SANDIA REPORT
SAND2011-9468
Unlimited Release
Printed January 2012

Hybrid AI/Cognitive Tactical Behavior Framework for LVC

Patrick G. Xavier, Brian E. Hart, Derek H. Hart, Charles Q. Little, Fred J. Oppel III, Nathan G. Brannon, Donna D. Djordjevich, John M. Linebarger, Eric P. Parker

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

Available to DOE and DOE contractors from
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865) 576-8401
Facsimile: (865) 576-5728
E-Mail: reports@adonis.osti.gov
Online ordering: http://www.osti.gov/bridge

Available to the public from
U.S. Department of Commerce
National Technical Information Service
5285 Port Royal Rd.
Springfield, VA 22161

Telephone: (800) 553-6847
Facsimile: (703) 605-6900
E-Mail: orders@ntis.fedworld.gov
Online order: http://www.ntis.gov/help/ordermethods.asp?loc=7-4-0#online

SAND2011-9468
Unlimited Release
Printed January 2012

LDRD Project Final Report: Hybrid AI/Cognitive Tactical Behavior Framework for LVC

Patrick G. Xavier
Brian E. Hart
Derek H. Hart
Charles Q. Little
Fred J. Oppel, III
Interactive Systems Simulation and Analysis

Nathan G. Brannon
W78 System Engineering

Donna D. Djordjevich
Systems Modeling & Software Engineering

John M. Linebarger
Telemetry & Data Systems

Eric P. Parker
Military & Operational Energy Systems Analysis

Sandia National Laboratories
P.O. Box 5800
Albuquerque, New Mexico 87185-1004

Abstract

This LDRD sought to develop technology that enhances scenario construction speed, entity behavior robustness, and scalability in Live-Virtual-Constructive (LVC) simulation. We investigated issues in both simulation architecture and behavior modeling. We developed path-planning technology that improves the ability to express intent in the planning task while still permitting an efficient search algorithm. An LVC simulation demonstrated how this enables "one-click" layout of squad tactical paths, as well as dynamic re-planning for simulated squads and for real and simulated mobile robots. We identified human response latencies that can be exploited in parallel/distributed architectures. We did an experimental study to determine where parallelization would be productive in Umbra-based FOF simulations. We developed and implemented a data-driven simulation composition approach that solves entity class hierarchy issues and supports assurance of simulation fairness. Finally, we proposed a flexible framework to enable integration of multiple behavior modeling components that model working memory phenomena with different degrees of sophistication.

ACKNOWLEDGMENTS

The authors would like to thank the following individuals for their contributions to the project:

Jon D. Bradley
John “Blayde” Jungels
Elaine Raybourn
J. Brian Rigdon
Michael J. Skroch
Kenneth L. Summers
Brian J. Titus
Raymond Trechter
Gregg D. Whitford

CONTENTS

FIGURES

NOMENCLATURE

3D	three-dimensional
AI	artificial intelligence
DHS	Department of Homeland Security
DoD	Department of Defense
DOE	Department of Energy
FOF	force-on-force
LDRD	Lab-Directed Research and Development
LOS	line-of-sight
LVC	Live-Virtual-Constructive
R&D	research and development
SNL	Sandia National Laboratories

1. INTRODUCTION

1.1 Project Overview

Exploiting its three-dimensional (3D) embodied agent simulation technologies, Sandia has delivered force-on-force (FOF) simulation-based tools, such as Dante, to the Department of Defense (DoD), Department of Energy (DOE), and other customers. Using these tools for precision decisions, systems engineering analysis, and other uses is critical. For its main goals, the Hybrid AI/Cognitive Tactical Behavior Framework for LVC project sought to address several limitations of these tools, especially within tactically-intensive 3D scenarios.

- When simulated human entities lack understanding of tactics, complex scenarios required long setup and debugging times. This could be prohibitive for some applications.
- Sandia's Live-Virtual-Constructive (LVC) simulation technology can also be applied to FOF challenges. However, a sufficiently competent automated behavior capability needed to be integrated to apply it to training, analysis, etc., while reducing the number of people required
- Lack of scalability in the LVC framework with respect to constructive entities impeded potential synergy between behavior modeling and LVC systems, such as to address validation and automated model construction.

An important benefit of considering tactical behavior for LVC simulation is the crossover of behavior capabilities and technology among the constructive entities, physical systems in the LVC simulation, and physical systems outside of simulation. For example, in previous work, Sandia has used the same extended state machines framework for agent behavior in a serious game and for autonomous control of a mobile robot.

We sought to develop, implement, and demonstrate advances in a behavior modeling framework that would build upon artificial intelligence (AI) and 3D algorithmic approaches, enable a path forward to including appropriate cognitive modeling, be integrated with a physics-based simulation architecture, and support transfer to physical systems. Our project leveraged the Umbra framework and components from various projects built on it. An Umbra-based approach has previously enabled offline analysis, interactive analysis, and LVC-based studies to use the same models and simulation system.

1.2 Previous Technology

1.2.1 Sandia Embodied Agent Simulation Technology

This project built on and extended prior Sandia expertise in both simulation infrastructure and embodied-agent behavior.

The Umbra framework, now well into its fourth generation, was originally developed over a decade ago to enable embodied-agent simulation for systems engineering analysis. Umbra-based applications typically feature a combination of data-flow-like networks of component

modules that simulate or implement system components and multiple world modules that, together with child modules, simulate interaction phenomena or implement interactions among entities [Gottlieb, et al, 2002]. Initially, Sandia used Umbra for constructing simulations of robotic manipulators (e.g., arms) and mobile robots. Related work at Sandia for DoD, DOE, and LDRD projects has included distributed behavior algorithms as well as low-level planning and control within individual behaviors for both real and simulated mobile robot and mobile robot swarms. Expanding the Umbra technology family, Sandia has developed training systems with simulated adversaries and the Dante FOF simulator. Another direction of R&D developed path-planning and sensor modeling capabilities for use in precision decisions and systems engineering analysis. Leveraging the simulation framework, Sandia developed additional capabilities for Live-Virtual-Constructive (LVC) simulation.[1] A major accomplishment was the development of a testbed for unmanned autonomous system communications in an LVC environment. [Parker, et al, 2009] This technology enables constructive assets to affect and be affected by live and virtual assets via high-fidelity communications models and system-in-the-loop technologies. Demonstrations with a live human, live and constructive unmanned systems, in an LVC environment augmented with constructive obstacles built on this work. Using related technologies, augmented-reality capabilities have been developed and used in fielded human-in-the-loop training systems.

Another family of work grew out of Sandia efforts focused on modeling the human. The 2004-2006 LDRD, "Simulating Human Interactions for National Security Interactions", demonstrated an Umbra-based integration of a partial cognitive framework, SCREAM + SHERCA, with 3D simulated entities [Bernard, et al, 2007]. The 2006-2008 LDRD, "Enabling Immersive Simulation", produced several integrations, including: Trainable Automated Forces and Cognitive Foundry cognitive models with Umbra; and NPS's Delta3D with ABL [Abbott, et al, 2009]. Another LDRD project integrated SCREAM+SHERCA-based cognitive modeling elements into the behaviors of non-player-characters (NPCs) in a serious game for training first response commanders [Djordjevich, et al, 2008]. Results of these projects supported the observation that while it may be scientifically important to expand the domains and levels of abstraction that a cognitive framework can cover, it would be also important to pursue behavior modeling that applies the most capable techniques for different aspects of the problem and enables their integration.

