Formalized Behavior Models for MOUT OPFOR Individual Combatant Weapon Firing

Richard Stottler
Stottler Henke Associates, Inc.
San Mateo, CA
stottler@shai.com

Stephanie Lackey
NAVAIR TSD
Orlando, FL
stephanie.lackey@navy.mil

John Brian Kirby
Stottler Henke Associates, Inc.
San Mateo, CA
kirby@shai.com

ABSTRACT

This paper describes techniques to augment the behavioral models of automated Opposing Forces (OPFOR) Individual Combatants (ICs) with a realistic, practical set of weapon firing behaviors for virtual Military Operations in Urban Terrain (MOUT) training environments. These behaviors represent a formalization of target acquisition and firing execution tactics and techniques based on doctrine, input from subject matter experts, and observations from the field. The formalisms are based on Behavior Transition Networks (BTNs), an extension of Finite State Machines (FSMs). A COTS toolkit allows for rapid visual behavior specification, testing and modification, easy simulation integration, and flexible architectures. The behaviors are designed hierarchically so that the actions and goals of a human combatant can be modeled at various levels of detail. Polymorphism is used heavily to alter the behaviors based on the type and current state of the IC at all levels of the model. For example, an IC just exposed to a stun grenade behaves very differently from one who has not been so exposed. A motivated, well-trained, elite OPFOR IC behaves very differently from a conscript. Uncertainty is incorporated both in the initial selection of attributes (boldness, speed, aiming accuracy, etc.) of the ICs to give their behaviors natural variation and in runtime execution of decision making to keep even a single IC from being too predictable. This paper also describes the behavior modeling process itself from knowledge engineering to formalization and implementation through validation. The initial prototype is described in which IC behaviors are implemented and interfaced to a real-time IC simulation.

ABOUT THE AUTHORS

Richard Stottler co-founded Stottler Henke Associates, Inc., an artificial intelligence consulting firm in San Mateo, California, in 1988 and has been the president of the company since then. He has been the principal investigator on a large number of tactical decision-making intelligent tutoring system projects conducted by Stottler Henke including projects for the Navy, Army, Air Force and Marine Corps. Currently he is working on OPFOR MOUT Individual Combatant targeting and firing behavior modeling for the Marine Corps and a Combined Arms ITS as part of the US Marine Corps Combined Arms Command and Control Training Upgrade System (CACCTUS). He has a Masters degree in Computer Science from Stanford University.

Stephanie Lackey is a Computer Engineer in the Science and Technology Division of the Naval Air Systems Command, Training Systems Division (NAVAIR TSD), in Orlando, Florida. She serves as Principal Investigator (PI) for the CGF evaluation effort within the Virtual Technologies and Environments program and as Co-PI for the Intelligent Training Support Tools (ITST) portion of the Air Warfare Training Development (AWTD) program. Stephanie earned her M.S. degree in Industrial Engineering from the University of Central Florida and is pursuing her Ph.D. in the same field.

John Brian Kirby graduated in 2003 from the University of California, Berkeley, with simultaneous Bachelor degrees in Computer Science and Cognitive Science. He has worked as an AI software engineer at Stottler Henke since graduation. As a member of the Stottler Henke team, he has contributed to OPFOR MOUT individual combatant targeting and firing behavior modeling for the Marine Corps, storage optimization and scheduling for NASA's Kennedy Space Center and aircraft carrier VLA lighting design for the Navy.

Formalized Behavior Models for MOUT OPFOR Individual Combatant Weapon Firing

Richard Stottler
Stottler Henke Associates, Inc.
San Mateo, CA
stottler@shai.com

Stephanie Lackey
NAVAIR TSD
Orlando, FL
stephanie.lackey@navy.mil

John Brian Kirby
Stottler Henke Associates, Inc.
San Mateo, CA
kirby@shai.com

PROBLEM DESCRIPTION

The need to address skills required for the Military Operations in Urban Terrain (MOUT) environment results from the dramatic shift in military tactics since the end of the Cold War. American military forces justifiably avoid combat in cities if at all possible, but the nature of combat has changed requiring combat forces to enter urban environments as a matter of course. This trend shows little sign of decreasing (Miles, 2003) since centers of power are located in cities and a projected 70 percent of the world population will occupy urban environments in the near future (Schwierz, Krenz, & Lipke, 2003). Although Desert Storm demonstrated the ability of U.S. forces to gather battlefield intelligence and target enemy forces at extended ranges, the events in Mogadishu, Somalia, two years later demonstrated the challenges of fighting in the post-Cold War era and revealed the clear need for improved tactics and training (Slear, 2004).

A crucial component of MOUT training is live training – combat towns. Live training itself presents challenges to the military including: (1) cost of production and maintenance and (2) availability of materials, facilities, and personnel. Unfortunately, some skills are difficult to train in a live environment due to safety issues and complexity levels. These issues are compounded in live MOUT training where the primary weapon is "the inherent complexity of an urban area" (Slear, 2004) and the nature of the mission depends on the opposing force that is intimately familiar with that urban area. These issues lead to the following questions: *How many combat towns can you build in order to meet the ever-increasing need for MOUT training? What is the best method to replicate opposing forces (OPFOR)? What is the training value for role-players?*

The modeling and simulation community has responded to these issues (Lyons, Schmorrow, Cohn, & Lackey, 2002) by providing virtual tools such as Synthetic Natural Environments (SNE) to simulate the real world, platform simulators to replace vehicles, and a variety of technologies to simulate the behavior of Individual Combatants (IC) including Semi-Automated Forces (SAFs) and Computer Generated Forces (CGFs). Traditional research has focused on SNEs and platform simulators. However, recent technological developments provide a greater opportunity to address the level of realism exhibited by simulated ICs.

