Trainable Automated Forces

Robert G. Abbott, Justin D. Basilico, Matthew R. Glickman, Jonathon Whetzel

Sandia National Laboratories

Albuquerque, NM

rgabbot@sandia.gov, jdbasil@sandia.gov, mrglick@sandia.gov, jhwhetz@sandia.gov

ABSTRACT

Simulation-based training requires realistic simulated friendly and opposing forces. Realistic graphics and physics alone are not enough; the tactics exhibited must be realistic for most learning to take place. Current approaches for driving the behavior of simulated forces include live human role-players, Semi-Automated Forces (SAFs), and intelligent/cognitive automated forces. Each of these approaches represents trade-offs between realism and various resource costs. Human role-players can provide maximal realism, but trained experts are a limited and potentially costly resource. Other approaches provide varying degrees of realism in exchange for varying costs associated with programming.

In this paper, we address another approach to simulated forces that aims to achieve increased realism at lower programming cost. Trainable Automated Forces (TAF) are computer-generated agents that mimic tactics demonstrated by human experts. First, a subject matter expert demonstrates the desired behavior (e.g., piloting an aircraft) in a simulator. Next, machine-learning algorithms are used to model the observed behavior. Finally, TAF controls a simulation entity using the model to predict what the human expert would do in the same situation. When TAF behaves incorrectly, the expert can step in to demonstrate the correct actions for the situation. This process can be repeated at any time with minimal help from technical experts, allowing TAF to generalize to a wider variety of situations over time.

We report here on the design and implementation of a prototype TAF capability, including both user interface design and experience with machine learning. In addition, we discuss the potential capabilities and limitations of TAF, surveying the inherent strengths and weaknesses of the general approach relative to other implementation techniques for automated forces in simulation-based training.

ABOUT THE AUTHORS

Robert G. Abbott is a Principal Member of the Technical Staff in the Cognitive Science & Applications group at Sandia National Laboratories in Albuquerque, NM, where his team develops software for automated behavior modeling. He holds a PhD in computer science from the University of New Mexico. He has been a member of the technical staff at Sandia since 1999. His current research focuses on automating the creation of human behavior models with the objectives of reduced cost and rapid development. Applications include trainable software agents to assume the roles of friendly and opposing forces, and automated student assessment for distributed virtual training environments. This line of research is supported primarily by the U.S. Navy and includes validation experiments with human subjects to assess the impact of new training technologies. Other research interests include distributed systems, security-related data mining, and computer vision.

Justin D. Basilico is a Senior Member of the Technical Staff in the Cognitive Science and Applications group at Sandia National Laboratories in Albuquerque, NM. He received his BA in computer science from Pomona College in 2002 and his MS in computer science from Brown University in 2004. He is the lead designer and developer of the Cognitive Foundry, a software platform for machine learning and cognitive simulation. His research interests include machine learning, information retrieval, user modeling, personalization, statistical text analysis, and human-computer interaction.

Matthew R. Glickman is a Senior Member of the Technical Staff in the Cognitive Science & Applications group at Sandia National Laboratories in Albuquerque, NM. He obtained his PhD in computer science from Carnegie Mellon University in 2001, focusing on search algorithms patterned after biological evolution. From 2001–2003, he was a postdoctoral fellow at the University of New Mexico, researching immunologically inspired computer-security architectures. Since joining Sandia in 2003, he has worked on projects spanning a variety of subject areas, including evolutionary optimization, cognitive modeling, autonomous character behaviors, behavioral simulation, and adaptive training systems. He has served as a technical reviewer for a variety of journals and conferences, including the *Machine Learning* journal and the *Journal of Machine Learning Research*.

Jonathan Whetzel is a Member of the Technical Staff in the Cognitive Science & Applications group at Sandia National Laboratories in Albuquerque, NM. He received an MS from Texas A&M University in computer science, with his graduate work focused on areas of machine learning and organizational psychology for improving training effectiveness in game environments. His research interests include serious games design, artificial intelligence, and cognitive science. He serves as a reviewer for the IEEE Games Innovation Conference and a chairperson for the Rio Grande International Game Developers Association (IGDA).

