Force Behavior Agents

Michael Denny, Bill DeSmedt, Kamal Gella, Mary Hiles, Don May, Sridhar Natarajan, Laurie Waisel, John Wass
Concurrent Technologies Corporation
Johnstown, PA
dennym@ctc.com, desmedtw@ctc, gella@ctc.com, hilesm@ctc.com, mayd@ctc.com, nataraja@ctc.com, waisell@ctc.com, wassj@ctc.com

ABSTRACT

We report on the development of an agent capability for operational decision-making within a real-time simulation world. A multiple agent system was developed to extend the native behaviors of entities (force units, vehicles, etc.) in tactical simulations. The system endows these entities with intelligent behavior capabilities allowing them to adapt to unexpected scenario situations. The agent system is designed to integrate tightly with the Semi-Automated Forces (SAF) simulators used in live-virtual-constructive simulation environments by DOD and others. Large-scale simulations often entail the necessity of human operators to direct or fill in the ongoing behavior of force units or other entities not being played by trainees or others in the scenario. Force Behavior Agents (FBA) eliminates this staffing requirement, achieves realistic conflict scenarios and, at the same time, simplifies the specification of complex mission scenarios rich in force interaction and variability. In contrast to federate level interaction in High Level Architecture (HLA) communication, FBA is designed to integrate directly with simulators at the fine-grained level of native task frame stacks and simulation state databases. Agent interaction with the simulator's state machines affords the means to adjust unit behavior, including disaggregation, transparently without disrupting normal simulator operations. Selection of alternative behavior tasks during runtime is governed by agents using situation look-ahead trials based strictly on the force unit's qualified sensor capabilities. These look-ahead trials are like sketchy simulations run by the individual agents to find the best alternative courses of action, much as a human commander will survey and compare the tactical options available to his unit. Ontologies define the tactical relations and doctrinal constraints on tasks, and a commercial agent platform provides the decision making environment. An early form of the FBA decision maker and its interface with Joint Semi-Automated Forces (JSAF) simulators was demonstrated to the Joint Forces Command (JFCOM).

ABOUT THE AUTHORS

Michael Denny is Principal Ontologist at Concurrent Technologies Corporation (*CTC*) where he investigates and designs decision support systems for multiple-source intelligence analysis and battlefield operations using simulation, agent, and semantic technologies. Dr. Denny earned a Ph.D. in Cognitive Psychology from Michigan State University.

Bill DeSmedt is Principal Knowledge Engineer at *CTC* where he designs knowledge representation systems for DARPA-funded projects concerned with how field commanders build mental models of evolving battlefield situations. Mr. DeSmedt has a background in computer science, artificial intelligence, and Russian/Soviet studies. He has led applied research on intelligent conversational agents, and is the author of a novel *Singularity*.

Kamal Gella is a Principal Director at *CTC*, where he leads a group of software/system engineers in support of decision support, enterprise architecture and case management systems. Mr. Gella has an M.S. in Computer Science from the University of South Carolina.

Mary Hiles is Principal Technical Manager at *CTC* where she leads multi-faceted teams to develop decision support and simulation systems. Ms. Hiles has a Mechanical degree from Lehigh University and an MBA from North Carolina State University.

Don May is Senior Systems Engineer at *CTC*, where he architects and develops decision support and case management systems. Mr. May has a B.S. in Computer Science from the University of Pittsburgh, Johnstown.

Sridhar Natarajan is Principal Technical Manager at *CTC*, where he provides technical leadership toward the development of decision support, enterprise architecture and case management systems. Dr. Sridhar has a Ph.D. in Engineering from the Ohio State University.

Laurie Waisel is Director of Decision Support Technologies for *CTC* where she has also held the titles of Principal Technical Manager and Principal Visualization Engineer. Dr. Waisel is Program Manager of the Representing Enriched Context program, which develops decision support technology to assist humans in using stories to reason about complex situations. She earned a Ph.D. in Decision Sciences from Rensselaer Polytechnic Institute.

John Wass is Software Engineer at *CTC* where he does application and agent system programming for real-time simulation, decision making, and data acquisition systems. Mr. Wass earned a B.S in computer science from DeVry University.

Force Behavior Agents

Michael Denny, Bill DeSmedt, Kamal Gella, Mary Hiles, Don May, Sridhar Natarajan, Laurie Waisel, John Wass
Concurrent Technologies Corporation
Johnstown, PA
dennym@ctc.com, desmedtw@ctc, gella@ctc.com, hilesm@ctc.com, mayd@ctc.com, nataraja@ctc.com, waisell@ctc.com, wassj@ctc.com

INTRODUCTION

The community of Semi-Automated Forces (SAF) users in live-virtual-constructive simulation environments has expressed a keen interest in gaining more autonomy in the behavior of simulation force entities, especially those of opposing forces. Such behavior autonomy can relieve users of the continuing need to fashion each simulation scenario in detail and to fill in the missing actions and correct inappropriate actions during simulation runs that result from the lack of adaptive behavior in computer generated forces (CGF). Behavior autonomy grounded in appropriate force doctrine can also lead to richer and more realistic scenario simulations. The community is exploring various architectural avenues to achieve more autonomy in native entity behavior including intelligent and cognitive agent-based solutions. The problem of inserting additional intelligence into native behavior capabilities within the SAF simulation environment, however, is challenging.