Virtual Technologies and Environments (VIRTE) is an advanced research and development initiative sponsored by the Office of Naval Research focusing on virtual technologies for MOUT training. Of particular interest to the U.S. Navy and Marine Corps is the ability to accurately model weapon firing behaviors of OPFOR entities. In order for effective virtual MOUT training to occur, accurate weapon modeling issues must be addressed. The development of weapon firing behaviors is crucial to advancing the science of simulated training and to address the needs of our warfighters in the evolving global combat environment.

REQUIREMENTS FOR MOUT OPFOR IC WEAPON BEHAVIORS

MOUT training requirements and learning objectives dictate in what types of situations automated OPFOR ICs must realistically behave, what types of actions they should be able to perform, and to what types of stimuli they should react. For example, building clearing procedures are the most crucial of the MOUT activities that should be trained, especially clearing individual rooms. Therefore, automated OPFOR ICs must behave realistically when defending the inside of a building and specifically inside a room to be cleared. US soldier trainees clearing a room may breech a wall or a door; may use a stun, compression or fragmentation hand grenade; might talk or make other noises outside the room; and of course, will enter the room and engage enemy found in this room. Therefore, the automated OPFOR ICs must react realistically to each of these stimuli. How they react will depend on their training, experience, motivation, and other factors. Reacting realistically implies that they have a number of actions available to them. In

addition to firing on the trainees bursting into the room (with varying styles and degrees of accuracy), the automated OPFOR ICs may seek cover (especially from a grenade or in anticipation of attack), reload, surrender, move forward to ambush from a hidden corner, and possibly drop the magazine during reloading (especially if nervous and inexperienced). Many of these actions support explicit learning objectives. In fact, the automated OPFOR fire reinforces the movement patterns of the soldiers as they enter the room, it ensures that they maintain their sectors of fire, and emphasizes that they engage the enemy quickly and accurately. The OPFOR ambushing actions reinforce the soldier's requirements to maintain security on obscured portions of the room. An explicit learning objective is to make sure when soldiers employ grenades that they are thrown into the room hard enough that they bounce around so that they cannot be easily grabbed and tossed back. This implies that at least some of the automated OPFOR should be capable of grabbing and tossing back a poorly thrown grenade.

Realistic weapon firing behavior also involves many other aspects. The weapon handling behavior must be able to realistically choose a weapon (when more than one is available). Then, making sure it is ready, it must choose a firing mode (i.e. burst, single shot) and posture (i.e. prone, standing, etc.), change the movement state if necessary (i.e. stop or slow down), select a target when more than one exists, turn to face the target with realistic speed, possibly coordinate fire with other team members, aim with realistic accuracy, select an aim point, decide on number of shots (i.e. one or two), fire the weapon, assess the results, and possibly re-engage.

In addition to behaving realistically inside the room being, or about to be cleared, automated OPFORs must also behave realistically in hallways, at different types of hallway intersections, at stairwells, and in large indoor open areas, such as in warehouses. This realism includes both defending and ambushing behavior. A complete automated OPFOR would also behave appropriately outdoors either offensively or defensively.

USE OF BTNS FOR BEHAVIOR MODELING

The architecture for authoring weapon firing behaviors and executing them in a simulation is based on the SimBionic toolkit. This toolkit utilizes the concept of behavior transition networks (BTNs), which are generalizations of finite state machines (FSMs). It was originally developed to provide intelligent simulation behavior inside military training simulations and commercial games. BTNs have current states and transitions like finite state machines, but also hierarchically decompose. They can have variables, communicate to each other through a blackboard, and can execute arbitrary perceptual or action-oriented code. A BTN state is called a "node" and consists of either an action or a sub-behavior. A large number of these can run in parallel. When used in conjunction with a simulation, the BTNs interface via a sophisticated native runtime engine that has been successfully demonstrated with a variety of real-time simulations.

In the authoring environment, behaviors are constructed by "drawing" them as flow-chart diagrams. Specifically, actions are represented as rectangles, sub-behaviors as boldfaced rectangles, conditions as ovals, and connectors (transitions) as arrows. Unlimited variable assignments, complex expressions, and explanatory comments can be added to any of these elements. Figure 1 shows a sample BTN containing actions, conditions, and connections. This is an actual behavior from the set of tactical behaviors developed for the initial prototype. This includes calls to control actions and sense events in the simulation environment.

Figure 1 also shows the Nervous *EngageEnemy* behavior. Notice the tabs at the bottom of the main construction panel: another *EngageEnemy()* behavior exists for Calm combatants. In this example, the sub-behavior *DetermineAccuracy()* binds the variable *accuracyVaration,* which is the amount to perturb the OPFOR's aim, which is passed to the action *TurnToTarget()*. The *TurnToTarget()* action is a direct command sent from this behavior to the simulation entity during execution. In a condition in this transition is the *RemainingAmmoInClip()* predicate, which gets the information from the simulation. If the *RemainingAmmoInClip()* is 0, then the left-most transition cannot be followed and the *ClipEmpty()* sub-behavior will be executed. The *ClipEmpty()* behavior is defined elsewhere in the same tactical module, and as such, it appears as a darker rectangle (and appears on the behavior menu on the left side of the screen) and can be called from any behavior in this module. In the run-time engine, each Artificially Intelligent (AI) directed entity associates with one or more behaviors that dictate how it will act in the simulation environment. Behaviors are represented as hierarchies of BTNs, which consist of two types of elements, nodes and transitions. A node in a BTN represents an action that the entity may possibly perform at some

point during the simulation. Two nodes in a BTN are of special significance. The *current node* of a BTN denotes the action currently being carried out by the associated entity. A given BTN may have exactly one current node at a time. Note that the actions represented by a node may be concrete and/or primitive – such as, *FireWeapon()* or *DoNothing(3 seconds)*. Or, they may be more abstract and complex – for example, *FindNearestEnemy()*. An action may also represent a deliberative or perceptual activity that has no direct physical effect on the simulation environment. Primitive actions tend to directly interact with the simulation engine through API calls, while complex actions are generally carried out by sub-BTNs or specialized behavioral modules.