Trainable Automated Forces

Robert G. Abbott, Justin D. Basilico, Matthew R. Glickman, Jonathon Whetzel

Sandia National Laboratories

Albuquerque, NM

rgabbot@sandia.gov, jdbasil@sandia.gov, mrglick@sandia.gov, jhwhetz@sandia.gov

INTRODUCTION

Simulation-based training and mission rehearsal are increasingly important in military training. The high deployment rates and budgetary demands of ongoing operations create urgency in the requirement to deliver training anytime, anywhere, and at reduced cost. However, simulation-based training has not yet reached its full potential to meet these requirements. Dramatic improvements have been achieved in visual and physical fidelity, but realistic simulated human behavior for friendly and opposing forces is still an unsolved issue.

Human Role-Players

The most straightforward way to provide human behavior for simulation entities is with human role-players. Currently this approach is a primary driver of cost and complexity in staging simulation-based training exercises. Hiring numerous role-playing contractors is expensive, and some do not have sufficient operational experience. Each role player requires that additional equipment be procured, configured, transported, and maintained. Using military personnel to fill "walk-on-roles" provides little training benefit to them, resulting in frustration and wasted resources. Networked simulations alleviate the need for everybody to meet in one place, but schedules must still be coordinated. Furthermore, high-speed networks are not universally available and raise costly security and configuration issues.

Semi-Automated Forces (SAF)

SAFs address the need for reduced-manpower simulation. SAF tools such as JSAF, OTB, and OOS allow entity behavior to be specified ahead of time.

At a computational level, SAF behavior specifications typically amount to some form of finite state machine (FSM). An FSM can be thought of as (1) a set of states, each of which corresponds to some behavioral state (e.g., patrol along a given path); and (2) a set of transitions between states (e.g., when an intruder is detected, move to confront).

Advantages of FSM-style SAF behaviors include clarity and predictability. FSMs are particularly good for implementing well-defined doctrinal behavior of limited complexity. However, SAFs have a limited capability to respond dynamically to changing circumstances. As allies, SAFs have little capability for coordinating or communicating with students. As enemies, SAFs are often little more than target drones. They do not model an adaptive, thinking enemy. Scenarios with such scripted behaviors have a very short useful life because students quickly learn to anticipate scenario events. Concurrency is also problematic, as FSMs are made obsolete by changing tactics, techniques, and procedures (TTPs) and require technical expertise to reprogram.

Intelligent Automated Forces

Intelligent automated forces go beyond conventional SAFs with the ability to generate behavior dynamically in response to simulation events. Examples are TacAir-Soar (Coulter et al., 1998) and ACT-R agents (Best, Scarpinatto, & Lebiere, 2002) for military operations in urban terrain (MOUT). The cognitive architectures underlying these capabilities are informed by a large body of accumulated psychological research. When used properly, these approaches can realistically mimic many aspects of cognition, particularly with respect to resource constraints such as reaction time and attention.

However, implementing intelligent automated forces is a challenging task. The conventional process for creating intelligent automated forces requires a pipeline of specialists: subject matter experts (SMEs), knowledge engineers, software engineers, and validation testers. Some of the SMEs' intent is lost at each stage of the pipeline. The number of specialties required makes coordination difficult, so the implementers may have little opportunity to interact

with the SMEs. Changing TTPs and scenarios may require significant reprogramming.

SME specification of tactics is especially difficult because spatiotemporal tasks are driven by procedural (also known as implicit) knowledge. Kornecki, Hilburn, Diefenbach, and Towhidnajad (1993) describe the difficulty of producing an explicit rule set for air traffic control (ATC): "In real ATC sectors, a controller's actions are based on a subjective evaluation of the situation and a rough mental projection of the aircraft courses and computation of a future possible separation violation. There is no extensive or precise arithmetic computation involving the geometric relation between objects in three-dimensional (3-D) space."

Recent tools, such as the Office of Naval Research–sponsored Discovery Machine provide a more user-friendly interface to help SMEs encode their own knowledge (see http://www.discoverymachine.com). Though useful, this approach still calls upon an SME to provide rules for all relevant aspects of behavior, a task which amounts to programming. Since an SME produces the rules outside the context of performing the task, the context recognition process that drives expert behavior is not operational. The resulting model captures what an SME thinks a person should do, rather than what the SME actually does.

