An Intelligent User Interface for Enhancing Computer Generated Forces

Brian Stensrud
Glenn Taylor
Soar Technology, Inc.
3600 Green Court, Suite 600
Ann Arbor, MI 48105
734-327-8000
{stensrud, glenn}@soartech.com

Bradley Schricker
Dynetics, Inc.
1002 Explorer Boulevard
Huntsville, Alabama 35806
256-964-4979
brad.schricker@dynetics.com

John Montefusco
Science Applications International Corporation (SAIC)
6723 Odyssey Drive
Huntsville, AL 35806
256-655-2655
john.montefusco@us.army.mil

Jeffrey Maddox
U.S. Army Aviation and Missile Command
Research, Development & Engineering Center
Redstone Arsenal, AL 35898
256-876-7716
jeffrey.a.maddox@us.army.mil

Keywords:
Computer generated forces, intelligent user interfaces, modeling and simulation

ABSTRACT: *The DoD widely uses airborne computer generated forces (CGFs) in simulation environments, however most CGFs are not autonomous or interactive enough to be directed naturally by a human controller. Full simulation capability requires detailed, doctrinally correct CGFs that can be controlled directly by airspace managers, such as air traffic controllers, air traffic services, or even ground commanders operating in a combat zone. Compounding the problem of controlling CGFs is that the military already employs a wide range of CGF behavior systems with different levels of autonomy, fidelity, and interactivity. This paper describes an ongoing effort to develop a "Controller's Assistant" to enhance the apparent capabilities of existing CGFs by introducing a speech-enabled, intelligent user interface (IUI) between a human controller and a set of CGFs. This Controller's Assistant allows for a human controller to use doctrinal airspace control commands to interact with a diverse set of CGFs in a distributed HLA simulation federation. The IUI translates spoken user commands into HLA commands that direct the CGFs, and also monitors CGF progress to provide spoken feedback to the controller when doctrinally appropriate.*

1. Introduction and Motivation

The DoD Modeling and Simulation community has used Computer Generated Forces (CGFs) as a way to reduce manpower requirements for running large-scale exercises. Most CGF systems are often described as *semi-automated forces*, meaning that they require some oversight by human operators to manage their progress (to ensure that they are performing their tasks correctly), to manually task them in sequence, or to dynamically re-task them as required by the mission. Each CGF system has its own native interface that the operator must learn and through which the operator must simultaneously with other interfaces to monitor and task the CGFs. These CGF systems can also vary widely in their capabilities – what tasks they can execute autonomously, and what tasking they can take during a mission – so the operator must know what those capabilities are for each CGF he or she is responsible for controlling. As the number and diversity of CGFs used in a distributed simulation grows, the manpower requirements to manage them grows, as well. In this paper, we describe an approach to reducing the manpower requirements for managing distributed simulations by consolidating CGF-specific controls into a single, uniform interface that is natural to the operations of the domain.

In particular, we are interested in airspace management, where the human simulation operator plays the role of an airspace manager – often an air traffic controller (ATC). The predominant mode of communication between human ATCs and human pilots is voice interaction. Even where most CGFs are capable of dynamic re-tasking, they tend not to be able engage in dialogue with a human controller using doctrinal speech. In this paper, we describe our ongoing development of a speech-based, intelligent user interface (IUI) we refer to as a "Controller's Assistant." This system allows for a human controller to use doctrinal airspace control commands to interact with a diverse range of CGFs in a distributed simulation environment. The Controller's Assistant serves as a mediator between the human controller and a set of CGFs with varying capabilities.

2. Related work

There have been a few efforts at providing speech interfaces to simulation environments. SRI's CommandTalk system [1] provides a speech interface to the ModSAF CGF system for creating CGFS and control measures, tasking entities and changing missions on the fly. QuickSet [2] is a multi-modal (speech and pen) interface to ModSAF for setting up and running CGF simulations. Other CGF systems provide doctrinal speech interfaces for command and control or teammate relationships; for example TacAir-Soar [3] is a CGF behavior model that can interact with a human ATC or another CGF through a speech interface. In each of these approaches, a single CGF system was extended to incorporate speech as a new interface for interacting with the CGF. The novelty in our approach is in the application of similar interface technologies across a range of different CGF systems with the goal of minimally changing those CGFs to effectively improve their capabilities.