As stated above, BTNs may be hierarchical. Any node in a BTN may link to another arbitrary BTN or set of BTNs. When a node with such links becomes current, execution passes to one of the linked BTNs, which begins at its initial node. By using a hierarchical structure, it is easier to abstract or adapt individual lower level tactical or decision-making components to fit a given simulation without major re-programming requirements. We are constructing a hierarchical breakdown of the subtasks associated with target acquisition and firing execution so that high level tasks and decisions can be expressed in terms of abstractions of lower level tasks and decisions. This provides a basis for implementation and also allows for a modular development process where lower level behavior elements may be elaborated or simplified in accordance with iterative development conclusions regarding their effective level of realism. These levels of abstraction also make the model easier to understand. In the current prototype there are dozens of behaviors, some of which are 5 to 6 layers deep.

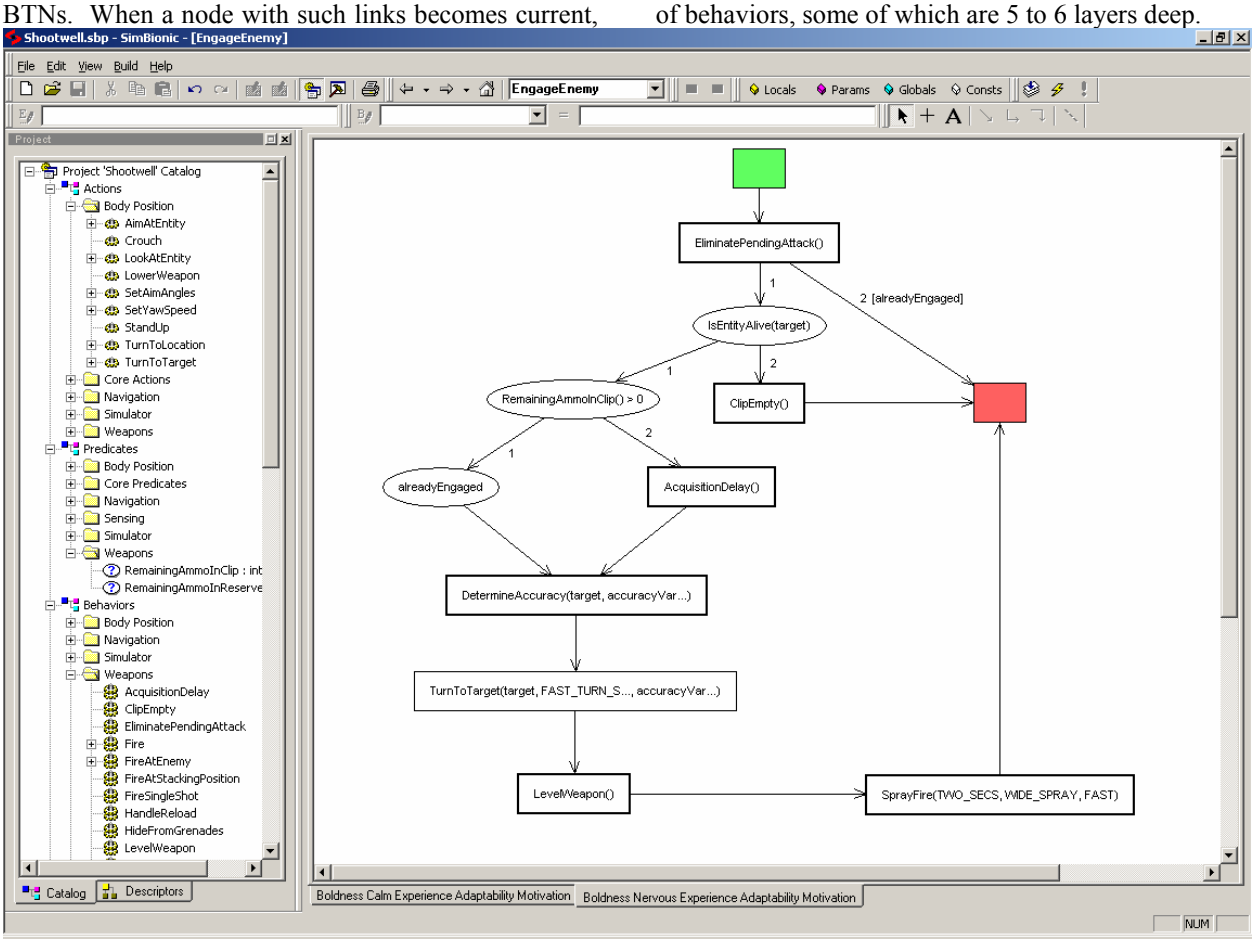

Figure 1. SimBionic Behavior Authoring Environment

A transition in a BTN is a directed arc connecting two nodes X and Y (feedback loop) and indicating a potential direction of control flow. A decision process, typically a set of logical conditions, but possibly something more sophisticated, is associated with each transition. A transition is said to be *active* if its decision process returns a "true" result. An active transition indicates that the BTN may change its current node from node X to node Y depending on

what other transitions from node X are active and their relative priorities.