Trainable Automated Forces (TAF)

Depicted in Figure 1, our vision is for SMEs, such as instructors, to train synthetic forces directly by demonstration, as in training human students. Technical experts (e.g., computer programmers) must initially implement TAF for each type of role player required. Subsequently, however, SMEs can directly interact with TAF to enhance the domain expertise of TAF over time, without further support from a technical expert. TAF then relieves the SME of role playing so that the expertise of a single SME can be shared with any number of students. In our vision, the sharp distinction between the construction and operational phases (as required for traditional expert systems) is blurred. If an instructor recognizes a skills gap during one exercise, the instructor should be able to alter the behavior of automated forces in the next exercise to address the gap. Ideally, the long and expensive development pipeline can be virtually eliminated. This is the TAF approach.

TAF training is an ongoing interaction between the instructor and the role-playing agent. The interaction is based on demonstrations of correct behavior by the instructor, and demonstrations by the system of its current understanding. Our goal is for the instructor to be able to interrupt and correct TAF when its actions diverge from the instructor's intent. TAF learns from such corrections and will not repeat the same mistake. Because this approach is data driven, objective behavior validation is more feasible compared with other approaches for automated forces.

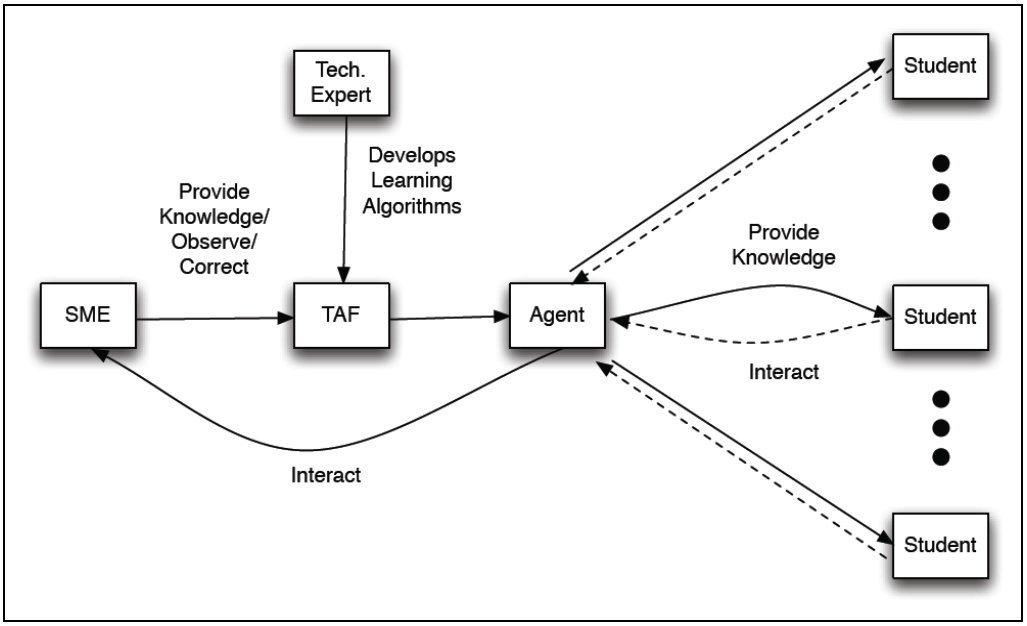

Figure 1. Training of Synthetic Forces by Demonstration

Related Work

TAF is based on behavioral cloning, which is an established technique for building agent behaviors. Widrow and Smith (1964) first studied the technique for the pole-balancing (or inverted pendulum) task. The term "cloning" implies that the agent simply replays previously recorded behavior, but most applications of behavioral cloning have some capability to generalize to new situations. Nevertheless, the performance of a clone will degrade as it encounters situations that are dramatically different from any encountered during training (Bratko, 1997). Behavioral cloning has been successfully applied in simulations of tasks such as piloting an airplane (Morales & Sammut, 1994), operating a crane (Suc & Bratko, 1999), and riding a bicycle (Suc & Bratko, 2000). Similar techniques are popular within robotics and are known as learning by observation or learning by imitation. Schaal, Ijspeert, and Billard (2004) provide an overview of learning by observation for robotics, including juggling, table tennis, and dance. However, these approaches do not necessarily include an *interactive* process for refining agent behavior, as does TAF.

To build an expert model, TAF must be able to observe experts performing relevant tasks. In particular, the system needs data that capture the stimuli and responses that characterize expert behavior. Although training simulations are inherently computer based, integration with complex preexisting applications is nontrivial. The training application must output sufficiently detailed information about its internal state. Popular network protocols for exchanging simulation state include Distributed Interactive Simulation (DIS) (Little, 1994) and (more recently) High Level Architecture (HLA) (Calvin, 1996). Both protocols transmit basic information, such as entity position updates. However, terrain information (ground elevation, appearance, and physical properties) is too large to transmit over the network in real time, and most simulators do not transmit detailed information, such as instrumentation in an airplane cockpit. TAF cannot learn correct behavior if the behavior depends on missing information.