1.2.2 Third-party Technologies

There exists much external technology related to this project. Military FOF simulation systems with entity-level behavior capability include JSAF, OneSAF, VR-Forces, and Stage. The best-in-class systems have as feature high realism (visual, sound, etc.), good simulation engines, solid sets of pre-packaged tactical behaviors that can be composed, and model libraries. However, they all failed to meet our needs in at least one of several crucial ways:

- They are not intended for as fine-grain or high-fidelity simulations as can be built on Umbra.

[1] LVC simulation integrates Live aspects (e.g., real people using real equipment in real locations), Virtual aspects (e.g., real people using simulated equipment in a simulated environment, and Constructive aspects (e.g., simulated people using simulated equipment in a simulated environment) .

- Behavior modeling when powerful is quite rigid due to need to support behavior composition by non-programmers.
- Their intended use model is for programmers only to write primitive behaviors and for SMEs to do the rest.
- Behavior software and capabilities composed on them are not readily portable to outside the simulation system. At the same time, agile extension of capabilities at a source-code level and distribution of the enhanced result is not realistic because of market and organizational factors.
- Their modularity and flexibility seem mainly intended to support construction of exercises and major simulations for analysis, not single agile analysis to cope with or try out something that is new at a component level.

[Abdellaoui, et al, 2009] found that FOF simulation based on serious gaming technology and AI middleware for game development generally suffered from less capability and had similar extensibility problems.

Modeling of cognitive phenomena, such as attention and working memory, are desired in the long run. Soar [Laird, et al, 1987] and ACT-R [Anderson & Lebiere, 1998] are the available cognitive architectures with the most productive research communities. [Best & Lebiere, 2003] and [Wray, et al, 2005] have, respectively, integrated ACT-R and Soar with Unreal Tournament. Silverman [Silverman & Johns, et al, 2006, and Silverman & Bharathy, et al, 2006] models human behavior more broadly, rather than focusing on cognition, and integrates performance moderator functions into a behavior architecture, PMFserv. Multiple behavior architectures for simulated humans interacting within a single virtual training environment was demonstrated separately by [vanLent, et al, 2004] and [McDonald, et al, 2006]. The latter included a framework for a richer set of interactions among human behavior models.

1.3 Summary of Achievements

We began the project by analyzing behavior and Live-Virtual-Constructive (LVC) simulation component technologies and their interdependencies. This engineering analysis included development of a conceptual architecture. The architecture captures relationships and allowable update latencies among behavior modeling and more general LVC simulation elements. We used this engineering analysis to help re-scope[2] the project by priority of unmet technical needs.

Our analysis indicated that we could improve tactical competence and raise the level of abstraction in behavior specification by taking an unconventionally general view of path planning. This enables much tactical knowledge to be expressed in a planning problem. As an example, we addressed the problem of user-specification of team-behavior—which stakeholders had identified as critical—in a demonstration of "one-click" breach team planning. This built on a modular, extensible planner that we had originally developed to support mission planning and analysis. We modified this planner to make it suitable for LVC simulation by encapsulating planning into task objects that are asynchronously served by threads. We then exercised this

[2] This scoping was necessary because our project is the result of first merging two three-year projects post-proposal and subsequently truncating it to two years without any per-year funding increase.

capability in a demonstration where the team members re-plan paths in a tactically coordinated way in response to updated perceptions. We integrated these improved tactical behavior capabilities into an LVC demonstration that includes a constructive breach squad, a live human, a physical unmanned ground vehicle robot (UGV), and constructive UGV and LVC communications. A new planner addresses limitations of the previous planner and introduces new capabilities to better support tactical behaviors, including bounding overwatch.

Practical behavior simulation depends on the simulation infrastructure. Towards this end, we experimentally studied multi-threading the module update loop of the underlying Umbra simulation framework. The results showed the significance of multi-threading Umbra worlds, which model entity interactions, and provide us directions for continued progress. At a higher software level, addressing simulation software modularity, we developed a solution for supporting an ever-growing set of entity types without growing the entity class hierarchy.

Memory and attention are important to behavior modeling both from practical and research standpoints. While simple techniques from game AI can be integrated into simple, low-level, behavior components, they are only sufficient in constrained or idealized situations. However, the science of memory and attentional phenomena is still emerging, particularly with respect to 3D embodied agents. We have proposed an extensible memory framework to enable integration of models of different levels of sophistication and abstraction. We have implemented a preliminary version of its core.

2 PROJECT RESULTS

2.1 Analysis and Scoping

This project was originally the product of merging two 3-year proposals after full-proposal development and submission. One project sought to build on AI and cognitive techniques to improve model fidelity, tactical behavioral competence, and overall robustness of simulated people and other entities in Sandia's FOF simulation systems. The other was broadly directed at creating a secure multiscale LVC framework for adaptive testing, training, and evaluation. Subsequent to the merger but prior to DOE approval, the combined project was truncated to two years without any increase in per-year funding. Needing to share a reduced level of resources, we realized that grand new frameworks for either behavior or LVC were out of the question.

We began by combining the engineering analysis phases from the originally proposed projects. Scaling back from plans for a workshop on Behavior Modeling in LVC, we interviewed eight SNL stakeholders to determine the priority of issues that needed to be addressed for future impact. The top three future needs expressed were to

- raise the level of abstraction needed to specify behavior,
- improve simulation scalability with latency and smoothness sufficient for LVC, and
- improve the modeling of team behaviors.

Another aspect of the analysis considered where LVC needed behavior modeling advances for a team with live, virtual, and constructive human members to function. Two challenges stood out immediately. First, communication between virtual-constructive and live team members needs to be sufficiently natural to the real humans. Second, the constructive team members need the ability to ascertain the state of live members and how they are progressing in their parts of a team plan. While these problems are very interesting, we concluded that they required too much investment in hardware, software, and overall integration before we could make progress. Rather than construct a full experimental LVC environment, we focused project R&D efforts on behavior, with the hope of leveraging opportunities to insert those results into LVC demonstrations.

2.2 Conceptual Architecture

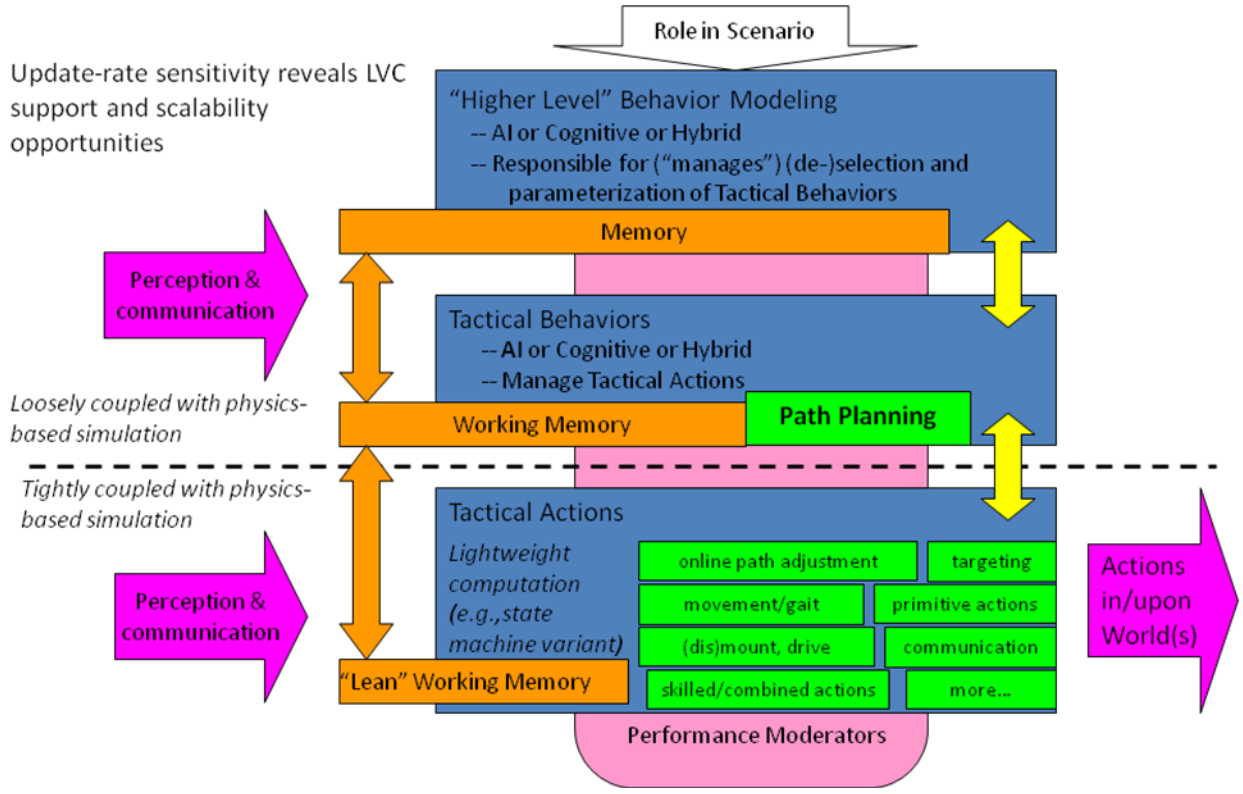
We developed a conceptual behavior simulation architecture with the following main goals:

- capture relationships from both the theoretical behavior-modeling and practical multi-entity 3D simulator system viewpoints;
- illuminate opportunities for R&D advances that would both be ripe for transition and provide long-term technology foundations;
- allow progress to full implementation in flexible phases compatible with plausible project opportunities,
- enable existing technologies to be leveraged, and

- provide opportunities for integrating technologies and future research results from the wider modeling and simulation community.

To allow demonstration of our work in context, we also desired the ability to leverage behavior modeling components existing or already in development in Umbra-based projects.

Viewed at the highest level, the architecture features simulated entities whose action effects and interactions are simulated by world modules that are responsible for different interaction phenomena. It thus builds upon our general approach for simulating embodied agents [WSC 2002 paper] in Umbra. For example, simulated visual perception would be mediated by a visual world module that takes into account line-of-sight, and another world module would simulate weapons effects. A new focus of the architecture is a multiple-level, conceptual human behavior model that features a high-level behavior model, a tactical behavior level, and a tactical actions level. At the high-level and tactical behavior levels, the model is agnostic with respect to whether an AI-based, cognitive, or hybrid approach is taken.



We believe this is practical in terms of computational load because the computational architecture of a simulator has particular opportunities to exploit human response latency at the tactical behavior level and above. Choice reaction time alone offers roughly a half second of latency to absorb parallel/distributed computing and network overhead and latency. When behavior involves an interim action concurrent with more involved decision-making, the latter offers even more time for its simulation. In contrast, with the need to keep up with a 30-100Hz

frame rate, tactical action simulation needs to be more tightly coupled or integrated with the physics-based level of simulation.

Previous research at Sandia has included work on integrating cognitive models with AI-based modeling at lower levels in embodied agent simulation and serious game architectures. These efforts and our previous FOF simulation work either did not address attention, working memory, and episodic/declarative memory or addressed them in very limited, ad hoc ways. Broadly speaking, the conceptual architecture includes memory components whose temporal characteristics correspond to the behavior modeling levels. It is agnostic about the actual inner representations of memory and of working memory in particular. Further description about the memory components is found in Section 2.6.

2.2 Extensible Path Planning

We identified path planning as critical to tactical behavior competence and overall system performance. This is because achieving, maintaining, and avoiding various spatial relationships are often critical elements of tactical behavior, and because it is necessary for entities to move in a timely manner. Our approach was to treat a computer-science-based optimizing path planner as a component that behavior model elements could apply by using a sufficiently rich interface for expressing optimization criteria. In addition, we identified path planning as capability that could exploit asynchronous computation parallel to the main update loop in order to enhance simulation scalability. We sought to develop a path planning framework

- that is modular and expressive enough to greatly raise the level of planning abstraction within behaviors, and
- that is suitable for use in user-driven analysis, offline and LVC simulation, and by autonomous physical systems.

One basic idea is that a tactical behavior can express tactical intent during planned movement as a cost function. For example, a cost function component might return high values at locations where there is line-of-sight to an adversary or on steep path segments. This idea has been explored by both the computer games community and the simulation community, e.g., [Reece, 2003]. A second basic idea is that path planning can be generalized from finding a path to a goal position to finding a path to a position where a goal predicate is true. At Sandia, our department first expressed the path cost as a weighted sum cost functions and used a goal function in implementation at least over a decade ago.

Shortly before the beginning of this project, we began implementing an extensible path planning library that built upon these ideas. This implementation extended our previous work by generalizing from cost functions and goal functions to extensible class hierarchies of “coster” objects and goal predicate objects. Another improvement enabled directing costers to compute their results offline ahead of planning; this data can be accessed in constant time during subsequent path planning.

Our path planning work in year 1 of this LDRD project added several new capabilities:

- expressed planning tasks as objects,
- enabled separate planning tasks to be performed asynchronously in separate threads, and
- integrated movement-related action-planning into path planning.

The resulting planner has been packaged in the Sandia Tactical Path Planning System, Version 1.0, which has been processed for Government Use.

Work in this LDRD project identified limitations in the design of this planner, and in year 2 we developed a new planner that addresses these limitations through new design and new features. This new planner provides a path forward for future development and application. One high-level advance is that the new planner enables multiple planning graphs of different resolutions and connection-generation patterns to be integrated. Because of this feature and extensibility far beyond the previous planner, we call the new planner the "Atlas-Based, Largely Extensible Path Planner", or "ABLE Path Planner". We now describe its key advances.

- A single planning graph is the main part of a chart. A planner's atlas contains a collection of charts and integrates them. For example, this can enable planning across different mapped areas or planning that combines different mapped information. We have defined a base class for charts that declares an interface (as pure virtual functions). If the interface is implemented by a subclass (making it concrete), then a planner can integrate a chart from that subclass into its atlas. The design enables easy extension to include chart classes that use their own methods for constructing their own graphs. Because the design does not require charts to follow a particular meshing or gridding pattern, charts that use geometry dependent, variable-resolution meshing can be integrated, for example. Furthermore, it will enable special-purpose maps to be integrated. For example, the atlas could integrate in a chart of roads, or trails, or tactical positions.

- The new planning software enables integration of problem-dependent topology by allowing problem-dependent edges that reference nodes in the underlying graph.

- The new planning software enables the use of non-binary goal predicates. An ordinary goal predicate has a Boolean-valued function that expresses whether or not its argument is a goal position. A non-binary goal predicate includes an evaluation function that takes a state (position, for this planning package) and computes a cost that is associated with a path ending there. This cost can be weighted when combined with the cost of the rest of the path. A motivating example is the problem of planning a path in order to observe something. There will be some distance at which you can see it sufficiently well, but you can see better the closer you get until you get to an optimum distance. Thus, the cost should be a function that grows as the distance from the observation target increases beyond the optimum. The new planning software can distinguish between when a position is being considered as the end of a path and when it is not, so that it returns the optimal path including this endpoint cost correctly. The algorithmic search variant is embedded into the special search space on which we apply our generic A* implementation.