USE OF POLYMORPHISM

The power of the hierarchical description is further increased by the use of polymorphism where the same behavior name can refer to different BTNs based on the value of an entity's descriptors. The user can create **descriptors** and **descriptor categories** that describe different facets or attributes of each entity's current state. For example, the prototype has a descriptor category called Calmness that contains descriptors called Nervous and Calm. These describe an aspect of the current emotional state of each simulated IC. Descriptors are also hierarchical so that lower level descriptors can be created to describe the entity in more specific terms than higher level descriptors. Using descriptor categories, the user can create **polymorphisms** which are variations on a behavior that are associated with particular combinations of descriptors. Then, during runtime, SimBionic selects and executes the polymorphic version of a behavior associated with the lowest level descriptor that matches each entity. For example, if the Calmness factor of a simulated soldier is Nervous, SimBionic will look for behaviors associated with Nervous soldiers. If there are no behaviors defined for Nervous soldiers, SimBionic will select and run the default behavior defined for all soldiers. Polymorphism allows one to extend existing behavior sets for new types of entities by specializing only the parts of the behavior "chains" that differ from one type of entity to another. In this way, the common parts of behaviors can be reused varying just the parts that change. This capability makes it easier to create rich sets of entities that behave differently depending on their individual characteristics.

POPULATION MODELING AND PROBABILISTIC DECISIONS AND ACTIONS

In the SimBionic environment, each enemy combatant has a set of attributes that dictate its gross behavior. Numerical ratings of Boldness, Calmness, Experience, Adaptability, Motivation, Hearing, and Accuracy are calculated based on a statistical model of a combatant's source population (e.g. US Marine, Iraqi insurgent, etc.). This model, separately defined for each population type, consists of a set of means and standard deviations for each of the listed parameters and is saved in a standard text file for ease of editing. During combatant instantiation, the values associated with each attribute are used to define a roughly normal distribution (also called Gaussian or bell-shaped) from which a rating is chosen at random. This gives the IC's natural variation while still being representative of different populations. Some of the attributes, such as the IC's nervousness or hearing (which may be damaged), may change during the course of a scenario while others are considered more permanent, such as Adaptability or Experience level. For repeatability in training scenarios, calculated ratings can be saved to a file and reloaded or edited at will. Instructors can also define the number and population type of enemy combatants to be instantiated in a given training scenario by editing another simple text file. Additionally, a specific IC's weapon related decisions and actions are probabilistic so that even a specific IC behaves differently from itself in the same and similar circumstances in reasonable and expected ways. Each of the behaviors is dependent in large part on the attributes of a particular combatant but is also mildly randomized to more accurately represent the dynamic nature of real-time decision-making. Similarly, firing behavior is dependent on attribute values but with a degree of randomness to avoid predictable results. Actual shot location is perturbed (relative to a perfect aim) slightly by the value of these attributes as well as by the speed of any required turn during aiming. If a combatant is surprised, for example, and spins around to face his target, his ultimate shot location will be affected by both his inherent quick aimed accuracy and by the speed of his turn--faster turns will lead to a greater degree of error in aiming. Should he need to reload, a low Calmness value could cause him to probabilistically drop his weapon.

MOUT IC BEHAVIOR MODEL EXAMPLES

The first example of a behavior was previously shown in Figure 1. This is the EngageEnemy behavior for a nervous OPFOR inside a room. (Note that the final node is to spray fire, fast, for two seconds in a wide arc, as is often the case for a nervous OPFOR.) The first step is to eliminate any future attack this OPFOR had planned. A planned attack might have been the result of hearing noises outside the room. This plan was preempted by seeing actual enemy entering the room in order for this behavior to be active. Typically, the first sub-behavior executed within this BTN is the *AcquisitionDelay()* (since the target is initially alive and the OPFOR generally has ammo) which is a one to two second time period depending on experience and mental state. Then, a determination as to the accuracy is made somewhat randomly, but also based on the OPFOR's accuracy, experience, and nervousness parameters. This nervous OPFOR then turns quickly

to its target, levels its weapon and sprays fire in the general direction of the target for two seconds. It then reassesses the condition of its target and reloads, if necessary, by calling the *ClipEmpty()* sub-behavior. Note that it will skip the *AcquisitionDelay()* when re-engaging the same target.