TAF Technical Description

Implementing TAF consists of creating a role and then populating the role with example behavior. The role consists of information (e.g., from a nine-line brief) such as the location of a target and the time on target. Any information that cannot be gleaned from the inputs specified for the role cannot influence the behavior of TAF; thus incomplete information may lead to incorrect behavior. However, extraneous information may also degrade performance by confusing TAF. If training data are relatively sparse (i.e., the number of inputs is large relative to the amount of training data provided), spurious patterns are likely to be found in the training data, and learning these patterns will lead to unpredictable TAF behavior.

This section describes the current implementation of TAF, addressing the numbered and unnumbered labels in the block diagram in Figure 2. Airstrike piloting is used as a running example.

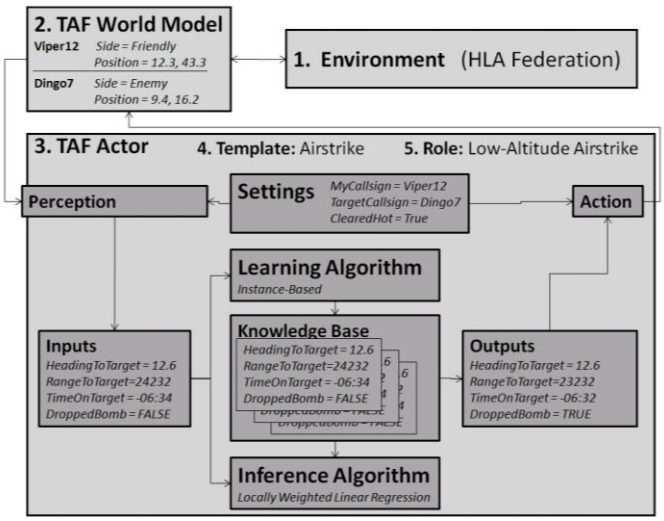

Figure 2. Internal Structure of TAF Actor

Environment

The *environment* is not part of TAF; it is the simulation in which TAF actors exist. (As with any SAF technology, TAF could also participate in live/virtual/constructive simulations and conceivably control robotic agents in the real world). To produce realistic behaviors, a simulation environment must simulate everything that influences decision making in reality. To support incremental refinement of behavior models, the environment must allow control of an entity to be handed back and forth between TAF and a human controller.

TAF World Model

The *TAF world model* provides a standard interface between TAF agents and the environment. The TAF world model presents ground-truth information about all known entities (whether or not these entities are controlled by TAF) and the current time. Each entity has a name, position, velocity, etc. The world model is shared by all TAF actors in a simulation. The world model in Figure 2 shows the callsign, side (friendly/enemy), and position of two entities—a strike aircraft (Viper12) and a target (Dingo7).

A specialized TAF world model is created for each type of environment. Currently, there are two world models: the HLA TAF world model, which allows TAF to interface with the distributed virtual training environment (DVTE) in Figure 4, and the Umbra world model (Figure 3). The HLA TAF world model allows TAF to control a variety of entities in the DVTE simulation environment. Umbra is a modeling and simulation framework developed at Sandia National Laboratories that is used in an augmented-reality training simulator for training dismounted security forces.

In the augmented-reality simulation depicted in Figure 3, participants with red helmets walk through a physical environment wearing goggles to overlay the actual view with simulated forces, walls, etc. (inset at lower left). Through the goggles the participants see a video of the same room with the computer-animated objects shown in the inset added to the scene.