- A new Property class hierarchy generalizes the previous concept of Costers. While a Coster object computes a real-valued cost (represented by a double in C++) associated with a given position or a path segment, the Property base classes have a DataType template parameter and compute a value of that type that is associated with a given position or path segment argument. Costers are special cases of Properties in the new planning software. The concept of an offline Coster that computes and stores costs associated with nodes and edges in the underlying planning graph and looks up the stored values when called during a path search has been generalized to offline Property classes. The OfflinableProperty base classes have a second template parameter, CachedDataType, to enable more efficient storage. (We refer to the stored property data as "cached".) Read and write access to the cached values is constant-time with respect to each node or edge in the underlying planning graph. The default implementations of Offlinable Properties use templated and base-class code to do their own caching.

- Properties, and in particular offline Properties, can be used to make a planner more efficient in terms of space, software organization, and computational cost. GoalPredicates and Properties (including Costers) can reference other Properties and use them in their computations. For example, the SafetyDistanceCoster from the our older planner estimates whether a position is at least a parameterized distance from obstacles based on 8 line-segment intersection tests (probes). It returns 0 cost for the safe case and infinity for the unsafe case. Because costs are doubles, when the previous planner makes the coster offline, it uses 8 bytes to store a single bit of information. Furthermore, each instance of SafetyDistanceCoster with a different safety-distance parameter requires its own cached data space. By defining an Offlinable Property with a DataType of `bitset<8>`, we can cache the data from the probes in a single 8-byte word. Other Properties and GoalPredicates can use the property, for example, to estimate how (un)stealthy a position is. Furthermore, we can define an Offlinable Property that uses a length-to-intersection version of line-segment testing and stores the result of each test with respect to the probe length as a single byte. The 8 bytes of data for each position can then be used by multiple (approximate) SafetyDistanceCosters with different safety distances, by other Properties, and by goal predicates for which the relative precision is sufficient.

2.3 Leveraging Tactical Path Planning Capability in Behavior Modeling

In project scoping, stakeholders identified team behavior as an area of current and anticipated difficulty. We hypothesized that game-AI-like hierarchical team behavior methods could be combined with the higher level of tactical abstraction afforded by our planning technology to greatly reduce software and use complexity. In year 1, we demonstrated "one-click" planning of a team behavior based on an explosive breach. This showed how our approach accelerates how quickly users can specify team behaviors in a scenario. In a simulation demonstration, the team

changes its plan and members re-plan paths in a tactically coordinated way in response to updated perceptions.

The basic idea behind using path planning in squad behavior is for squad and team member behavior models to take a minimal description of a specific task and elaborate on it to obtain path planning tasks. For example, in the explosive breach example task the minimal description consists of the breach location and the hazard radius of the explosive charge. The behavior model for the task type is responsible for other task parameters that are used in fleshing out costers and goal predicates. In the example, the task goal predicate is a conjunction that includes predicates that account for the maximum permissible distance from the breach location and minimum permissible distance between teammates. Costers penalize locations in the terrain where there is line-of-sight to known adversaries. To avoid the curse of dimensionality, squads take a greedy approach to coordinated path planning. Squad members plan one at a time and pass the set of planned destinations to each successive member so that (s)he can account for them in goal predicates and costers. In the demonstration code, the squad members' AI share access to a data structure with that information, but simulated communication could be used instead in a real application. During a simulation run, the characters' sensing/perception models update the set of known adversary locations. When new threats are observed, the characters can take interim actions while taking turns revising path plans using asynchronous calls to the planning engine. In the simple demonstration for LVC feasibility, each character proceeds on his revised path once planned. Using advances in the ABLE Path Planner in year 2, we built upon this approach to demonstrate an approach for autonomous, extensible bounding overwatch behavior. In particular, we have seen that non-binary goal predicates are valuable for this problem.

Figure 1: Red Squad paths avoid region visible from watch post, then branch to role-specific tactical positions. (Model based on Robot Vehicle Range.)

Early in year 2, we leveraged Sandia's previous work in LVC with physical robots and applied the year 1 path planner to the real and the constructive robots in a demonstration of our LVC technology. In the scenario, a Red Team robot uses notional tactical path planning to flee, avoid, and hide from a Blue Team robot. A line-of-sight based goal predicate models hiding, while a coster that penalizes approaching or getting to close to the adversary is critical for shaping the evasion path. The full demonstration integrates live and constructive assets that include a constructive breach squad, a live human, a physical unmanned ground vehicle robot (UGV), a constructive UGV, and LVC communications. In addition to demonstrating the tactical behaviors integrated into our full LVC technology, this was our first demonstration of our path planning technology for a physical robot.

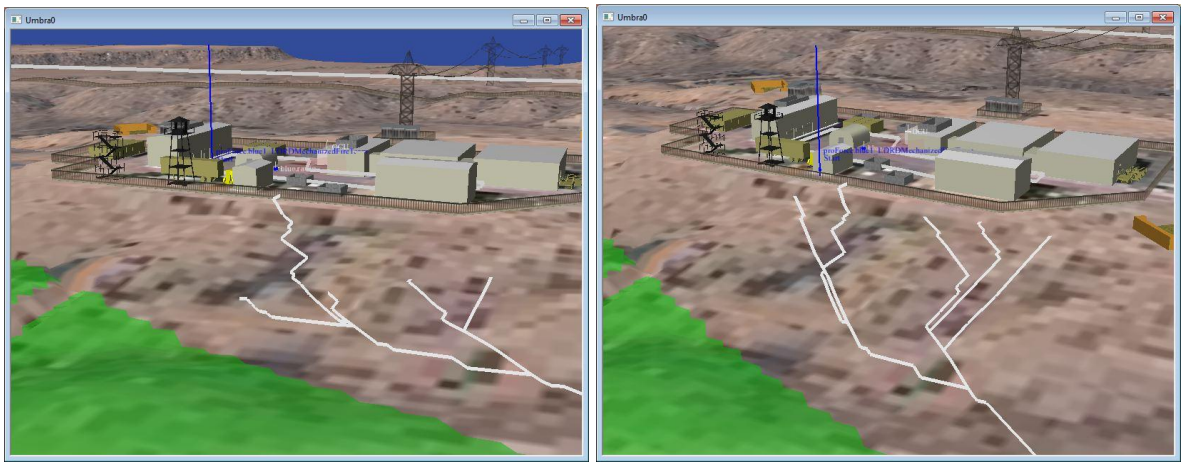


Figure 2: Red squad alters paths in response to Blue guard.