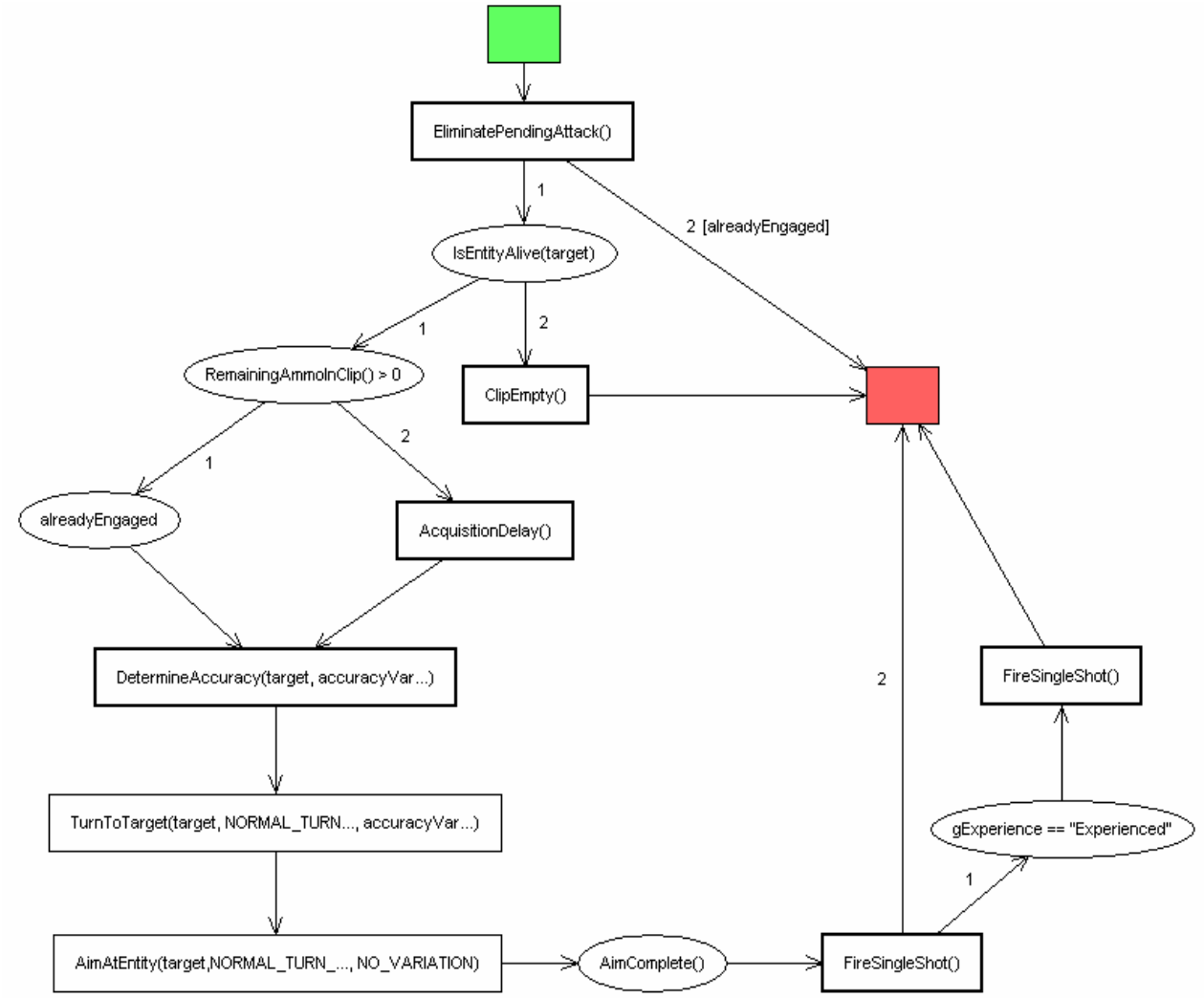

Figure 2. The Calm EngageEnemy Behavior

Figure 2 shows the *EngageEnemy()* behavior for a calm OPFOR which contrasts with the same behavior for a nervous OPFOR. This is an example of polymorphism. A higher level behavior calls an *EngageEnemy()* behavior whenever an enemy becomes visible to the OPFOR without reference to its calmness. In fact, calmness is set to an initial value based on the type of person the OPFOR is but changes dynamically during the course of the scenario as events occur. When the enemy becomes visible to this OPFOR, the SimBionic runtime dynamically selects the correct *EngageEnemy()* behavior based on the value of the OPFOR's Calmness. Note that this BTN makes use of some of the same predicates, actions and sub-behaviors as the BTN in Figure 1. This shows some of the power of hierarchically defining BTNs so that they can be reused. The first half of this BTN is the same as in Figure 1; but the calm OPFOR takes the extra time to aim at its target, complete the aiming process, then fire one or two shots depending on its level of experience before reassessing the target. This example also shows how OPFOR parameters (such as Experience) can be used to vary a behavior without using polymorphism.

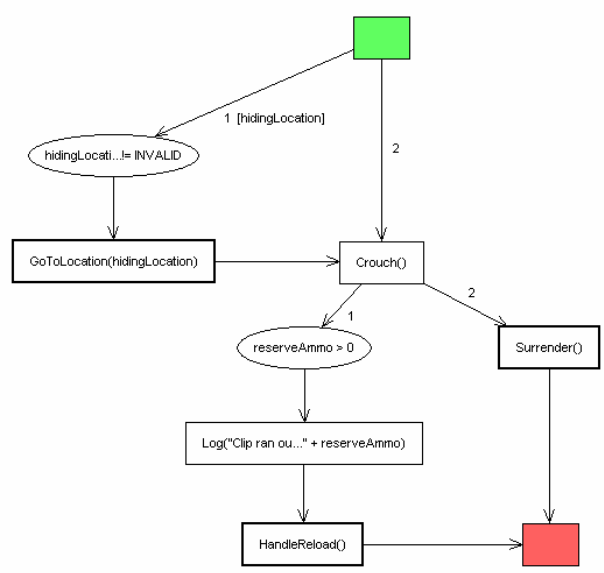

Figure 3. The ClipEmpty Behavior

The *ClipEmpty()* behavior (see Figure 3) is executed when the simulated combatant has run out of ammunition in the clip. This behavior variation, for a timid combatant, leads him to run to a cover position, if one exists, crouch, then reload. A braver combatant might reload on the spot instead.

The *Surrender()* behavior (see Figure 4) is called from multiple BTNs whenever the particular BTN determines it is appropriate to surrender. For example it is executed from *ClipEmpty()* BTN, (Fig 4), in the event that no ammunition remained in the combatant's reserve. The combatant drops his weapon, raises his arms and waits. This is a good example of how a low level behavior may be very simulation specific. In this case, CounterStrike requires two separate drop weapon calls. Several high level behaviors use the *Surrender()* BTN such that if the underlying simulation was changed, those high level behaviors could be re-used, as-is, in the new simulation and only the *Surrender()* behavior would need to be changed.