Figure 3. Interactions with TAF Agents in Augmented Reality

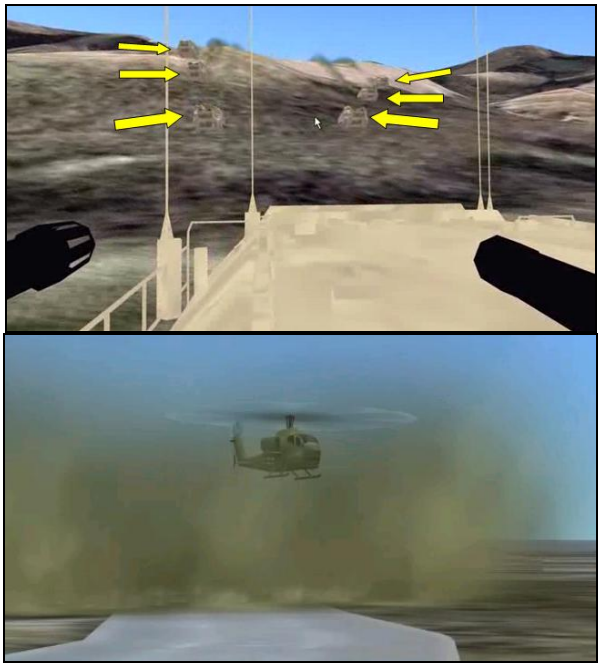

Figure 4. Distributed Virtual Training Environment (DVTE). Above, several TAF-controlled vehicles follow a human leader. Below, TAF pilots a helicopter.

Settings

The objective of TAF is to learn patterns of behavior that are apparent mainly through the actions of entities. However, the user must manually input some information, such as, at a minimum, specifying an entity for TAF to control. In TAF, these inputs are called *settings*. Each role has a different set of settings.

In the airstrike role, two of the settings are TargetCallsign and TimeOnTarget. The TAF graphical user interface (GUI) includes a panel (Figure 5) for specifying the settings.

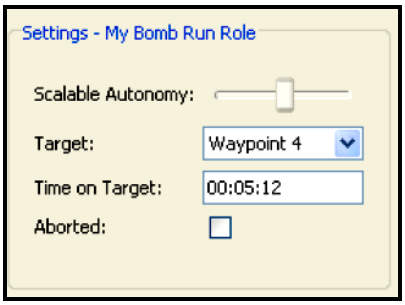

Figure 5. TAF Settings Panel

Perception

The world model presents ground-truth information from an allocentric (or global) perspective. Each TAF actor must be provided with a perception model to filter and adapt this information for its own needs. Thus the perceptive needs for a role must match the perceptive capabilities of a given world model for that role to be applicable.

For the airstrike role in Figure 2, the relevant information includes the heading and range to the target and the time on target (relative to the current time). To determine this information, the TAF actor must know its own identity and that of the target; these are settings specified by the user. *Perception* then transforms the allocentric information provided by the world model to egocentric inputs for the learning or inference algorithm. In the airstrike example, perception retrieves the position of self and target and computes the distance between them, the heading to the target, etc. The assumption of modeling from an egocentric perspective is that behavior is determined relative to other entities in the simulation rather than by cardinal directions, e.g., following a heading of 0 degree relative to the target will always lead TAF to the target.

Inputs

As shown in Figure 2, information from the world model and setting is filtered and transformed by perception to form the *input* for TAF's *learning and inference algorithms*. The completeness (or incompleteness) of the input is a constraint on the realism of TAF behavior. Several factors prevent TAF from receiving a complete set of inputs, leading to degraded behavior.

First, TAF lacks the instinct and knowledge that humans possess as a result of evolution and life experience.

Second, the environment provides insufficient information. Simulation environments are not reality; they only contain whatever their implementers have put into them. Human role-players sometimes "fill in the blanks," using side agreements and their imaginations.

Of course, humans often do not have access to all the information they need, especially in combat. Enemies use secrecy, subterfuge, and attack command-and-control assets to evade, mislead, and blind each other. In an HLA simulation, TAF receives ground-truth information that may be unavailable to human combatants, granting TAF (in effect) a 360-degree field of view through walls, mountains, water, jammers, and darkness. In practice, software agents may use this unfair advantage to counter their own inherent limitations in perception and intelligence, with uneven results. More realistic perception models can also be implemented if necessary.

Role Template

The *role template* specifies the learning algorithm, inputs, and outputs. The role template is implemented by the technical expert and not modified by the end user, so the set of role templates defines the general limits of what TAF can learn.

For example, the airstrike template in Figure 2 would list inputs and outputs by name and type (e.g., HeadingToTarget and RangeToTarget as real-valued numbers), but these inputs and outputs are not linked to specific data streams in the simulation until the role is instantiated and the callsigns of the TAF agent and the target are entered.