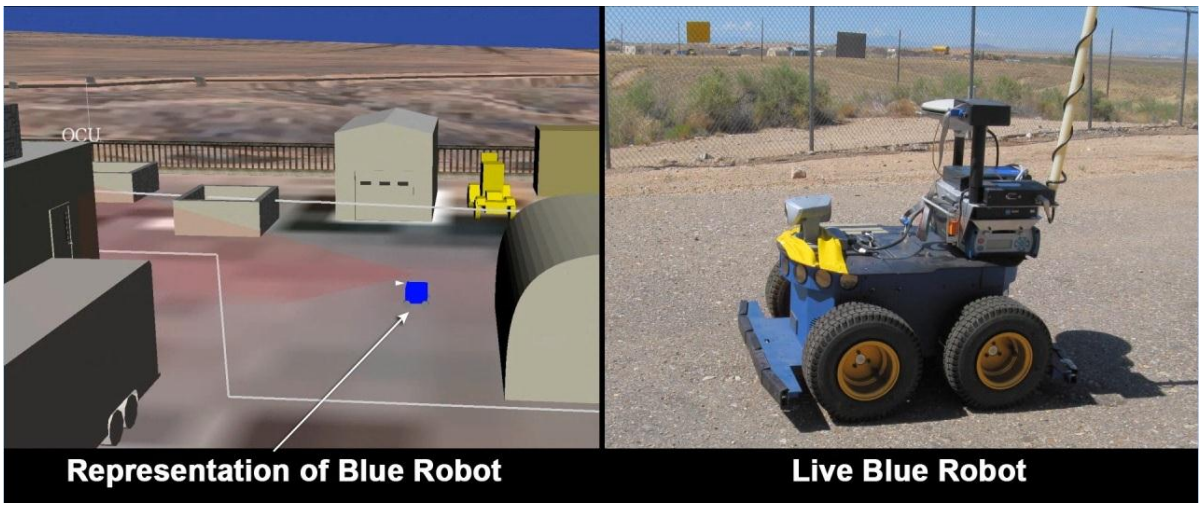


Figure 3: Blue Robot

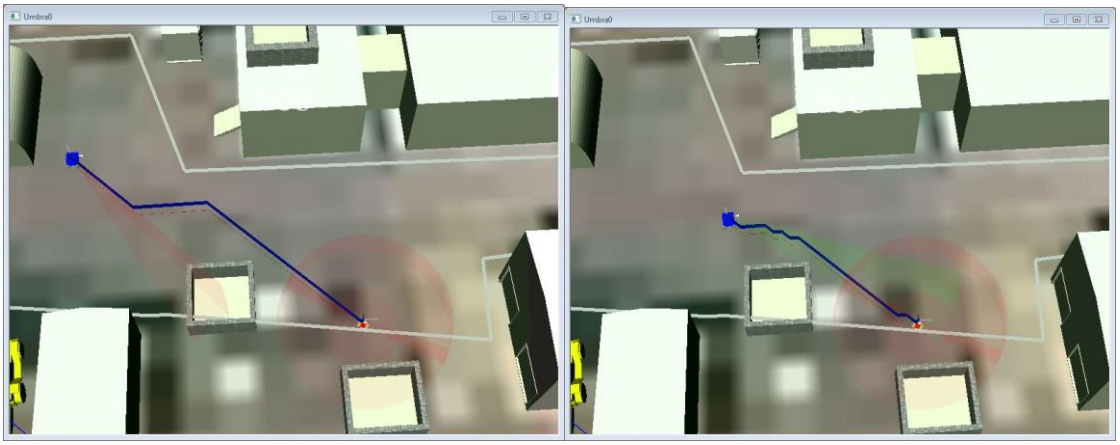


Figure 4: Blue robot plans to Red robot position sent by Blue UAV (not shown), then re-plans path from live position when its simulated sensor detects the Red robot.

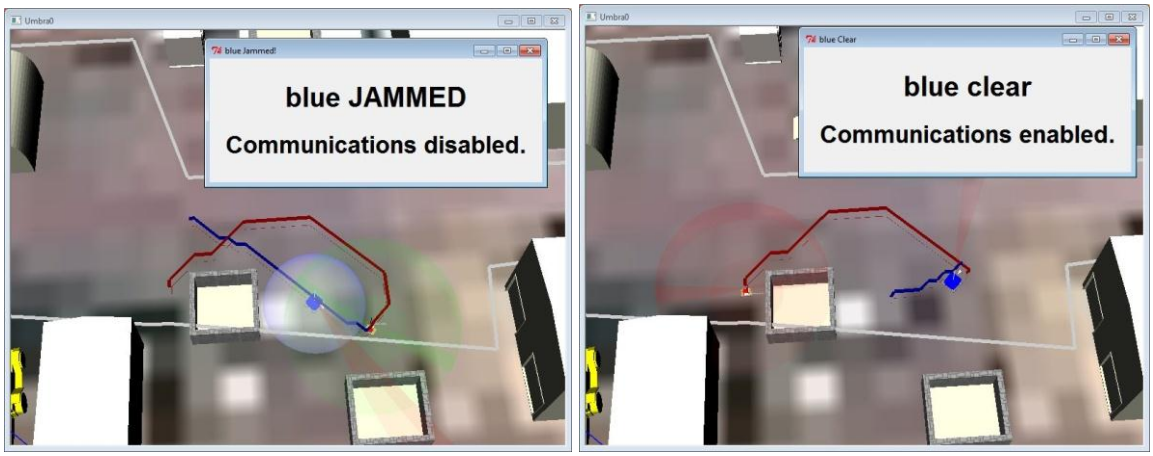


Figure 5: Red robot jams Blue robot radio and plans evasive path to out-of-sight location; jamming ends when line-of-sight is lost.

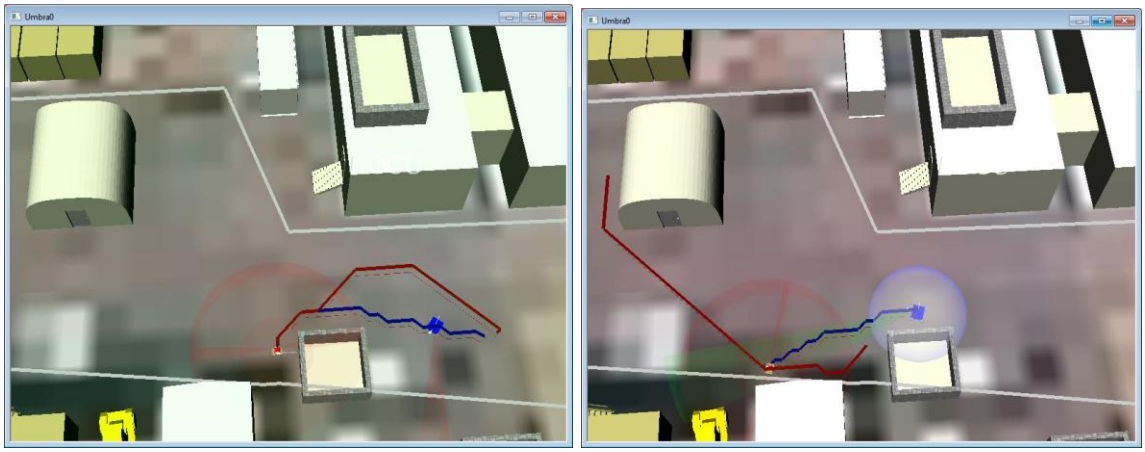


Figure 6: Blue robot plans to Red robot location communicated by Red UAV, then re-plans path to location updated by its own sensor. When there is line-of-sight again, Red robot then jams Blue robot radio and plans evasive path to out-of-sight location.

2.4 Scalability

Off-loading path-planning to separate threads enabled our emerging behavior framework to be applied in a small LVC demonstration. Scalability requires a more general approach to parallelization of the overall simulation. Because our LVC simulation technology and so much of our FOF work has leveraged the Umbra framework, the natural question is how generally to parallelize Umbra-based simulation. Previous distributed Umbra-based simulation includes HLA federation of instances of Umbra runtimes that individually had no explicit parallelization. An end-of-year funding in FY2010 enabled us to conduct preliminary, experimental research on automatic extraction of parallelism in Umbra and on where parallelism should be applied in Umbra-based simulations.