Previous efforts to achieve more autonomous entity behavior have adopted differing technical approaches. In one, agent platforms are embedded into the simulator environment and then borrow existing entity communication capabilities, such as passing radio messages, to direct the entity's behavior in the simulation (Koss et al 2000). Another adds separate agent-based federate platforms, using High Level Architecture (HLA) communication for example, to assume the role of the entity (Evertsz et al 2007). Another replaces native entity behaviors before a simulation run with newly compiled behaviors tailored for specific upcoming scenarios (Garcia and Griffith 2005). With these approaches, effective communication with the simulator has proven to be a bottleneck. Federates, for example are susceptible to high latency and possibly unreliable simulation data communication. They also tend to put data exchanges with the agent at a generic level far removed from the actual processes governing the simulator's runtime selection and execution of an entity's tasks. This isolation means that individual native behaviors and the immediate selection of these behaviors cannot be modified (without human intervention) once the simulation begins, unless software agents are allowed to act as federates populating the simulation federation. These agent-based federates either become the subject entity and completely take over control, or they observe the scenario externally and issue federation level commands giving the entity new tasking as native behaviors play out in the simulation scenario. In the former case, the validity of native entity behaviors built into the SAFs is lost, and in the latter case the agility of composing tasks into complex alternative behaviors is lost. Generally, these solutions: 1) do not offer truly flexible and realistic behavior in the face of varying scenario situations; 2) do not scale readily to large quantities of entities so enhanced; and 3) do not ensure that the entities continue to adhere to their sanctioned native behavior constraints including entity-appropriate situation sensing in the simulation.

To address these shortcomings, we have developed an agent–simulator communication scheme that operates by seamlessly extending the simulator's own behavior control mechanisms. Doing so ensures that entities continue to behave in legal ways, while gaining the flexibility of selecting behaviors according to actual scenario dynamics and an entity's (plausible) assessment of the current situation. The approach provides a low level agent interface into SAFs allowing entity tasks to be recomposed dynamically during a local or distributed simulation. Being internal to the simulator, the implementation does not depend on HLA or Distributed Interactive Simulation (DIS) communication protocols or bandwidth.

Our solution, Force Behavior Agents (FBA), is a simulator utility that assumes the role usually played by human operators to formulate mission-specific behaviors in a scenario and guide the actions of constructive entities as the scenario unfolds. The multiple agent system affords simulation entities (force units, vehicles, etc.) the intelligent behavior capabilities

that enable them to adapt in real-time to unexpected tactical situations. FBA helps users create complex simulation scenarios where each force unit executes realistic behavior under the command of their own FBA agents. FBA combines a general purpose agent tactical decision maker with its simulation runtime interface. The decision maker and SAF interface have been demonstrated as a Joint Semi-Automated Forces (JSAF) capability for JFCOM using a focused tactical insurgency scenario.

In varying degree of contrast to other agent approaches, FBA: 1) exploits the efficient internal communications of JSAF; 2) monitors entity states internally, rather than externally as a federate; 3) works cooperatively with JSAF behaviors, rather than taking complete control of the entity; 4) provides continuing adjustment to behaviors and behavior substitutions, rather than simply branching to alternative behaviors; and 5) works dynamically during a scenario without having to pre-compile any new behaviors, behavior sequences, or behavior branching.

After introducing the high-level FBA architecture, we describe the agent–simulator communication interface implemented for JSAF that allows FBA agents to monitor a simulation and to dynamically alter entities' behaviors. We then describe the look-ahead reasoning, analogous to a human commander's problem solving, that agents use to select their entity's next behavior in a situation. Finally, we introduce the concept of agent-mediated communication in a synthetic battlespace as a future enhancement to FBA.

FBA ARCHITECTURE

FBA was conceived as a utility that could extend native entity behaviors in a JSAF simulator in as direct and transparent a manner as possible. The design objective was a minimal impact on simulator internals and simulator performance, while maximizing the behavior flexibility achieved. At the heart of the utility is a runtime messaging scheme between the agent platform and JSAF that enables the collection of simulation state information on any force units engaged in the simulation scenario and the runtime control of any unit's tasks declared as "adaptive" during scenario authoring. The scheme's two-way communication also allows products of agents' situation deliberations, such as determining regions of unit vulnerability, to be projected back to the simulation view in a JSAF plan view display (PVD) if desired. This feature along with graphical depictions of agent activity during look-ahead simulation trials, lend useful behavior explanation capabilities to FBA.

The FBA architecture is predicated on a simple use case. When a user wants a native entity behavior to adapt dynamically to the unfolding planned simulation scenario and display a more autonomous and flexible behavior response, they simply select an "adaptive behavior" option that extends the standard JSAF editors. The user is theoretically free in this way to turn any entity's behavior into an adaptive behavior. As each entity behavior requires some semantic specification to enable proper monitoring and reasoning about it, the present range of JSAF behaviors so supported by FBA is limited to the few used during development testing and FBA demonstration. Preparation of other native behaviors is straightforward.