Soar Technology has recently completed two efforts directly related to our work on the Controller's Assistant. The first is work on an intelligent forces (IFORs) architecture that can control insurgent opposition forces (OPFOR) and civilian crowds. This architecture features a Plan Execution System that matches resources to plan objectives and develops assignments for CGF entities on a team and individual level. In this system, the CGF entities have varying capabilities and so must be tasked differently to accomplish the user's goals. The IFOR system features a graphical plan-editing user interface, which allows the user/operator to construct plans that span multiple, diverse CGF types.

The second, AutoATC [4], is another Soar Technology product that features an intelligent air traffic controller agent capable of assessing a simulated battle space for potential conflicts and generating warnings about those conflicts. This system uses an HLA/DIS interface to tap into a simulation environment and applies knowledge about airspace control to monitor for conflicts. While the Controller's Assistant is not automating the job of the controller, it must know enough about airspace control to understand the controller's commands and to properly coordinate the behavior of the underlying CGF entities. (In fact, we are re-using some of the elements of the AutoATC system for the Controller's Assistant.)

3. Design Constraints and Challenges

We have designed the Controller's Assistant to meet a number of requirements and constraints highlighted in this section.

Minimize changes in existing CGFs. An important requirement for the Controller's Assistant is that it must interact with existing CGF platforms with only a minimal amount of development required on the CGF side. Our approach is to use existing CGF communication protocols, as developing an entirely new communications protocol would require significant development to ensure that each CGF platform is compatible with that protocol. Since most existing CGF platforms support communication across a distributed simulation using HLA or DIS protocols, we can reduce the amount of changes required in those CGFs by communicating through them .

Speech Interface. As discussed above, human ATCs typically interact with air assets through speech, and thus a requirement of our Controller's Assistant was that it must provide a similarly "natural" interface for interacting with CGF systems. Specifically, all interactions between the human ATC and each CGF must be through speech. Note that this is not a general requirement for our approach. A graphical interface could be suitable for controlling CGFs as well, but speech is more fitting for the airspace management domain.

Interaction with Multiple Diverse CGFs. Our most significant design challenge for the Controller's Assistant was the requirement that our system support interactions with multiple, diverse CGFs. As discussed in the introduction, there are a wide variety of off-the-shelf CGF systems in existence, even among Army air assets. Each of these CGF systems was developed at different locations, by different people, and (most likely) for different specific purposes. Consequently, the specific capabilities of one CGF system may vary greatly from another even within a very small slice of the domain. Consider a simple *follow-route* command, where the human ATC wishes to instruct a CGF to fly some known, named route. While most airborne CGFs are capable of flying routes, they may be tasked in a variety of ways. If the CGF knows what a route is and what waypoints the route consists of, it may only be necessary to simply send the CGF the name of the route. If not, the system might instead require the sequence of waypoints that constitute the desired route. It might even be the case that the CGF system can only handle commands to adjust its altitude/speed/heading (or possibly as detailed as yaw/pitch/roll). Here, the command to fly a route breaks down into a long sequence of these adjustments simply to fly a route.

To interact with a variety of CGF platforms, therefore, the Controller's Assistant must be aware of the tasking capabilities for each CGF with which it interacts on behalf of the human controller. The system must then know how to translate and decompose incoming ATC messages into a sequence of equivalent CGF commands.

Depending on the complexity of the command from the human, the sequence of resultant CGF commands and the timing by which they are issued must also be handled by the Controller's Assistant. Consider the following command:

> *"Eagle1 this is Tower1, proceed along route BULLDOG to waypoint ROMEO, then divert to route ZULU and report when you reach waypoint SIERRA, over."*

It is quite possible that the CGF representing Eagle1 has no mechanism for detecting the achievement of a waypoint and dynamically changing course once that waypoint has been achieved. In these cases, the burden falls on the Controller's Assistant, who must track the position of Eagle1 as it proceeds along BULLDOG and detect when it reaches ROMEO. At that point, the system must then send a follow-up command to the Eagle1 CGF to change its course to route ZULU.

Finally, the Controller's Assistant must have the ability to communicate to the human ATC when the CGF lacks that capability. For instance, consider again the route-following command above. This command requires two responses from the CGF: an immediate acknowledgement that the command was heard and understood and a report when the CGF has reached waypoint ZULU. Absent a CGF capability to issue these utterances, it is the responsibility of the Controller's Assistant to (1) know the required content of the report to the controller, (2) identify when the report is required, and (3) know what phraseology to use when generating the speech.