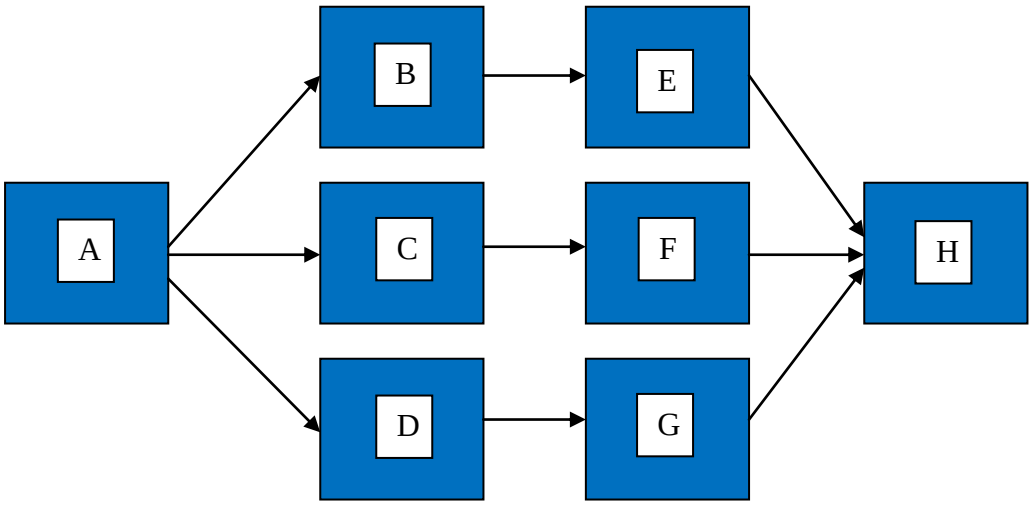


Figure 7: Example MainGraph of Modules.

First, we investigated the feasibility of having the Umbra core automatically extract parallelism. In a simplified view, the main components of an Umbra simulation form an acylic directed graph whose nodes are Umbra Modules and whose edges are non-feedback data connectors. Within an iteration of the update loop (i.e., a simulation step), the `update()` function of each Module is executed in an order consistent with the graph. In FOF simulation, most Modules in the top-level Graph, the MainGraph, are actually Systems, which themselves encapsulate a Graph of lower-level Modules.

Our approach applied multi-threading a step through the simulation loop at the MainGraph level. The basic idea is to use a pool of worker threads to execute the Module::update() in a way that obeys the partial ordering implied by the graph structure. In a legal simulation configuration, the MainGraph will be acyclic. Therefore, it will have at least one Module that has no predecessors. With some simplifying assumptions (e.g., no Callbacks), it is clear that the Module updates can be performed asynchronously while preserving correctness as long as each Module's update begins after the update's of all its predecessor completes. In the implementation, we construct a job for each module:

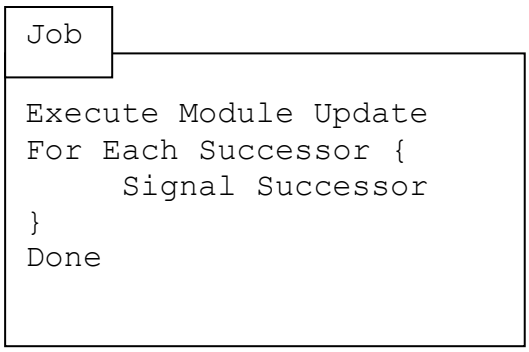


Figure 8: Job executed in a worker thread for a given Module.

When a Module signals its successors, it tells them that one of their predecessors has completed execution. Each Module has a counter that is initialized to the number of predecessors it has. As each predecessor finishes and signals its successors, the count gets decremented. When the predecessor counter reaches 0, then that Module's update job gets added to the thread pool job queue.

To understand where parallelism could generally benefit performance, we experimented with generated a simulation with a number of Dante::Characters. The Dante::Character class is a heavyweight Umbra system that models a human being including features such as visual perception, auditory perception, and mobility. We instrumented the Umbra core to provide some timing data such as time per module update and overall update time.

In initial testing running the threaded update loop on eight threads provided roughly a 50% improvement in speed over single-threaded execution of the update loop. Going from one to eight threads and only getting a 50% speed improvement is poor scaling. Upon further investigation it turned out that over half of the processing time for this test was being done in a single module, the VisualWorld, which computes what each character "sees". When we activated an existing threaded computation capability in VisualWorld, the overall processing time dropped to about 25% of the original processing time (4x speed-up). Another World module, the AcousticWorld, accounted for a substantial portion of the update loop time and would be a candidate for threading.

It is still unclear how much real benefit the threaded update loop could bring to a simulation. Likely, it is highly dependent on the type of simulation being executed. A simulation with little use of Worlds might find improved performance by using the threaded update loop. However, one still has to weigh that improvement against the more restrictive requirements of writing modules to work in a threaded update loop (careful use of callbacks and shared resources such as static member variables). The larger opportunity for threaded optimization could be within the Worlds themselves. Typically, Worlds model interactions of different entities within a simulation. By their nature, these interactions often scale as the square of the number of modules involved, thus consuming a lot of processing time. Each type of interaction also tends to be simulated in a single source code file. This makes Worlds easier to thread without unwanted consequences. Therefore, it seems that threading Worlds would be the most successful approach to gaining performance via threading in the Umbra update loop.

2.5 An Entities and Capabilities Architecture Supporting Fairness

Initial domains for Umbra-based simulations typically consisted of sensors, radio communications and the interaction of mobile robotic vehicle platforms with terrain. However, the set of desired simulated entity types is ever-growing. Interaction phenomena of interest now include weapons use and effects. Entities include human characters with different combinations of roles, force-on-force alignments, skills, capabilities, etc, as well as many combinations of behaviors, mobility, participation in interaction phenomena, and other characteristics. Because of the number of combinations and overlaps, applying classic object-oriented software approaches to define and construct types of entities while avoiding code duplication tends to result in unmanageable class hierarchies or bloat of the base entity class.

We have developed an approach to solve these problems in Umbra-based simulation. The underlying idea is that one basic entity class would be sufficient if capabilities can be added to instances of that class without it knowing about the types of capabilities beforehand. Constructing entities this way from data files rather than procedural scripts is a form of data-driven, component-based simulation composition. Recent game object systems (e.g., [Passos, et al, 2009]) that apply data-driven composition of characters and entities are similar at a high level. Our approach exploits and builds on Umbra's initial world-module and child-module extensions of data-flow-based simulation and also its more recent hierarchical modeling structures. Not only does our approach enable us to solve the problems that confound classical object-oriented methodology, but it enables the data-driven composition of simulations that are fair.

At the heart of our approach are the ComposableBase and CapabilityBase base classes. The main idea is that there will be relatively few, broad subclasses of ComposableBase, such as Character and Vehicle, but many specific types of entities. An entity (instance) of a specific type is created by creating an instance of one of these classes and adding instances of the capability classes that characterize that entity type. Capabilities are aspects of a model and can include sensing/perception subsystems, mobility, participation in various interaction phenomena (e.g., seeable, hearable), and various action (e.g., weapons user) and behavior components. For example, the first step to adding the capability for an entity to participate in an interaction phenomenon entails having the world for that interaction phenomenon create the children needed to model that aspect of the entity.

Our progress builds on two building blocks that have been added to the Umbra core in the past few years. The first is support hierarchical modeling via *systems* and *components*. In our C++ implementation, the System base class is a subclass of the Module class. The System interface enables other modules to be made components of a system module (instance). Component modules belonging to a system module are updated (i.e., have their `update()` functions called) within the scope of that system module's update in the simulation loop. Connectors belonging to component modules cannot be connected to modules outside their owner system module; however, the connectors of that system module can transfer data to/from them via ports. The second building block is basic support for data-driven simulation composition, which Umbra's uxml library provides. The uxml library supports XML templates; many instances of an entity type can be created by using the corresponding template rather than using many copies of the

XML code. The ShapeSystem is a particularly useful base System class that is XML-creatable. It includes a *frame* component module for position and orientation relative to the world frame, and it provides an interface for adding geometry. ComposableBase is a subclass of ShapeSystem.