The role template also specifies a learning algorithm. The current implementation of TAF uses the open source Sandia Cognitive Foundry (Basilico, Benz, & Dixon, 2008; http://foundry.sandia.gov), which supplies a wide range of learning algorithms, e.g., linear regression, nearest neighbor with locally weighted linear regression, ID4 rule induction, support vector machines, and backpropagation for neural networks. The role template also specifies the type of knowledge base for each algorithm (e.g., learned link weights to parameterize a neural network, or the rule set created by ID4).

Role

The specific actions to be taken by a TAF actor are determined by its *role*. The role template specifies the learning algorithm, inputs, and outputs but does not

specify a mapping from inputs (situations) to outputs (actions). The role provides this mapping.

To create a role, the user supplies a descriptive name for the role and then supplies examples of desired behavior in the simulator, along with the settings of the role that are applicable to the behavior.

Each algorithm learns a different set of parameters from the data (e.g., link weights derived for a neural network, or the rule set created by ID4). This is the knowledge base, which is empty until the desired behavior is demonstrated and refined

IMPLEMENTATION

A prototype of the core TAF architecture described above is implemented in Java and includes world models for HLA and for Umbra, The TAF software is freely available for U.S. government applications.

HLA Interface

We integrated TAF with the DVTE via the HLA protocol in collaboration with Lockheed Martin Simulation, Training & Support. HLA was an attractive means of integration for several reasons. First, HLA provides a suitable level of information detail for the TAF world model (e.g., entity state updates over time). Second, HLA integration facilitates the application of TAF in other HLA-based simulations. Third, interfacing with a simulation via network data exchange (rather than, say, direct linkage against an application programming interface) minimizes software dependencies and maximizes modularity.

However, dynamically handing off control between TAF and a human supervisor is more technically difficult than anticipated. HLA-standard techniques for entity hand-off are typically not implemented or not available due to the use of an unsynchronized ("connectionless") mode to conserve network bandwidth. Also, manual intervention in TAF control (e.g., moving an entity from the wrong trajectory) interferes with the simulator's internal dynamics model (position and speed) and thus may impart enormous momentum on the simulation entity.

GUI Design

Although the goal of TAF is to interact with users mainly through the 3-D virtual environment provided by a training simulator, TAF also relies on a conventional GUI, implemented using the open-source NetBeans Platform, for certain operations (Figure 6). These operations are (1) creating new TAF instances to control simulation entities, (2) specifying parameters for the TAF entity, and (3) executive control such as entity hand-off between TAF and the human supervisor.

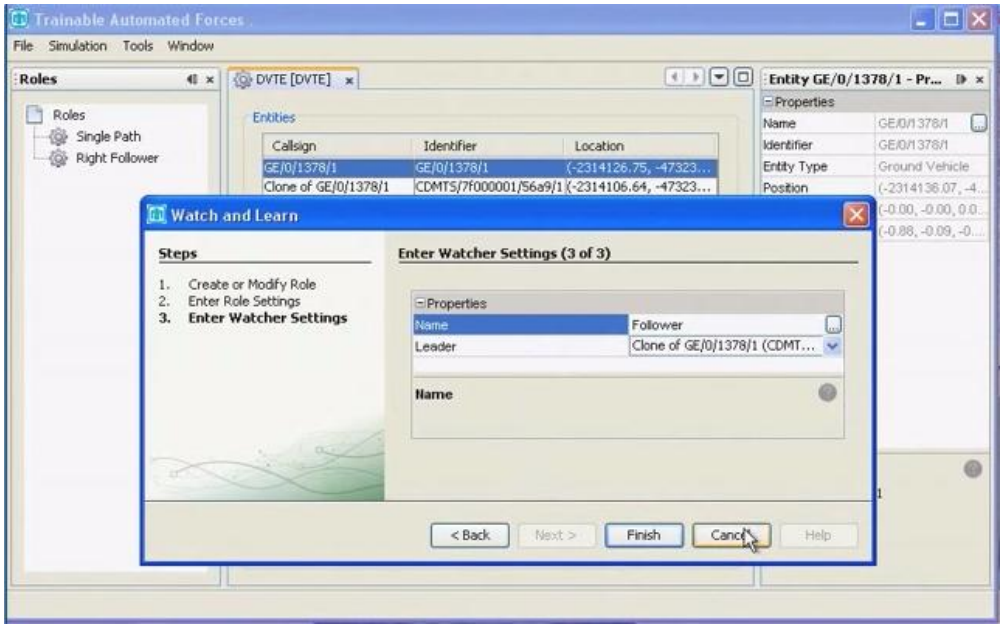

Figure 6. Example of TAF GUI Setting Parameters, e.g., Identity of the Entity to Control

Implemented Role Templates

Single-Path

We began development with the most basic use case: rote replay. During observation, the single-path drone simply records the exact sequence of positions and orientations that a designated entity exhibits over time. In the operational phase, the drone simply "replays" the behavioral sequence. This functionality is similar to After Action Review (AAR) logging, but for a single entity with finer-grained control and meant for replay into another simulation. Although this capability is simple (no generalization), it is not commonly available in training simulations and could potentially be quite useful.