4. Approach: Controller's Assistant as an Intelligent User Interface

To address the challenges described in the previous section, we have framed the Controller's Assistant as an intelligent user interface (IUI), effectively a mediator between the human controller and the CGFs in the battlespace.

4.1 Controller's Assistant Concept

To address the challenges described in Section 3, our Controller's Assistant IUI was designed to perform three major functions. The first is to manage dialogue and interactions with the human ATC. The component of the IUI in charge of this function is called the **Facilitator**. The facilitator is responsible for interpreting all incoming commands from the ATC and handling all outgoing speech from the CGF platforms to the ATC (in the cases where the CGFs are incapable).

The **Tasker** is responsible for decomposing incoming ATC commands into appropriate CGF-level commands and for delegating other internal tasks to the other two components. This requires knowledge of the capabilities of each CGF platform, which is specified *a priori* to the Controller's Assistant. If a CGF is instructed to follow a route but is only capable of following waypoints, for example, it is the Tasker's responsibility to decompose the incoming route name into the correct sequence of waypoint commands and properly time the issuance of that sequence.

Finally, the **Monitor** watches the state of the environment– particularly the controlled entities – to provide required situational awareness to the other two components. For example, in cases where CGFs are unable to issue their own status reports (e.g., the achievement of waypoints or routes), it is the Monitor's responsibility to track the aircrafts' states and report specified milestones to the Tasker or Facilitator.

Each component requires the proper operation of the other two for the system to operate correctly. The Monitor, as described above, is responsible for tracking the status of CGF aircraft. The Tasker uses the results of the Monitor to determine when to issue follow-up commands. The Facilitator also uses the output of the Monitor to determine when to voice messages to the human ATC. Similarly, the Facilitator parses and hands off all incoming ATC commands to the Tasker, who in turn sends requests for information back to the Facilitator when required.

The specific operation of the Controller's Assistant and each component depends heavily on the capabilities of the CGF platform being controlled. The less a CGF platform is able to do on its own (in terms of executing ATC commands), the more responsibility is placed on the Controller's Assistant to bridge that gap. However, the system should identify at run-time when CGFs are capable of certain behaviors that the Controller's Assistant would otherwise handle and avoid overwriting (or worse, dumbing down) those behaviors.

The desired effect of this interface is to augment the behaviors of each CGF being controlled so that they all are equally capable of executing ATC commands, making them appear identical to the eyes of the human controller.

5. Overall System Design

Figure 1 illustrates the system architecture for our Controller's Assistant. The core piece of the IUI, which houses the Monitor, Facilitator and Tasker components, is implemented within a single agent using the Soar cognitive architecture [5]. The human controller sends speech commands to the system and receives voiced responses via the SoarSpeak Speech Interface. SoarSpeak [6] provides speech-to-text (STT) and text-to-speech (TTS) services to Soar agents using COTS engines, such as Nuance and AT&T's NaturalVoices™. The Controller's Assistant is connected to each CGF (to which it is providing services) over HLA. All commands sent to CGF platforms from the IUI and all reports sent back to the IUI from the platforms are encapsulated as HLA interactions. Knowledge about the capabilities of each CGF is encoded *a priori* and input to the system at start-time, where it is loaded into a registry. The IUI matches incoming commands against this registry to determine what functions/behaviors are required that the target CGF does not support.

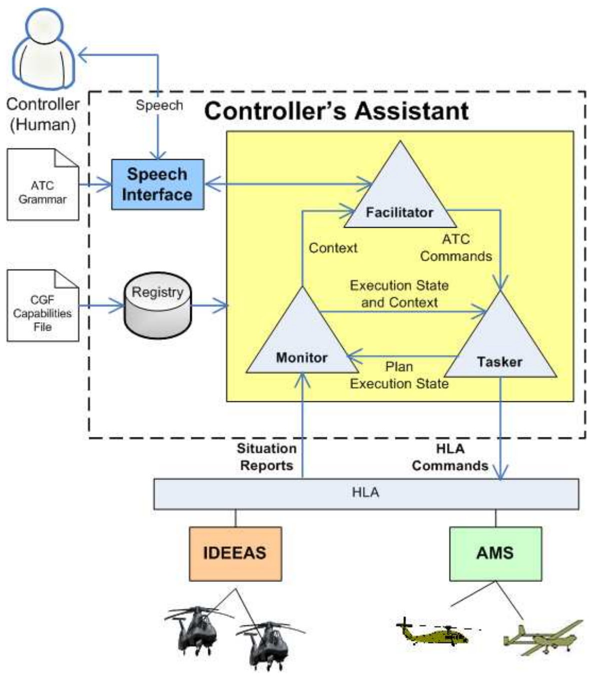

Figure 1. System Architecture.

