System (GCS) format, which measures position as a triplet of latitude, longitude, and altitude. We converted between the two systems using a third-party library to translate GCS into Universal Transverse Mercator (UTM), and vice versa. This was useful since UTM uses meters in three dimensions from a grid point on the globe. A simple offset correction yields the BARS coordinate. The reverse of this process converts BARS coordinates into GCS.

5. Conclusions and Future work

We designed a system that can help trainees in situations requiring engagement between individual combatants, such as those in MOUT scenarios. By using mobile AR, synthetic forces are inserted and engaged realistically in the real world. A connection to JSAF allows the synthetic forces to behave intelligently and gives trainers a familiar interface with which to control the scenario. This system gives the trainee the benefits of both live training and of having synthetic actors for a predictable, repeatable scenario. We are currently testing this system and comparing its performance to a previous prototype built at NRL [2].

Although the basic pieces are in place to use mobile AR for embedded training, there is still much work to be done. We have in mind several improvements as future work. These improvements would yield a more effective system:

- Implement a method to convert BARS terrain models into the CTDB format used by the SAF systems, and thereby enabling the BARS occlusion model to exactly match the model used by the SAF entities.
- Make the synthetic forces look more realistic in the AR display. The forces are currently drawn without respect to environmental conditions, shadows, or any occluding items that are not already in the occlusion model.
- Increase the accuracy of the weapon tracking system. The current tracking methods are accurate enough for measuring the user's viewpoint, but

even slight errors in tracking the weapon will greatly reduce the accuracy of the user's aim.

6. References

[1] Stytz, M.R.: "Distributed Virtual Environments" IEEE Computer Graphics And Applications, pp. 19-31, May 1996.

[2] Brown, D., Baillot, Y., Julier, S.J., Armoza, D., Eliason, J.J., Livingston, M.A., Rosenblum, L.J., & Garrity, P.: "Data Distribution for Mobile Augmented Reality in Simulation and Training" Proceedings of the 2003 Interservice/Industry Training, Simulation, and Education Conference, Orlando, December 2003.

[3] Space and Naval Warfare Systems Center – San Diego: "Synthetic Theater of War – Frequently Asked Questions" Retrieved February 5, 2004 http://www-code44.spawar.navy.mil/STOWFAQ/

[4] Julier, S., Y. Baillot, D. Brown, & L. Rosenblum: "BARS: Battlefield Augmented Reality System" NATO Symposium on Information Processing Techniques for Military Systems, Istanbul, Turkey, October 2000.

[5] Ceranowicz, A: "Modular Semi-Automated Forces", Proceedings of the 1994 Winter Simulation Conference, pp. 755-761, 1994.

[6] Calder, R.B., Smith, J.E., Courtemanche, A.J., Mar, J.M.F., Ceranowicz, A.Z.: "ModSAF Behavior Simulation and Control" Proceedings of the Third Conference on Computer Generated Forces and Behavioral Representation. Orlando, Florida: Institute for Simulation and Training, University of Central Florida, March, 1993.

[7] Brown, D.G., Julier, S.J., Baillot, Y., Livingston, M.A., Rosenblum, L.J.: "Event-Based Data Distribution for Mobile Augmented Reality and Virtual Environments" Presence: Teleoperators and Virtual Environments, Volume 13, Issue 2, April 2004.

[8] Institute of Electrical and Electronics Engineers, International Standard, ANSI/IEEE Standard 1561.1-2000, Standard for Modeling and Simulation High Level Architecture, Federate Interface Specification, September 2000.

Author Biographies

DAVID ARMOZA is a Computer Scientist at the Naval Research Laboratory. He received his B.A. in Computer Science from University of Maryland in 1989 and his M.S. in Computer Science from The Johns Hopkins University in 1996. He works in the area of Distributed Simulation. His current research interests include experimentation with the US Navy's Joint Semi-Automated Forces (JSAF) simulation system, and distributing stand-alone applications with DMSO's High Level Architecture (HLA).

DENNIS G. BROWN is a Computer Scientist at the Naval Research Laboratory. He received his B.A. in Computer Science from Rice University in 1996 and his M.S. in Computer Science from the University of North Carolina at Chapel Hill in 1998. He works on the Battlefield Augmented Reality System (BARS) and multi-modal virtual reality projects. His research interests include ubiquitous computing and data distribution. He is a member of IEEE.