Dynamic Follower

The next step in development was a role that dynamically responds to inputs from the environment. This is the "follower" role, which provides the basis, for example, of a wingman or column of vehicles that follows a leader in a dynamic path. In this role, TAF learns parameters that capture information such as (a) the time and distance to lag behind the leader and (b) the bearing to maintain relative to the leader.

Two settings are necessary to begin training a follower-role instantiation: the entity ID of the self (i.e., the follower) and the entity ID of the leader. Given these settings and one or more sample behavior sequences, the system derives a series of training examples that pair the position of the follower relative to the leader at a given time step with a position and orientation at the subsequent time step.

A given instantiation of the follower role thus determines how to move relative to a leader. One instantiation might represent the goal of following behind a leader at a close distance, while another might represent following farther behind and to the left. Figure 4, presented previously, shows six TAF instances, with three following to the left and three to the right, thus forming a chevron formation.

Replacement and Branching

Beyond learning behavior from a given set of sample behavior sequences, TAF must support "iterative refinement": when a human supervisor observes suboptimal TAF behavior, the supervisor can intervene and further refine the learned behavior by demonstration. Implementation of iterative refinement presents additional challenges related both to learning and to arriving at an effective user interface.

Our first milestone toward iterative refinement was a rewind-and-replace capability applied to the single-path drone. Pressing this button when observing a TAF-driven entity behaving autonomously rewinds the entity's position and orientation to where it was 30 seconds ago and hands control of the entity to the supervisor. The rest of the recorded path is discarded and learning (or recording, in this case) begins again.

We have extended beyond rewind-and-replace to implement a prototype branching capability whereby newly demonstrated behavior is added to the model rather than replacing older behavior. Instead of simply replaying a recorded temporal sequence of entity states observed in the past, at each time step the new role chooses the action previously performed by a human under the most similar conditions. With this role, an entity can learn conditional behavior, e.g., execute a prescribed path when no enemy is present but move to intercept when an enemy appears.

TAF for Dismounted Operations

We have further bridged TAF with an augmented reality trainer for Close Quarters Combat based upon Sandia's Umbra modular simulator. Instead of HLA, TAF is linked with Umbra via a JNI bridge between Java and C++. We have thus far experimented with two role templates: the single-path drone and the follower. One of the most exciting parts of this application is the potential of capturing highly implicit procedural knowledge via direct perception of human movement in real 3-D space.

DISCUSSION

Use of the Machine Learning Package in Sandia's Cognitive Foundry makes it straightforward to implement roles by experimenting with different learning algorithms. To this point, we have found instance-based learning methods to be particularly advantageous for iterative refinement because when a user identifies incorrect behavior, it is straightforward to identify and replace the data that are responsible.

Difficulties in implementation may be broadly grouped under two headings: (1) more conceptual issues in learning from observation and (2) more pragmatic issues of system integration.

On the learning side, there is the general issue of machine perception versus human perception.

The system constraints often make it difficult for TAF to perceive the same features of the environment that humans do. In many cases, this limitation may not have a strong impact. In some cases, TAF may be able to compensate by using additional information that is not available to human role players. For some target behaviors, however, this limitation may be a limiting obstacle. Further research is necessary to illuminate which classes of behaviors are more amenable or less so to TAF's learning-by-demonstration approach.

In addition, one of our goals was that the TAF-defined roles could be applicable to multiple simulators and world-model implementations. To achieve this generality, we have mostly employed simple world models consisting of entities in 3-D space with positional information. While these features were sufficient for the simple roles we developed, they are also limiting because other information about the entities or environment was not used. Simple, position-based world models have also led to control of the entities via the same absolute coordinate system without any simulation of the output of the control. For example, TAF learns to control a simulated airplane by observing its trajectory in 3-D coordinate space rather than by observing the motion of a joystick controller. Improved interfaces and standard implementations for perception and control that tie deeper into the underlying simulation would allow the role implementations to focus more on the aspects of the learning algorithms and less on the mechanics of creating the appropriate perceptions and ensuring that the agent's actions are reasonable.