5.1 Integration Framework

To date, we have been developing an HLA-based integration framework for the Controller's Assistant. HLA was chosen for initial development over other distributed simulation methods based in part on how prior exercises have been run at AMRDEC and in part because some HLA FOMs provide interactions relevant to airspace management. Specifically, we are using MATREX FOM v3.0.

However, the general architecture is ambivalent about the particular distributed simulation protocols, and the use of HLA is encapsulated in a plug-in architecture. The system can support alternative protocols with the development of additional plug-ins, one for each protocol. Which protocol to use with a CGF is encoded in the *a priori* knowledge about that CGF, and the translation to HLA, or any other protocol, is simply a step in a pipeline of converting commands from a representation in the Controller's Assistant to a representation suitable for that protocol.

5.3 Computer Generated Forces

For an initial implementation of the Controller's Assistant IUI, we are connecting to two different CGF platforms: Dynetics' Aviation Mobility Server (AMS) and SAIC's Interactive Distributed Engineering Evaluation and Analysis System (IDEEAS). Both of these systems support the MATREX version of HLA and have implementations that subscribe to and publish HLA interactions relevant to airspace management.

The **Aviation Mobility Server (AMS)** provides a central interface to numerous high-fidelity Army aircraft models for use in distributed simulation experiments. These models include:

- *Micro Air Vehicle (MAV)* – a 29-inch ducted fan UAV (Unmanned Air Vehicle) model
- *Tactical UAV (TUAV)* – essentially a Shadow 200 – a six-degrees-of-freedom (6-DOF) model that includes aerodynamics, mass properties, engine performance, fuel flow rates, sensor, and embedded flight computer models
- *Generic Rotary Wing (GRW)* – a generic representation of a rotary-wing UAV
- *Raven* – a high-fidelity RQ-11 model including wind-tunnel-based aerodynamics, control laws, and inertia and mass models based on lab measurements

IDEEAS is a physics-based warfighting simulation designed to solve specific scientific and engineering problems through constructive and virtual simulation. IDEEAS uses engineering-level models and predictions to conduct weapon system analysis based on performance calculations. IDEEAS is designed for evaluation of changes in equipment, tactics, weather, terrain, and C4I in the analysis of current and future weapon systems. Studies focus on system and subsystem issues and measures within the context of relevant battlefield vignettes and environmental conditions. Relevant to this effort, the IDEEAS simulation includes generic fixed- and rotary-wing CGF platforms that can be controlled via HLA commands.

In our current implementation, we are using the following MATREX FOM interactions with different support from the two different CGF systems, as seen in Table 1.

Table 1. Used MATREX FOM interactions

Interaction	**AMS**	**IDEEAS**
Move-Aircraft-Along-Route	X	X
Route-Report (achieving waypoint)	-	X
Entity-Create (at startup)	X	-

Our goal for the current implementation of the Controller's Assistant is to support the following mission behaviors for both CGF platforms:

- route following
- waypoint following
- hold at waypoint
- establish holding pattern using route
- adjustment of heading
- adjustment of altitude
- verbal acknowledgment of commands
- report current position
- report at waypoint
- divert to alternate route at waypoint

To perform these behaviors, the Controller's Assistant must exploit the capabilities of the individual CGFs as appropriate and fill in gaps elsewhere.

5.3 Controller's Assistant Operations

The Controller's Assistant IUI is implemented as a single Soar agent, where the Facilitator, Monitor, and Tasker components are modeled as separate persistent roles that share a single memory. Both intra-component tasking and communication is handled by reading and writing to this shared memory.

Typically, interactions between components occur on handoffs, where one component needs to inform another about a new task for which it is responsible or about the status of a completed or in-progress task.