This FBA capability affords agents the means of treating the current simulation scenario analogously to having a planned course of action (COA) for its corresponding force unit or vehicle and forming a current operating picture (COP) of the SAF simulation. Simple approximations of both are represented in the FBA agent's own internal simulations used to reason about the action and situational semantics of an ongoing SAF simulation scenario. An FBA agent acts to complete an adaptive behavior as if striving to satisfy the mission goal of completing a COA by adapting the COA if it begins to fail the force unit. The agent may at any time elect to alter the unit's behavior, for example in reaction to obstacles reducing the likelihood of it completing a task that advances it toward the mission goal. The agent will accordingly find and choose alternative behaviors that adapt its scenario tasks, as that part of a COA, in order to exploit the current operational situation manifest in the SAF simulation. The selection of alternative behaviors unfolds in a look-ahead test-modify-test cycle of reasoning. During the reasoning cycle agents are able to consult simulation domain and tactical domain ontologies that formalize semantics of simulation events and behavior tasks. The simulation ontology facilitates the marshalling of current simulation state information, and the tactical ontology facilitates the selection of alternative tasks used to adjust current behavior.

The overall system is made up of four software modules that provide communication between and integration with a JSAF simulation and agents running in their own agent platform environment. Each module communicates through a combination of function calls and a messaging protocol that was designed for connecting JSAF to the agent environment.

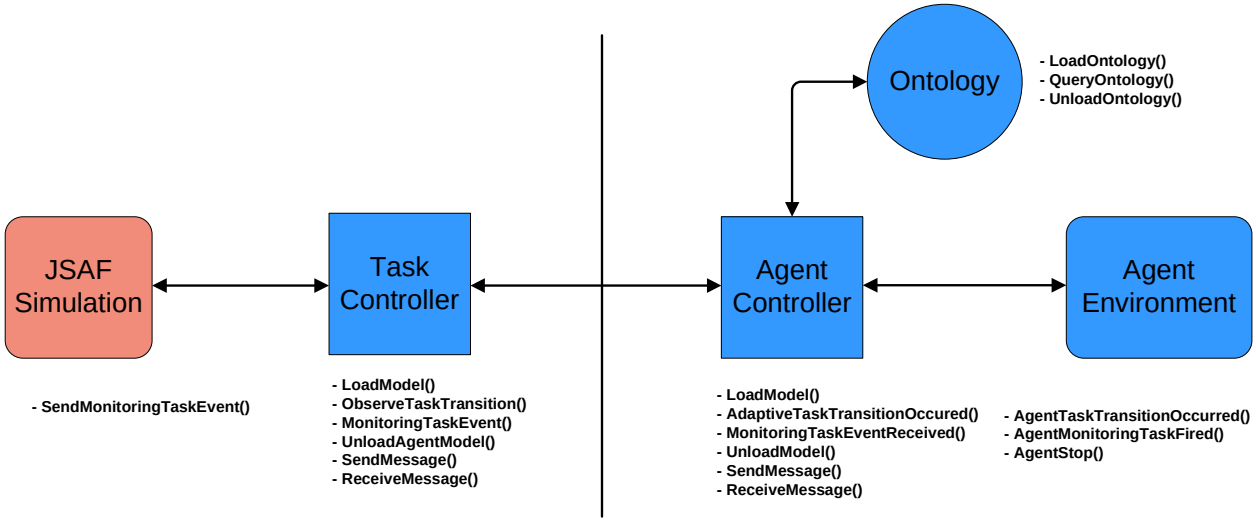

Figure 1. Overview of FBA Functional Modules

Interfacing with JSAF is handled completely by the Task Controller as shown in the Figure 1 diagram, and interaction with the agent environment is handled completely by the Agent Controller. The Task Controller (to the left of the agent platform boundary seen as the vertical line above) includes a thin layer that extends JSAF software libraries. The modules communicate using a messaging protocol that is currently being passed using the Java Native Interface (JNI) but the communications interfaces are designed in a way that it would be possible to replace JNI with another solution such as TCP/IP if desired. The Agent Controller is also connected to a Web Ontology Language (OWL) ontology through the integration of an OWL reasoner and parser. The primary methods supported by FBA's inter-module linkage are listed in the diagram.

AGENT-SIMULATION INTERFACE

A SAF simulator interface was tailored for JSAF integration to enable the exchange of real-time simulation information between JSAF and the FBA agent platform in both directions during scenario runtime. The integration scheme works by monitoring the progress of a JSAF unit's *task frames* on the JSAF *execution matrix,* or *task frame stack*, and manipulating *task frame* content in a dynamic manner to respond to changing simulation circumstances.[1] Using this dynamic task frame technique, the behavior agent can interface with the running JSAF simulation in a manner that does not impede its execution, yet is able to sense simulation conditions and inject new behavior tasks during runtime. The communication used to integrate the agent platform with JSAF was designed to minimize the simulation data traffic necessary for each entity agent during a simulation scenario.

Each entity (unit, vehicle, etc.) involved in a JSAF simulation has its own *execution matrix* specifying its behavior during a scenario. If the *execution matrix* contains an adaptive task enabled by FBA, its task sequence as planned in the scenario is subject to change where tasks may be added and deleted. Thus, as seen in the next graphic, Figure 2, the task sequence is transformed to deal more effectively with the actual conditions of the current simulation – one sequence as a conventional scenario of planned tasks, and one sequence resulting from a *move* task in the scenario having the adaptive option. This transformation is realized as a change in *task frames* residing on the JSAF unit's *execution matrix*. To achieve the elaboration seen in the adapted matrix, the agent must be able to monitor the currently executing task, detect the current state of the current task, react to any problems in the current task, and counter them with other promising behavior tasks.