PROTOTYPE DESCRIPTION

Our prototype utilizes the commercially available CounterStrike FPS (First Person Shooter) as its simulation engine, which provides the fundamental physics model and graphical representation of the game world. Software hooks, to the AI middleware product, SimBionic, drive enemy combatants within

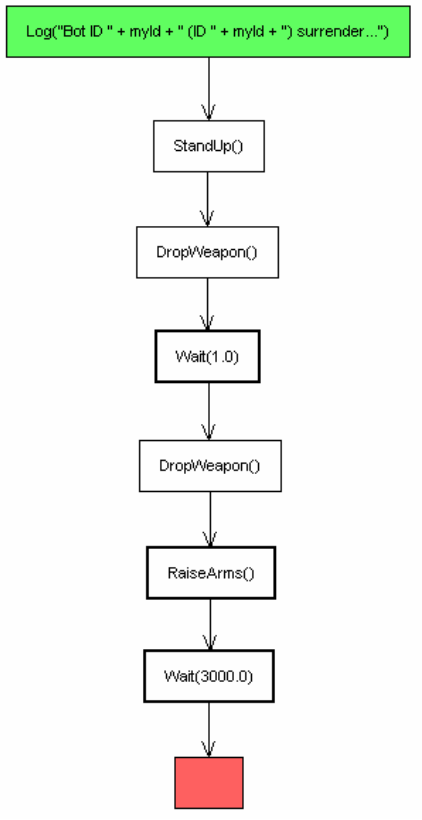

Figure 4. The Surrender Behavior

CounterStrike. SimBionic is a high level development environment that abstracts the fundamental predicates and actions of simulated entities from their implementations in the simulator. Transition to different simulation environments therefore involves only the reimplementation of these fundamental predicates and actions; the behaviors themselves remain intact and will function without alteration (dependent upon differences in simulator features).

In the simulated environment, we have equipped the enemy combatants with accurate sensing apparatuses through which to perceive their surroundings; these sensory abilities are made instantly and are directly accessible to the SimBionic environment. Combatant field of view, parametrically variable, we defined to be 90 degrees to the left or right of the current eye direction. They are able to hear and distinguish among the sounds of enemy gunfire, voices, grenade and breaching explosions, doors opening, and footsteps on hard surfaces. The location of loud sounds can be accurately determined if in an adjacent room, but softer sounds and those occurring at a greater distance provide only general location information. In the event of injury, the direction of attack is known. Deafness can be induced by close proximity to a detonating

grenade. Grenades, depending on type and distance, can stun them. In general, the simulated combatants' sensory abilities are a good approximation to those of real-life humans'- though they are defined separately and polled sequentially, the net behavioral result is a combatant with simultaneous access to all of his senses.

Behaviorally, simulated combatants are capable of both immediate reactions to stimuli and longer-term planning, both of which are effected by the values of their attributes. Upon hearing the sound of footsteps on tile floor in an adjacent room, an experienced combatant (Experience attribute above a certain threshold) will turn towards the sound and might spray fire through the wall; an inexperienced combatant wouldn't realize they could fire through walls, and would instead focus their attention on the door to their room, preparing for an attack. If an experienced combatant heard the voice of an enemy outside the door of the room they were hiding in, instead of immediately firing through the door, they concentrate on it, ready to fire, but after a few seconds try spraying fire at the wall adjacent to the door in an attempt to hit Marines in a stacking position. When suddenly confronted by an enemy, a nervous (Calmness attribute below a certain threshold) combatant wildly sprays fire in the general direction of the enemy, while a calmer combatant would take the few extra tenths of a second to aim properly and fire a more accurate burst. Upon seeing a grenade tossed into a room with minimal bounce (a stand-in for a softly tossed grenade, as CounterStrike does not allow players to change throw velocities), a timid combatant (Boldness below a certain threshold) runs to a cover position to avoid the blast, while a bold combatant would run towards it in an attempt to throw it back. If an experienced combatant hears two grenade detonations, one following the other after some brief interval, he is very likely to preemptively retreat to a cover position in anticipation of further grenade attacks. If a nervous combatant hears many different engagements (defined as shots fired, grenade detonations, etc.) over a period of time, he becomes afraid, drops his weapon, and will immediately surrender to a visible enemy. Each of these behaviors is dependent in large part on the attributes of a particular combatant, but is also mildly randomized to more accurately represent the dynamic nature of real-time decision-making.

Each combatant has three accuracy attributes. The first is moving aimed accuracy, which comes into play when the target is positioned such that the combatant must turn more than fifteen degrees to face it. The second attribute is still aimed accuracy, used when the target requires less than a fifteen degree change in body angle. The final attribute is quick aimed accuracy, used when the combatant is surprised and needs to make a hasty aim.

If he runs out of ammunition, he will immediately surrender. There is also an inherent acquisition delay for all combatants when they first see an enemy, the length of which is dependent upon their experience and calmness.

Prototype Demonstration Sequence

As the demo begins, the human player (hereafter referred to as "the player") makes his way quickly across the tiled lobby of the office. An IC, waiting with attention focused on his room's only doorway, hears the player's footsteps as he approaches; having a relatively high level of experience, he knows the penetration capabilities of his AK-47, and decides to preempt an attack by spraying fire through the wall at the rough location of the sound. Once the smoke has cleared, he waits a few seconds for any audible sounds of life on the other side of the wall before refocusing his attention on the room's door. As the player (walking slowly now to avoid giving his position away a second time) bursts through the door, the IC, weapon already trained on that location, is able to place quick and accurate fire on his opponent, pausing between shots to reassess whether the player has yet been killed.