In the current implementation, we use XML for describing what capabilities to add to an entity. Additional XML data within a Capability XML element is passed to that capability (instance) to customize parameters and construction. The implementation of a capability typically involves several modules, some of which might be systems, in addition to the entity and capability module itself. When a specific capability (instance) is added to an entity, a function is called that is responsible for making sure that the necessary modules are instantiated and for appropriately manipulating connections, ports, system/component memberships, and explicit update-order constraints. The last of these is very important because adding a capability sometimes also involves providing modules pointers to each other. Umbra's scheduling algorithm considers all connections, ports, system/component memberships, and explicit update-order constraints to do the actual scheduling of module updates within the simulation loop. Adding a capability effectively only provides input to the scheduler relevant to the capability implementation but does not do the scheduling.

Simulation applications require that simulation be fair to the extent possible. For example, it is unacceptable for simulation results to be skewed by software object creation order when not differentiated by simulated time. Underlying fairness issues in game implementation can sometimes be hidden from the player by the small timestep in a game combined with the limits of human reaction time. In contrast, simulation applications often seek to run with as large a timestep as possible while still getting valid results.

By connecting chains of modules to the frame component of an entity, capabilities can use that component as an anchor point in the entity's internal constraint graph for component module updates. For example, when a MovementSystem capability is added to an entity, the output of its final output module is connected to the frame module of the entity, and when a VisualSensing capability is added, the module that receives the sensor position and orientation is downstream from the frame module. Other component modules are similarly used to serve the combined role of holding state data and constraining read-write order. Explicit update-order constraints are used when there is a data read-write order that is not expressed through connections, ports, and system/component relationships.

The above methods enable capability class hierarchies (such as a class hierarchy of movement systems) to be written such that when capabilities are used to compose simulated entities, the module update orders are fair and consistent internally across those entities. To avoid conditions that can result in simulation unfairness due to the order of reads and writes of phenomenon-coupled data, we adopt the following constraints. First, writes to the phenomenon-coupled variables of a child module are only done under the control of the `update()` of the world module that produced the child. Second, we avoid interleaving execution of a world module's `update()` with module updates that can entail a read of a phenomenon-coupled variable in any of that world's children. Capabilities are written so that all within an update cycle

- all reads of phenomenon-coupled variables will occur within the scope of an entity's `update();`
- all entity `update()` scopes will precede world `update()` scopes.

When a world module creates a child module, a constraint is added to force Umbra to call the child's `update()` before the world module's `update()` in the simulation loop body. In our new scheme, only a capability type can enable an entity to participate in a simulated interaction phenomenon. A capability that adds an interaction phenomenon to an entity makes such child modules components of the entity system. In Umbra, when a module becomes a component module of a system, the update-order constraints that applied to the module are transferred to apply to the system.

Required world modules can be found by type and/or name via Umbra's instance management system and created if needed. Simulation elements that a capability must attach to or manipulate that are internal to an entity are either part of a specific capability type added to the entity or provided by the broad ComposableBase entity subclass. When a capability is being added, the ComposableBase interface enables it to access prerequisite capabilities or to add them. Thus, our scheme provides Dependency Injection.

2.6 Enabling Flexible Modeling of Working Memory

Modeling working memory and attention are important both from practical and research standpoints. While simple techniques from game AI can be integrated into simple, low-level, behavior components, they are only sufficient constrained or idealized situations. However, the science of memory and attentional phenomena is still emerging, particularly with respect to 3D embodied agents. Although the R&D communities around cognitive modeling frameworks such as SOAR and ACT-R have been developing useful approaches, we desire a behavior framework that enables exploiting straightforward computer science for simulation where it is sufficient.

We have proposed an extensible working memory framework to enable integration of models of different levels of sophistication and abstraction. We have implemented a preliminary version of its core. The focus of this software module is flexibility rather than promoting cognitive modeling fidelity; however, flexibility includes use with cognitive modeling components.

There are several basic ideas behind the design, including:

- Input content can come from perception, communication, and both spontaneous and behavior-requested recall; however, perception is distinguished.
- Memory elements can have different data types.
- Memory elements each have a non-negative activation level that is used to determine availability.
- An overall behavior model will have multiple components that include internal models of how various memory elements should be updated.

The basic question the last of these addresses is how to update activation in the absence of perception. A minimal model for this updating, sometimes previously folded into sensing and

perception or reactive behaviors, would de-activate a memory element when the duration since its most recent perception reaches a threshold. Behavior elements associated with specific contexts might include models that should overrule this default. Even stronger models could include estimation of data and uncertainty from the most recent perception and additional knowledge.

The Flexibly Updatable Memory module provides an interface for inserting memory elements, registering memory element updaters, retrieving active memory elements. Memory elements are identified by a combination of an attribute identifier and a vector of entity identifiers. These identifiers can be generated with or without associating them with a string (name). The basic retrieval function gets all elements with a given attribute identifier and activation over a given threshold. Each memory element has up to three versions of content: what was perceived, what is proposed, and what is the current value that can be retrieved. The general part of content includes activation, the time the content was updated, and the strength of the model that updated the content. The Flexibly Updatable Memory's update function applies the registered updaters to memory elements as appropriate. It applies a default decay model to memory elements that the updaters miss. A memory element updater will not override proposed content set by a stronger model at the current update time. The data part of memory element content is a templated data structure, enabling different attributes to have different types. To enable further generality, there is a simple value template parameter and an estimated value template parameter. The latter, for example, could include statistical data. We assume that the behavior model components that interface with the flexible working memory component will consistently associate attribute identifiers with data types.

3. FUTURE WORK

Because this project produced results at several levels of concreteness and technical readiness, it provides a broad range of starting points for future work. We describe a few directions here.

Although ABLEPathPlanner is already suitable for use in applications, there is much that can be done to extend it. The simplest form of extension is to develop new Costers, Properties, and GoalPredicates. This does not require modifying the library itself. For example, a Coster with knowledge of the locations of non-static obstacles could prevent paths from intersecting them. Behavior modeling research could leverage this simple extensibility by exploring how Costers and Properties could be used to model imperfect and/or limited knowledge of the environment. The ABLEPathPlanner architecture is also designed for additional capabilities to be added to its source code. For example, chart classes that use feature-dependent variable-resolution mesh generation and that integrate road data would be desirable. Planning tasks could include specification of specific charts to include, and hierarchical planning techniques could be built on this capability. Support for charts with local coordinate systems would enable a further extension to make non-static objects traversable. Finally, optimizations should be considered in addition to these and other extensions.

Our work on a flexible working memory simulation framework is ripe for integration into an embodied agent behavior model. As part of an upgrade path for behavior modeling in Dante, perception modules could be adapted to use the Flexibly Updatable Memory and memory element classes instead of having separate representations of memory. We would also develop a knowledge-level communication model that integrates with the new memory model. Behavior triggers, whose activations drive the selection of behaviors within a task context, would also be adapted to use the flexible memory system. Thus, behavior triggers would only have to interface with the memory system instead of each perception modality and communications separately. The same would hold for other behavior modeling elements responsible for context recognition. Finally, work on integrating cognitive modeling aspects, stronger and more general behavior AI, and long-term memory can build on the working memory framework.

Results from this project's limited investigation of scalability using multi-threading include a clear next step. Worlds that simulate entity-entity interactions generally need to be threaded. Also within the multi-threading paradigm, we would investigate patterns and structures that make writing modules to work in a threaded loop less burdensome. Beyond the multi-threading approach, message-passing parallel and distributed architectures for Umbra simulations should be investigated. Latencies between processing nodes within world module and the locations of child modules are obvious issues. However, in an LVC setting, human response latency provides a finitely small threshold to meet.