[1] Terms that refer to specific JSAF functional constructs are italicized in our descriptions.

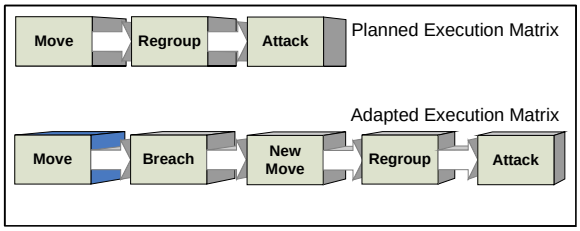

Figure 2. Adaptive Behavior Transforms Output of Execution Matrix

Our focus on extending native JSAF behaviors to exhibit adaptive capabilities had us look closely at the simulator's existing mechanisms for scheduling and executing unit behavior tasks. It became our contention that you could use this raw machinery to establish a proxy-like interchange between agents and JSAF. A capability was developed by inventing new forms of JSAF tasks to serve as intermediate carriers and consumers of information within normal JSAF processing. These new behavioral functions are created and compiled into JSAF as part of the FBA agent integration. They run in the background during a simulation to monitor the progress of a unit's current task. Progress equates to aspects of task performance related to the unit's continued ability to complete the current task successfully. There are two types of special intermediate tasks created as new JSAF behaviors – the Instruction Task and the Monitoring Task which can be invoked along with a native behavior task. Most of the JSAF modifications required by FBA are in the form of these new simulation behaviors which run unobtrusively as JSAF *background tasks* that enable the unit to monitor task status and to pick up new task assignments (as Instruction Tasks). The rest of the JSAF modifications are the machinery to programmatically insert new *task frames* into the current JSAF *execution matrix* during runtime independent of any JSAF user interface. See Figure 3.

During a unit's adaptive behavior tasks, Monitoring Tasks monitor significant capabilities of the unit like the unit's current health, strength, mobility, or task gain (e.g., advance, kills, reserve level). Conditions are set on these *background tasks* to execute when a given threshold is passed (mobility falls, attrition rises, etc.). Each Monitoring Task is surveyed by the agent and when one or more eventually execute, the agent interprets which Monitoring Tasks fired and the simulation state information they deliver in order to produce a new COA for the unit performing an adaptive task. The new COA is passed as instructions to JSAF for revising the unit's *execution matrix*. The basic process is illustrated at a high level in Figure 4.