The player, still on the tiled part of the lobby floor, approaches the second room more slowly. Once he has positioned himself in an appropriate stacking location near the door, he radios to his team that he is in position to clear the room. The IC waiting inside hears his voice through the door, and prepares for an entering attacker by training his weapon on it. After a few seconds pass uneventfully, however, the well-experienced IC hypothesizes that his attackers are probably stacking outside the room; as such, he turns his attention away from the door and sprays fire through the wall at where he assumes the most likely stacking location would be. The player, having avoided this onslaught, tosses a flash bang grenade into the room. Noticing the relatively predictable angle at which the grenade approaches the wall (a stand-in for a softly tossed grenade), and being particularly bold, the IC runs at the grenade in hopes of catching it and throwing it back. He is too slow, however, and is incapacitated (though not mortally wounded) when it detonates.

As the player enters the room, he engages the IC inside. The IC, though unable to see the player

through the overturned table, senses the direction of fire he is taking when he is injured, and sprays fire back through the table at the player. The player bounces a frag grenade at the IC, who, based on the difficult angle of bounce off the second wall, tries to run for cover before it detonates. Though not killed by the blast, he is severely injured and stunned.

The fourth room the player enters contains two ICs: the one directly opposite the door has extremely poor accuracy, and is largely unable to hit the player; the one in the corner is quite nervous, and elects to spray fire in the general direction of the visible player despite having a clear line of sight.

The fifth room is convex in design, and contains an IC hiding out of line of sight of the door. As the player enters, the IC hears the squeak of the door hinges, and tries to creep up behind the player as he pies off the room.

Once he has eliminated the IC, the player loads his magazine with a breach round and blows through the wall into the office stairwell. Hearing this loud explosion, and understanding it to be a breach, the IC in the adjacent convex room quickly moves away from his hiding spot in fear of being hit by a similar breach into his room.

BEHAVIOR MODELING METHODOLOGY

Over the course of the project we utilized a process for modeling the weapon related behaviors which appears appropriate for modeling most types of OPFOR behaviors to support simulation training. There are 9 steps:

1. Training Requirements Investigation
   - What kinds of things should the IC do from a training perspective?
   - How do the learning/training objectives impact the IC's behaviors?
   - What types of situations, types of actions, and stimuli/reactions are required of the IC?

2. Knowledge Engineering
   - What kinds of OPFOR are there?
   - What are their behaviors and decision-processes?
   - What are the variations in OPFOR behavior and why?
     - Are there variations between different types of OPFORs? Are there variations within a population (type of OPFOR)?
   - Create Scenarios. Define range of scenarios. Define associated behaviors.

3. Design Behaviors
   - Draw BTNs, with different states reflecting either different mental states (e.g. ready, surprised, alert, concentrating, etc.) or actions.
   - Actions may be probabilistic (e.g. shooting accuracy) and links may be probabilistic (e.g. the decision to grab the grenade or seek cover).
   - BTNs are polymorphic and parameterized.

4. Design parameters for the behaviors based on what's needed by the behaviors and what is different between different ICs or the same IC at different times.

5. Determine population definitions in terms of parameter ranges (means and standard deviations).

6. Instantiate ICs from relevant populations for each scenario.

7. Assign appropriate locations and behaviors to the ICs (i.e. give them orders).

8. Test, Evaluate, and Validate
   - Collect accuracy and other statistics and compare to the population definitions.
   - Observe OPFOR actions in different circumstances for qualitative evaluation of realism.

9. Improve behaviors and population definitions based on tests and feedback and iterate.

LESSONS LEARNED

There were several lessons learned from this effort. The first is that BTNs worked well for modeling realistic OPFOR weapon behaviors. The use of a graphical tool for editing the BTNs was essential, since it allowed new behaviors to be created in minutes and existing ones to be modified in seconds. Parameterizing the behaviors also facilitated the behavior improvement process, since an observation that some action occurred too slowly or quickly, are not large enough and could be corrected by changing a parameter. The use of probabilistic methods was important to achieve realistic levels of variation in the OPFOR behavior. Having a system capable of hierarchical representations was important to keep the complexity at a manageable level and the understandability high. Even seemingly simple

behaviors are very hierarchical - often 5 to 6 levels deep. Because of this depth, use of different levels of abstraction was important. Polymorphism was also an important tool especially to create variants of the behavior for the different types of OPFOR. It was very useful to initially create one version of the behavior, then create alternative specializations of the behavior for different types of OPFOR. Each of the above contributed to being able to quickly create a working version of the behaviors then vary incrementally increasing their realism and complexity.

FUTURE WORK

There are four main thrusts to our future efforts directed toward realistic automated OPFOR weapon behaviors. The main one is to refine and further increase the realism, complexity, available actions, breadth and depth of, and implement behavioral models for weapon firing tactics, techniques and procedures for a variety of MOUT situations. OPFOR weapon firing behaviors will be developed with close adherence to military doctrine, enemy intelligence, and thorough elicitation from and evaluation by subject matter experts. A hierarchical breakdown of the subtasks associated with target acquisition and firing execution in a variety of MOUT contexts will be refined, detailed, and implemented, so that high level tasks and decisions can be expressed in terms of abstractions of lower level tasks and decisions. This provides a basis for implementation, and also allows for a modular development process, where lower level behavior elements may be elaborated or simplified in accordance with iterative development conclusions regarding their effective level of realism.