The Facilitator posts ATC commands to the Tasker, which is responsible for developing a plan for a specific CGF to accomplish the assigned ATC command. This is essentially a process of plan refinement, generating plan execution steps that match the capabilities of the CGF – or, where the CGF has no relevant capabilities, generating tasks for the components internal to the Controller's Assistant. If a CGF is assigned an execution step, the Tasker packages that step into a CGF-appropriate network protocol (here, in HLA MATREX FOM interactions) and sends it over the wire. If a component within the Controller's Assistant is assigned an execution step, the responsible component will execute that action and inform the system of its achievement by marking the execution step as complete.

To illustrate how this is done in practice, we revisit the sample ATC command introduced in Section 3:

> *"Eagle1 this is Tower1, proceed along route BULLDOG to waypoint ROMEO, then divert to route ZULU and report when you reach waypoint SIERRA, over."*

The parsed message from the Speech Interface is first read by the Facilitator, who identifies the nature of the command and enters it (along with the appropriate parameters) into shared memory as a sequence of high-level commands. At this point, the Facilitator does an initial "sanity check" on the message to make sure, for example, that BULLDOG is a route in the current set of known routes, or that waypoint SIERRA is on route ZULU. If any of the tasking does not make sense, the Facilitator can immediately ask for clarification.

Once an initial check has been performed, the Tasker picks up the commands and generates an initial high-level set of plan steps, which are independent of any CGF capabilities. For example, here the doctrinal steps are:

1) acknowledge the command (when understood)
2) move along route BULLDOG
3) at waypoint ROMEO, divert to route ZULU
4) when waypoint SIERRA is reached, report

Knowing that Eagle1 has been tasked, and knowing the capabilities of Eagle1 with respect to each of these steps, the Tasker can begin to refine this high-level plan into a plan specifically for Eagle1 and its capabilities. In this case, the platform driving Eagle1 only supports commands to fly to individual waypoints. However, execution of the ATC command as stated requires verbal acknowledgements, route flying, waypoint tracking, and reporting waypoints.

The Tasker first creates an execution step to acknowledge the ATC command and assigns that step to the Facilitator, who responds:

> *"Tower1 this is Eagle1, roger, will proceed along route BULLDOG to waypoint ROMEO, divert to route ZULU and report at waypoint SIERRA, over."*

The Tasker then constructs a sequence of waypoints that comprise the desired composite route, which includes points from both routes BULLDOG and ZULU. Since Eagle1 is capable of waypoint following, the Tasker can achieve equivalent behavior by sending each waypoint as a separate command.

However, the timing of these commands depends on when the CGF reaches each intermediate waypoint – the Tasker cannot simply send the entire batch of waypoint commands at once. Furthermore, Eagle1 does not support direct reports to the ATC indicating achievement of waypoints. Therefore, the Tasker must add an execution step to monitor the achievement of the waypoint to go along with each waypoint command. These execution steps are assigned to the Monitor, who determines when Eagle1 has reached its next waypoint. Once this is the case, the Tasker can then send the subsequent waypoint command to Eagle1.

Finally, the Tasker creates a new execution step to verbally report the achievement of waypoint SIERRA and assigns this to the Facilitator. Once the waypoint and the Monitor's execution step are achieved, this reporting step is handled by the Facilitator:

> *"Tower1 this is Eagle1, achieved waypoint SIERRA, over."*

6. Future Work and Conclusions

We are in the process of designing an evaluation of the Controller's Assistant to be conducted later this year. There are two hypotheses we would like to test. The first hypothesis is that providing a single, natural interface for a simulation operator will allow that operator greater span of control, lead to fewer errors, and grant more task efficiency in managing an airspace filled with a diverse range of CGF entities. Testing this hypothesis will entail enlisting participants to play the role of an airspace controller and to interact with a range of CGFs in representative tasks. In the control condition, the airspace controller will interact with CGFs using their native interfaces to control the CGFs in the same tasks. In the experimental condition, the airspace controller will interact with CGFs using the Controller's Assistant as a uniform interface to all the CGFS. The results would include a comparison of the control versus experimental conditions in areas of span of control, controller errors, and task efficiency. The second hypothesis is that the Controller's Assistant can effectively render a range of CGFs as equivalently capable in the perception of the controller. That is, regardless of the underlying capabilities of the CGFs in the experiment, the Controller's Assistant can exploit their capabilities and fill in the gaps where they are incapable. The experiment would be a kind of Turing Test for CGFs, to see if a human controller can detect any differences between the CGFs in the experiment. This would include a mix of both subjective (from the perspective of the participants) and objective metrics to measure perceived differences.