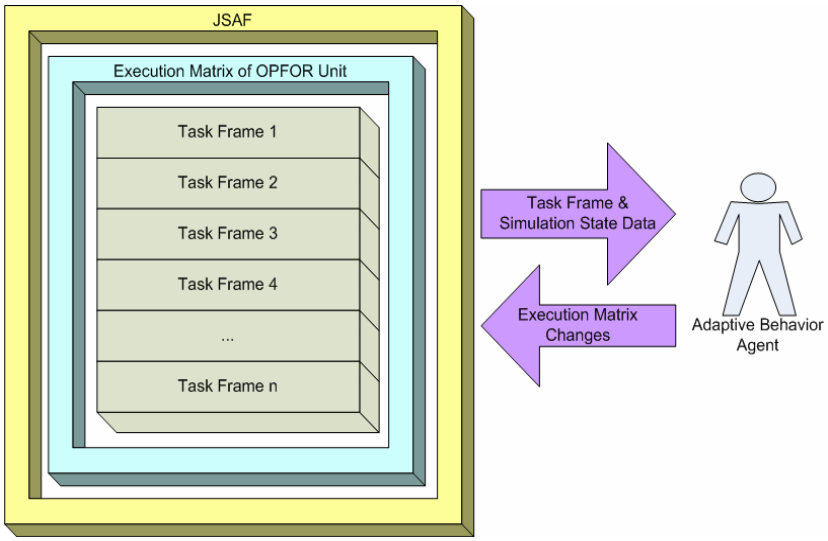

Figure 3. FBA Agent Manipulates a Force Unit's Execution Matrix during Runtime

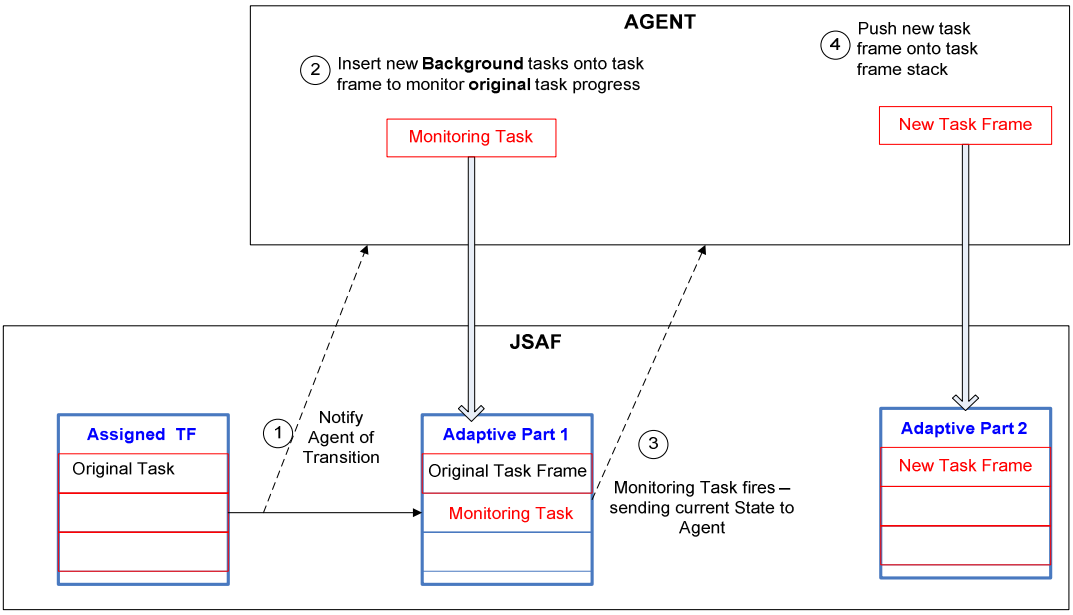

Figure 4. Agent Integration Solution uses JSAF Task Structures

In this sequence, the steps comprising the adaptive behavior execution are:

1. The original (*assigned) task frame* begins to execute. The agent is notified of the transition from the previously executing *task frame* to this *task frame* that includes an adaptive behavior.
2. The agent inserts a new Monitoring Task into the *task frame* as a *background task* to monitor the status of this adaptive task. There could be multiple Monitoring Tasks inserted to check for different capabilities (e.g., CheckMobility, CheckHealth, etc.).
3. The Monitoring Task fires, notifying the agent that the currently executing task is ineffective or puts the unit in jeopardy, and sending current simulation state information to the agent.
4. The agent pushes new *task frames* onto the unit's *task frame stack* in order to replace the currently executing *task frame*. Note: this new task could also be an adaptive behavior, in which case the sequence of events would start over again (recursively).

AGENT-LED BEHAVIOR

Whenever the author of a simulation scenario decides to make one or more of an entity's behaviors "adaptive" in FBA extended JSAF, it is in fact assigning an agent to govern that entity's behavior during simulations and invoke new tasks if and when the prescribed behavior task begins to fail. Thus, an individual agent is created for each force unit or vehicle entity given adaptive behaviors in a simulation scenario. This agent strives to achieve a successful completion of the scenario by treating tasks planned initially for its unit in the scenario as an adaptable course of action, possibly with specified mission goals. If and when an entity's agent detects task failure or a lack of progress, the agent begins to look for a more effective task alternative.

Decision making in FBA is a form of abductive action selection used to dynamically compose unit behavior. Action selection means selecting an alternative behavior task from JSAF's repertoire for that entity, one that exhibits better success or progress in a situation look-ahead trial than the currently underachieving task. Such selection may simply involve skipping the task and moving on to the next one in sequence, or it may entail replacing or inserting several behavior tasks. Selection is abductive because the agent accepts the best performing task relative to other candidate tasks it finds and tests. FBA action selection is not accomplished as cognitive agents might by undertaking the complexities of modeling sensation, perception, attention, memory and other functions of cognition as discrete interacting processes. Instead, the

agent zeros in on simple holistic processes of command decision making, and on communicating its conclusions to other interacting agents.

The selection of alternative behaviors unfolds in a look-ahead test-modify-test cycle of reasoning. Encountering an obstacle, the agent will attempt to find a different way to achieve the stalled operational task by choosing from its behavior repertoire to, for example, either breach or bypass. It initially consults an ontology of behavior tasks to find all candidate replacement tasks that may have similar effects on the object of the agent's course of action. The ontology represents the tactical operation semantics of each task in the agent's behavior repertoire (functionally relating them to more abstract operational tasks). Repertoires are specific to an agent's operational role. The agent is guided by the nature of its preceding look-ahead trial failures in finding functionally equivalent replacement tasks.

Situation look-ahead trials in FBA enlist a standing collection of agents to represent each entity in the immediate area of a unit's operations to serve in an approximate or sketchy simulation with a short horizon. JSAF is not involved in any active way in these agent-based simulation trials. They are hosted strictly on the agent platform, as diagrammed in Figure 5 below, using agents as surrogates for the simulator's simulation entities or force units.

Agent Architecture

The FBA design partitions its prototype agent functionality into four modules that connect in a mostly linear manner. The Agent Controller talks to JSAF through proxies and directly to the Scenario Manager Agent which in turn relies on the Decision Maker Agent to invoke Monitoring Tasks and decide on alternative behavior tasks by consulting the Situation Look Ahead Complex. This module tests candidate alternative tasks by running each through its own look-ahead trial that begins with the currently known JSAF state.

In Figure 5 below, the Agent Controller is an external agent that handles agent complex querying and messaging to and from JSAF (1). It exchanges simulation information with the Scenario Manager Agent (2). The Scenario Manager Agent handles interpretation and routing of JSAF message content (2), updates World facts (3), and controls the Decision Maker Agent (4). Rather than relying on the Situation Look Ahead Complex to induct asserted facts as beliefs, the Scenario Manager Agent may also communicate current simulation information to it directly as beliefs (5). The Decision Maker Agent reasons about the selection and results from Monitoring Tasks and the selection and results from look-ahead simulation tests. Based on this reasoning, it selects and specifies the alternative task(s) to substitute for the initial or partially completed adaptive behavior task (4). It also controls the Simulation Look Ahead Complex (5).