In the pragmatic realm, we have had issues related to HLA technicalities, such as the hand-off of control of an entity between two members of the federation and also the inability to access a sensitive aerodynamics model. So far, the lesson we are taking away from this experience is that bolting a fundamental capability like TAF onto a preexisting architecture presents its own set of challenges that must be explicitly considered, especially with regards to entity ownership.

In general, there is a trade-off between power and ease-of-use in human-computer interaction. While TAF has a lot of promise, our experience in this and in many other systems suggests that getting this balance right may be decisive in determining the ultimate utility of the system.

ACKNOWLEDGEMENTS

This work is supported in part by the Office of Naval Research Grant N0001408C0186, the Next-generation Expeditionary Warfare Intelligent Training (NEW-IT) program. Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy's National Nuclear Security Administration under Contract DE-AC04-94AL85000. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Office of Naval Research, Sandia National Laboratories, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for government purposes notwithstanding any copyright notation hereon. We further wish to acknowledge excellent contributions to this work on the part of colleagues Damon Gerhardt, Eric Goodman, Kiran Lakkaraju, and Kevin Dixon as well as help from Lockheed Martin Simulation, Training & Support with the HLA hand-off.

REFERENCES

Basilico, J., Benz, Z., & Dixon, K. R. (2008). The Cognitive Foundry: A flexible platform for intelligent agent modeling. In *Proceedings of the 2008 Behavior Representation in Modeling and Simulation (BRIMS) Conference*, Providence, RI.

Best, B., Scarpinatto, C., & Lebiere, C. (2002). Modeling synthetic opponents in MOUT training simulations using the ACT-R cognitive architecture. *Proceedings of the 11th Conference on Computer Generated Forces and Behavior Representation*. Orlando, FL: University of Central Florida.

Bratko, I. (1997). Qualitative reconstruction of control skill. In *Proc. QR'07 (11th Int. Workshop on Qualitative Reasoning)* (pp. 41–52). Pavia, Italy: Instituto di Analisi Numerica.

Calvin, J., & Weatherly, R. (1996). An introduction to the high level architecture (HLA) runtime infrastructure (RTI). In *Proceedings of the 14th Workshop on Standards for the Interoperability of Defense Simulations* (pp. 705–715), Orlando, FL.

Coulter, K., Jones, R., Kenny, P., Koss, F., Laird, J. and Nielsen, P. (1998). Integrating intelligent computer generated forces in distributed simulations: TacAir-Soar. In *STOW-97, Proceedings of the 1998 Spring Simulation Interoperability Workshop*, Orlando, FL.

Kornecki, A., Hilburn, T., Diefenbach, T., & Towhidnajad, M. (1993). Intelligent tutoring issues for ATC training system. *IEEE Transactions on Control Systems Technology 1*(3), 204–211.

Little. R. (1994). Architectures for distributed interactive simulation. In *Advances in Modelling and Simulation Conference*, Redstone Arsenal, AL.

Morales, E. F., & Sammut, C. (2004). Learning to fly by combining reinforcement learning with behavioural cloning. In C. E. Brodley (Ed.), *ICML '04: Proceedings of the Twenty-first International Conference on Machine Learning* (p. 76). New York: ACM Press.

Schaal, S., Ijspeert, A., & Billard, A. (2004). Computational approaches to motor learning by imitation. In C. D. Frith & D. Wolpert (Eds.), *The Neuroscience of Social Interaction* (pp. 199–218). Oxford: Oxford University Press.

Suc, D., & Bratko, I. (1999). Symbolic and qualitative reconstruction of control skill. *Electronic Transactions on Artificial Intelligence, 3*, Section B, 1–22.

Suc, D., & Bratko, I. (2000). Qualitative trees applied to bicycle riding. *Electronic Transactions on Artificial Intelligence, 4*, Section B, 125–140.

Widrow, B., & Smith, F. W. (1964). Pattern recognizing control systems. In J. T. Tou & R. Wilcox, *1963 Comp. and Inf. Sciences (COINS) Symp. Proc* (pp. 288–317). Washington: DC: Spartan.