It is also conceivable to apply the Controller's Assistant concept to other domains besides airspace management. However, given the ways in which CGFs might be commanded in other domains and at other levels of aggregation, the concept might not map directly. ATC is, doctrinally, very formal, and that plays to the strengths of speech-based interfaces. However, in an environment where there is a more informal dialogue between entities (such as intra-team communication), such an interface might not be feasible or desirable.

One interesting transition application for our research is in the UAV domain. As UAV platforms become more automated, human controllers will interact with them directly. As with current CGF systems, UAVs will also have a range of capabilities that airspace controllers will not want to have to address individually. The Controller's Assistant could assume management of the fine-grained particulars of UAV capabilities, leaving the human controller to focus on managing the airspace.

7. Acknowledgements

This work is currently being funded under contract # W911W6-07-C-0052.

8. References

[1] Stent, A., Dowding, J. Gawron, J.M., Bratt, W.E., Moore, R. *The CommandTalk Spoken Dialogue System*. in *Proceedings of the Thirty-Seventh Annual Meeting of the ACL*. 1999. University of Maryland, College Park, MD: Association for Computational Linguistics.

[2] Cohen, P., M. Johnston, D. McGee, S. Oviatt, J. Pittman, I. Smith, L. Chen, and J. Clow. *QuickSet: multimodal interaction for simulation set-up and control*. In *Proceedings of the 5th Conf. on Natural*

Language Processing. 1997. Washington, DC: Morgan Kaufman.

[3] Jones, R., J.E. Laird, P.E. Nielsen, K. Coulter, P. Kenny, and F. Koss. *Automated Intelligent Pilots for Combat Flight Simulation*. In *Tenth Annual Conference on Innovative Applications of Artificial Intelligence*. 1999. Menlo Park, CA: AAAI Press.

[4] Taylor, G., B. Stensrud, S. Eitelman, C. Durham, and E. Harger. *Toward Automating Airspace Management*. In *Computational Intelligence for Security and Defense Applications (CISDA)*. 2007. Honolulu, HI: IEEE Press.

[5] Wray, R.E. and R.M. Jones, *An introduction to Soar as an agent architecture*. In *Cognition and Multi-agent Interaction: From Cognitive Modeling to Social Simulation.* 2005. Cambridge University Press: Cambridge, UK. p. 53-78.

[6] Nielsen, P.E., F. Koss, G. Taylor, and R.M. Jones. *Communication with Intelligent Agents*. In *Proceedings of IITSEC.* 2000. Orlando, FL.

Author Biographies

BRIAN STENSRUD, Ph.D., is a Research Scientist at Soar Technology and lead behavior developer for the Controller's Assistant IUI described in the paper. Dr. Stensrud received his Ph.D. in Computer Engineering from the University of Central Florida in 2005. He also holds B.S. degrees in Computer Engineering, Electrical Engineering, and Mathematics from the University of Florida. Brian has over seven years experience in the areas of knowledge-based systems and artificial intelligence.

GLENN TAYLOR is a Senior Scientist at Soar Technology and Principal Investigator for the Enhanced Computer Generated Forces and AutoATC projects. His research and development activities include agent-based systems, human behavior modeling, and natural dialogue systems. He received his B.S. in Computer Science in 1994 and his M.S. in Computer Science and Engineering in 1996, both from the University of Michigan.

BRADLEY SCHRICKER is a Senior Engineer with Dynetics, Inc., currently working on numerous projects pertaining to Unmanned Aerial Vehicle simulation. He has 10 years of experience in modeling and simulation, focused in the areas of distributed simulation, discrete event simulation, and behavior representation. Mr. Schricker received a B.S. in Computer Science with a minor in Mathematics from Florida State University in 1998 and an M.S. in Modeling and Simulation from the University of Central Florida in 2007.

JOHN MONTEFUSCO is a Software Engineer for Science Applications International Corporation (SAIC) in Huntsville, AL, working on the IDEEAS simulation in support of AMRDEC. He graduated in 2005 from the University of Alabama in Huntsville with a B.S. in Computer Science. He has worked on a number of simulation experiments both in constructive and distributed simulation.

JEFFREY MADDOX is a System Engineer for the US Army AMRDEC at Redstone Arsenal, Alabama, in the Advanced Prototyping, Engineering and eXperimentation (APEX) Lab. He earned a B.S. in Electrical Engineering and a B.S. in Math and Physics Education, all from Auburn University.