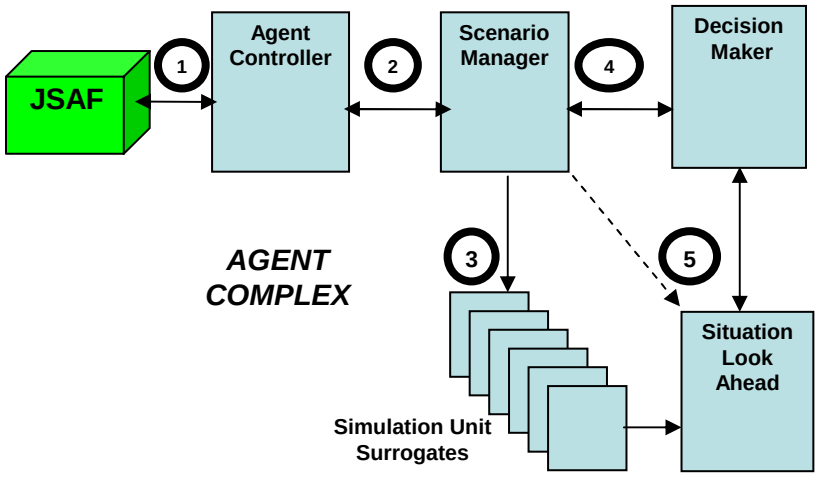

Figure 5. Modular Design of FBA Agent System

The agent capabilities developed for this JSAF implementation were intended simply to demonstrate that an agent of our design could communicate effectively with JSAF in a way that would eventually provide a native solution for adaptive behavior in simulations. These agents exhibit rudimentary decision making compared to what is possible within the FBA architecture. The overall agent capability rests on an agent platform that enables multiple interacting agents to cooperate as situated in a specified operational environment. The capability is able to provide adaptive behavior to many independent units participating in the same simulation. This agent model for behavioral decision making is sufficiently general and flexible to support the full range of possible conventional and asymmetric tactical behavior tasks of forces on all sides.

Look-Ahead Reasoning

The FBA complex of agents conducts each situation look-ahead trial, conveniently referred to as a SLAT, as a sketchy simulation of the unit's current situation to judge possible outcomes of hypothetical unit actions in the immediate future of the actual JSAF simulation scenario. This function is roughly comparable to what the commander of the subject unit might do to respond to the current operating situation and plan his next tactical move.

Each unit assigned at least one adaptive behavior in a JSAF scenario is represented as an agent by FBA. Other units proximal in the scenario that could potentially affect the unit directly or through their shared environment are also represented as agents. These units are called Units of Influence (UOIs) and may be specified by the simulation user during scenario authoring. The SLAT is initialized with the known state of the JSAF simulation at the time that the adaptive behavior determines that the success of the adapting unit's current task is in jeopardy. At this point, the surrogate unit agents are initialized to correspond with the current JSAF simulation state space. Note that including UOIs requires FBA to obtain the simulation state information for these proximal force units, as well as for the adapting unit, from JSAF during runtime. A SLAT run is made for each of the possible alternative unit actions until one resulting in a success outcome is found. Each unsuccessful SLAT returns pertinent information on the run's terminal situation (outcome) resulting from the trial including the progress achieved by the unit (such as its total advance and effect on other units) and its final operational status (such as its health and mobility). This information provides a basis for determining the next most promising action to be evaluated in a subsequent SLAT.

SLATs execute in an agent based simulation "world" that consists of the adapting unit agent (own unit) and selected UOI agents as JSAF scenario unit surrogates. Unit agents in the SLAT world behave in the context of geographic areas and physical objects corresponding to salient environment features of the JSAF scenario which are created in FBA from JSAF scenario-specific data. The activities of a surrogate unit agent are governed by simple agent behavior models for each alternative action defined within the FBA hierarchy of tactical tasks.

In order for the situation look-ahead module of FBA to work, it must be provided with basic information about the current JSAF simulation scenario. The present notion for future FBA development is to provide an adaptive behavior constraint editor as an extension of the JSAF Unit Editor. The editor will do two things: 1) acquire scenario data specified via existing JSAF editors by converting their output into FBA model specifications; and 2) acquire from the scenario author via a compatible GUI the factors constraining the adapting unit's choice of alternative behaviors. At present, FBA considers behavior constraints on four general behavior selection factors – situation monitoring, task alternatives, units of influence, and allowing disaggregation – and the extended editor should allow specification of these factors. Situation monitoring, for example, covers criteria for unit progress, unit health, and unit mobility, where progress, for example, entails rate of advance, effects on enemy, acquisition of resources, and reduction of mission drive.

Reasoning Visibility

The agent platform provides visibility into the agent's look-ahead reasoning via a graphical timeline record of each SLAT as an agent-based simulation run. Figure 6 shows a fragment of one such look-ahead trial where two agents are engaging one another. The weapon carried by one of the agents is also shown in the fragment. The vertical lines indicate the actual shots that are fired and the hits that are taken.