The second thrust is to integrate the improved OPFOR behaviors with VIRTE. This integration is currently planned through integration with OOS, the OneSAF Objective System, to which VIRTE will transition. This would also make the realistic OPFOR behaviors developed accessible to other simulations using OOS.

A third thrust is the development of tools to support the Behavior Modeling Methodology. The modeling process includes requirements gathering, knowledge elicitation, behavior design, parameter identification, population construction, simulated IC instantiation, behavior assignment, testing, validation, and evaluation. Tools will be developed to support each of these steps. The validation step is especially important on the implemented weapon firing behaviors to determine that there is an adequate level of realism. Validation metrics which include both statistical comparison and observation should be used to test simulated entities in an array of explicitly defined conditions for which a specification was developed outlining the acceptable OPFOR actions given each condition or set of conditions.

The fourth thrust is to expand the breadth of contexts in which the automated OPFORs can function. The current implementation addressed the highest priority OPFOR behavior - realistic weapon firing and related behaviors within rooms being cleared. The next priority would fill out the defensive situations within a building--hallways, hall way intersections, and stairwells, including both defending and ambushing behaviors. The next highest priority context is defending against an enemy (US soldiers) attacking from the outside. Additional behaviors would include defending a building, proper position/cover selection, defending and ambushing at road intersections, firing around corners and from behind walls, and scanning behavior. The third priority context is the OPFOR engaged in offensive MOUT operations, primarily from outside the buildings that the trainees are defending. Behaviors include tactical movement down roads, crossing intersections, entering defended buildings, reconnaissance and surveillance, and the use of fire to destroy or disrupt the defense. The last priority context for the OPFOR are offensive operations within buildings, since training U.S. soldiers to defend a portion of a building from attack from within it, is a low priority.

RELATED WORK

There is a considerable history of work in the development of behavior sets for complementary focus areas other than weapon firing. (Wray et al, 2002) describe an approach and architecture for defining simulated ICs that can navigate and engage in combat in MOUT environments, using the Unreal Tournament game engine as a simulation platform. This approach is based on the Soar cognitive architecture, which not only models IC decisions and actions, but also cognitive states and artifacts such as goals and memory. Here, the emphasis on realism is achieved with attempts to actually model human reactions and deliberative abilities given the parameters of a simulated environment. (Reece et al, 2000) present a set of path planning algorithms for MOUT environments, which consider factors like threat avoidance and opportunities for concealment. (McDonald et al, 2001) developed CGF behaviors that also incorporate issues associated with different rules of engagement into the tactical decision-making rules.

(Henninger and Taylor, 2002) and (Middleton et al, 1997) prepared studies which focus on weapon firing and suppression tactics and techniques.

The notion of using a visual environment to specify behaviors for simulated autonomous entities has a number of precedents in academia and in industry. (MacKenzie et al, 1997) describe the MISSIONLAB system that allows an author to specify the behavior of multiple robots. The author does this visually using similar hierarchical state and transition links. KHOROS (www.khoral.com) is a popular visual editor for image processing that allows users to string together operators into a flow diagram. Each operator comes from a standard library, or is defined by the user using standard C code. In industry, there are few visual editors for games. Perhaps most notable is the Motivate package from the Motion Factory (www.motion-factory.com). Its use is limited to development companies and is not freely available for research use. Also, visualization toolkits have been developed for the interpretation of behaviors defined with the Soar architecture described above.

REFERENCES

Henninger, A., and Taylor, G. (2002). Development of Individual Weapons Firing Algorithm for Air Force Security Forces Distributed Mission Training. Unpublished manuscript.

Lyons, D. M., Schmorrow, D., Cohn, J. C., & Lackey, S. J., (2002). Scenario Based Training with Virtual Technologies and Environments. *Proceedings of the Image 2002 Conference*, Scottsdale, AZ.

MacKenzie, D., Arkin, R.C., and Cameron, J., (1997). Multiagent Mission Specification and Execution. *Autonomous Robots*, 4(1):29-57.

McDonald, B., Weeks, H., and Hughes, J. (2001). Development of Computer Generated Forces for Air Force Security Forces Distributed Mission Training. In *Proceedings of I/ITSEC 2001*. Orlando, FL.

MCWP 3-35.3, Military Operations on Urbanized Terrain (MOUT), US Marine Corps.

Middleton, V., D'Enrico, J., and Christenson, W. (1997). Simulation of Suppression for the Dismounted Combatant. In *Proceedings of 5th Conference in Computer Generated Forces and Behavior Representation*. Orlando, FL, March 1997.

Miles, J.J., (2003). Maximizing MOUT Training. *Marine Corps Gazette*. Retrieved May 26, 2004, from http://www.mca-marines.org/Gazette/2003/03Miles.html

Reece, D.A., Kraus, M.K., et al (2000). Tactical Movement Planning for Individual Combatants, Orlando, FL. Simulation Interoperability Standards Organization.

Reece, D.A., Kraus, (2003). VIRTE Demo II Final CGF Systems Analysis", SAIC, Orlando, FL.

Schwierz, K.P., Krenz, F., & Lipke, L., (2003). The IRIS Model. *MS&T: The International Defence Training Journal, 5,* 13-16.

Slear, T., (2004). Bringing Out the Worst in Armies. *MS&T: The International Defence Training Journal, 1,* 8-13.

Wray, R.E., and Laird, J.E. (2002). Intelligent Opponents for Virtual Reality Trainers. In *Proceedings of I/ITSEC 2002*. Orlando, FL.