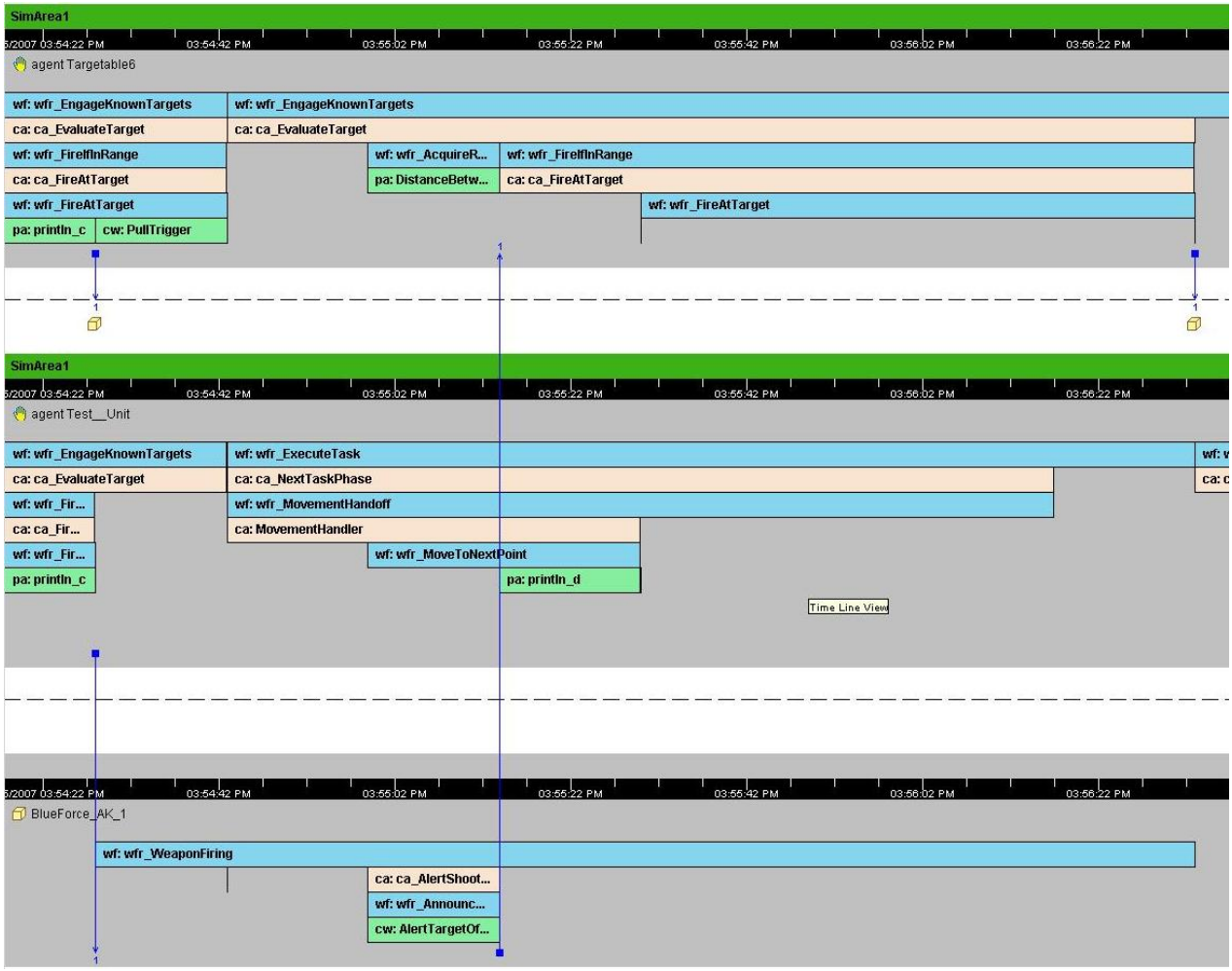

Figure 6. Timeline Fragment from a Situation Look-Ahead Trial (SLAT)

FBA Ontology

FBA relies on a suite of ontologies, expressed in the OWL language, to support agent reasoning about simulation situations and behavior decision making. The ontologies also enable the mapping of SAF simulation concepts to FBA agent concepts. FBA ontologies cover several domains such as: 1) real world concepts like terrain, sides, military equipment and order of battle; 2) operational tasks related by military doctrine; 3) simulation-specific concepts including mappings to corresponding real world concepts; and 4) missions, plans and behavior performance concepts. The simulation domain ontology, for example, specifies entity properties used in determining which Monitoring Tasks to install on a unit's *execution matrix*.

Scenario Authoring

JSAF's unit editor is used to build and edit simulation scenarios. During scenario editing, JSAF users routinely assign specific behavior tasks to constructive OPFOR and BLUFOR units. The FBA implementation extends this editor to allow users to indicate whether they want the behavior task to be performed in an adaptive fashion. This action updates their scenario and the underlying JSAF execution matrix. Ideally, the same scenario authoring environment should also allow the user to specify the desired range of behavior adaptivity in respect to allowable task variation. The GUI of the unit editor extension for specifying adaptive behavior constraints has yet to be implemented in JSAF. This editor will enable the user to define the acceptable range of alternative tasking and, thus, the latitude given to an FBA agent in finding candidate tasks for SLAT evaluation. Any or all of the behavior constraint factors mentioned earlier might be prescribable to a given adaptive behavior task but none are essential.

Ultimately, the information gained from the extended editor input, along with the standard JSAF scenario

specifications, will be processed by a utility to convert it automatically into agent specifications. The conversion happens at scenario design time and provides the agent with scenario and behavior characteristics needed to form the agent model that will be active during simulation runs. This model is actually relayed in runtime as well as design time in parameter messages to the FBA agent.

CONCLUSION

FBA provides a consistent way native SAF behaviors may be made more autonomous using simple but realistic tactical decision making. The emphasis in the early development of FBA has been to establish a robust means of communication between the SAF simulator and the intelligent agents affording the decision making capability. The capability is based on straightforward concepts of behavior task progress in the context of a tactical scenario and how tactical tasks relate doctrinally to one another. This FBA design has been implemented for JSAF where planned entity behavior can be altered dynamically and transparently within the simulator's own task execution control mechanism. Considerable flexibility is achieved to orchestrate adaptive behaviors that can, for example, disaggregate and delegate force subordinates for new behaviors in real time. This includes the ability to change force leadership in mid-course.

FBA also provides the supporting functions and two-way communication necessary to monitor and assess a force unit's health multidimensionally. The results of these intermediate assessments, as well as comparable assessments gained for comparison in the look-ahead simulation trials (i.e., the SLATs) conducted by agents, may optionally be projected to SAF viewers for visibility into the agents' decision making process.

Future Work

The utility of the FBA system depends on its implementation in conjunction with domain ontologies specific to the simulator environment and tactical missions being simulated. For general applicability, the current range of behaviors that can be reasoned about during behavior decision making needs to be expanded across all common tactical tasks. In a separate effort, we have been developing a tactical action ontology initially based on the U.S. Army Universal Task List (AUTL). We have tentatively extracted from this model several aspects of tactical tasks that are more-or-less common to a broad scope of tactical behavior and situations.

The tactical task ontology, for instance, is used by the extended JSAF unit editor as well as by behavior decision agents to select candidate tasks for SLAT evaluation. In the editor, one of the behavior constraints that is asserted during the scenario editing process is the range of alternative tactical tasks eligible for consideration as part of an adaptive behavior response. The user is presented their parent task choices in a task subsumption hierarchy branching to the task they are making adaptive.

There remains a significant opportunity to enhance the intrinsic intelligence of the FBA agents by elaborating their decision making process. Presently, agents select a set of candidate alternative behavior tasks in a simplistic manner, relying on the most doctrinally similar tasks relatively independent of the scenario and current situation context. This selection of candidate tasks to be comparatively tested in SLATs can be improved to be more goal-oriented within the situation context. Basically, this enhancement will depend on axiomatization of the real world tactical ontology to represent the relations entailed in the means-ends analysis of battle goals (and their decompositions), the causal relations this entails, and the representation of command intent in terms of these same goal statements.

FBA domain ontologies also provide a semantic grounding that FBA agents will eventually be able to use to mediate a flexible and general purpose real time exchange of information with other simulators and operational systems. We are currently mapping initial FBA domain ontologies to the Joint C3I Enterprise Data Model (JC3IEDM) in line with how Coalition Battle Management Language (C-BML) and Military Scenario Description Language (MSDL) are accommodating this information model (Hieb and Schade 2007; de Reus et al 2007).

It was the initial objective, and is the continuing focus, of the FBA initiative to provide a behavior decision making agent agile enough to be used with SAFs other than JSAF, such as the U.S. Army's OneSAF, and other live/virtual/constructive simulators. Presently, the agent simulator communication scheme relies on the task frame stack construct, but the agent's SLAT capability is much more generic. We expect to be investigating how FBA can be applied to simulators with different architectures and tied into broader operational initiatives.

ACKNOWLEDGEMENTS

The initial development of the FBA architecture and software was supported in part by the U. S. Joint Forces Command's Joint National Training Capability (JNTC) under award N00140-05-C-0095. We would like to thank Raymond Wade for his guidance on JNTC simulation objectives and their use of JSAF.

REFERENCES

Evertsz, R., Ritter, F. E., Russell, S., & Shepherdson, D. (2007). Modeling rules of engagement in computer-generated forces. *Proceedings of the 16th Conference on Behavior Representation in Modeling and Simulation*. 123-134. Orlando, FL.

Garcia, C. J., & T. W. Griffith (2005). A Composable Behavior Modeling System for Rapidly Constructing Human Behaviors. *Proceedings of the 27th_Interservice/Industry Training, Simulation & Education Conference (I/ITSEC)*, Orlando, FL.

Jones, R., Koss, F., Nielsen, P., & Taylor, G., (2000). Communication with Intelligent Agents. *Proceedings of the 22nd Interservice/Industry Training Systems and Education Conference*, Orlando, FL.

de Reus, N., Borgers, E., Voogd, J., & Huiskamp, W. (2007) Research and Development towards Application of MSDL and C-BML in The Netherlands. *Proceedings of the Simulation Interoperability Workshop, Spring 2007*, Orlando, FL.

Hieb, M. R., & Schade, U. (2007) Battle Management Language: A Grammar for Specifying Reports. *Proceedings of the Simulation Interoperability Workshop, Spring 2007*, Orlando, FL.