Additional directions for future work include applying the behavior modeling technology more generally to robotics and addressing low-level aspects of behavior, such as path-smoothing and reactive collision avoidance.

4. CONCLUSIONS

The basic purpose of this project was to advance Sandia's force-on-force (FoF) simulation technologies to better support Precision Decisions and Live-Virtual-Constructive (LVC) simulation. We obtained results in several areas that serve this purpose. The conceptual architecture we develop enabled us to prioritize what components to improve and to identify where latencies in human behavior might be exploitable in a simulation architecture. We extended the capabilities of our previous extensible path planner and demonstrated that it will enable raising the abstraction level at which behaviors can be specified. We applied it to the tactical behaviors of a simulated human squad and live and constructive robots in an LVC demonstration. We developed a new path planner that has greater extensibility in both structure and planning task expressiveness and better space efficiency. We did preliminary experimental work with multi-threading Umbra-based simulation to help define future directions for improving scalability. We developed and implemented a data-driven simulation composition approach based on adding capabilities to a simple entity at construction time that also simplifies assurance of simulation fairness. Finally, we developed the core of a flexible, modular framework for simulating working memory in character behavior.

The results of this project provides foundations for more advanced practical applications development, provides structure into which we can integrate research results, and provides ideas that we can build on in future research.

5. REFERENCES

Abbott, R. G., Basilico, J. D., Glickman, M. R., Hart, D., Mateas, M. McCoy, J. and Whetzel, J.H., (2009). *Enabling Immersive Simulation.* Sandia Report SAND2009-0840. Sandia National Laboratories, Albuquerque, New Mexico, February 2009.

Abdellaoui, N., Taylor, A. and Parkinson, G., (2009). Comparative Analysis of Computer Generated Forces' Artificial Intelligence. *NATO Modelling and Simulation Group (NMSG) Symposium (MSG-069): Use of M&S in Support to Operations, Irregular Warfare, Defence against Terrorism, and Coalition Tactical Force Integration,* Brussels, Belgium, October 2009.

Anderson, J. R. and Lebiere C., (1998). *The Atomic Components of Thoug*ht. Lawrence Erlbaum Associates, Mahwah, New Jersey.

Bernard, M. L., Glickman, M., Hart, D., Xavier, P., Verzi, S. and Wolfenbarger, P., (2007). *Simulating Human Behavior for National Security Human Interactions.* Sandia Report SAND2006-7812. Sandia National Laboratories, Albuquerque, New Mexico, January 2007.

Best, B. J. and Lebiere, C., (2003). Spatial Plans, Communication, and Teamwork in Synthetic MOUT Agents. *12th Conference on Behavior Representations in Modeling and Simulation*, Scottsdale, Arizona, May 2003.

Djordjevich, D. D., Xavier, P. G., Bernard, M. L., Whetzel, J. H., Glickman, M. R., and Verzi, S.J., (2008). Preparing for the Aftermath: Using Emotional Agents in Game-Based Training for Disaster Response. *2008 IEEE Symposium on Computational Intelligence and Games* Perth, Australia, December 2008.

Gottlieb, E. J., McDonald, M. J., Oppel, F. J., Rigdon, J. B. and Xavier, P. G., (2002). The Umbra Simulation Framework as Applied to Building HLA Federates, *2002 Winter Simulation Conference*, San Diego, CA, December 2002.

Laird, J. E., Newell, A., and Rosenbloom, P. S., (1987). Soar: An Architecture for General Intelligence. *Artificial Intelligence*, **33**(3): 1-64.

McDonald, D., Lazarus, R., Leung, A., Hussain, T., Bharathy, G., Eidelson, R. J., Pelechano, N., Sandhaus, E. and Silverman, B. G., (2006). Interoperable Human Behavior Models for Simulation. *15th Conference on Behavior Representation in Modeling and Simulation*, Location?, Month? 2006.

Parker, E. P., Miner, N. E., Van Leeuwen, B. P. and Rigdon, J. B., (2009). Testing Unmanned Autonomous System Communication in a Live/Virtual/Constructive Environment. *ITEA Journal*, **30**:513-522, International Test and Evaluation Association, December 2009.

Passos, E. B., Sousa, J. W. S., Clua, E. B. W., Montenegro, A. and Murta, L., (2009). Smart Composition of Game Objects Using Dependency Injection. *ACM Computers in Entertainment*, **7**(4), Article 53, December 2009.

Reece, D. A., (2003). Movement Behavior for Soldier Agents on a Virtual Battlefield. *Presence: Teleoperators & Virtual Environments*, **12**(4): 387-410, August 2003.

Silverman, B. G., Johns, M., Cornwell, J. and O'Brien, K., (2006). Human Behavior Models for Agents in Simulators and Games: Part I: Enabling Science with PMFserv. *Presence: Teleoperators & Virtual Environments*, **15**(2): 139-162, April 2006.

Silverman, B. G., Bharathy, G. K., O'Brien, K. and Cornwell, J., (2006). Human Behavior Models for Agents in Simulators and Games: Part II: Gamebot Engineering with PMFserv. *Presence: Teleoperators & Virtual Environments*, **15**(2): 163-185, April 2006.

Van Lent, M., McAlinden, R., Probst, P., Silverman, B. G., O'Brien, K. and Cornwell, J., (2004). Enhancing the Behavioral Fidelity of Synthetic Entities with Human Behavior Models. *13th Conference on Behavior Representation in Modeling and Simulation*, Location?, May 2004.

Wray, R. E., Laird, J. E., Nuxol, A., Stokes, D. and Kerfoot, A., (2005). Synthetic Adversaries for Urban Combat Training. *AI Magazine*, **26**(3): 82-92, Fall 2005.

APPENDIX A: PROJECT-RELATED INFORMATION

Awards: None.

Publications and Presentations: None.

Patents: None.

Disclosures of Technical Advance:

- (In preparation.) A Flexible Path Planning System
- An Entities and Capabilities Architecture Supporting Simulation Fairness, SD 12203.
- (In preparation.) Path Planning Architecture for Expressiveness, Extensibility and Efficiency.

These titles are tentative.

Copyrights:

- Contributed to development of Sandia Tactical Path Planning System v. 1.0 beta. SCR 1427.0, Government Use.
- Contributed to development of State Machines. SCR 1348.0.
- A copyright assertion will be filed for the path planning framework described in Section 2.2, "Extensible Path Planning". It is also believed that the library will be included in copyrighted distributions of applications such as Dante 2.x, Umbra Terrain Utility and OpShed.
- The Capabilities/Entities implementation (Section 2.5) is being copyrighted. It will also be included within higher-level (e.g., application) distributions.
- SquadCoordinator is being copyrighted. It will also be included within higher-level (e.g., application) distributions.

Follow-on Work:

- Path planner being incorporated into several projects:
    - Dante 2.0 and later
    - Umbra Terrain Utility, next version
    - OpShed, next version
- "One-click squad planning"
    - Dante 2.0 and later

    - Close Quarters Battle simulation and training projects
- Entities and Capabilities: now one of the three major modularity patterns used in writing Umbra-based libraries and applications.
- Flexible Working Memory framework expected to be starting point for new memory model in Dante 2.x.

Distribution

1	MS 0483	Nathan G. Brannon, 02112 (electronic copy)
1	MS 0986	John M. Linebarger, 02661 (electronic copy
1	MS 1004	Brian E. Hart, 06134 (electronic copy)
1	MS 1004	Derek H. Hart, 06134 (electronic copy)
1	MS 1004	Charles Q. Little, 06134 (electronic copy)
1	MS 1004	Fred J. Oppel, III, 06134 (electronic copy)
1	MS 1004	Michael J. Skroch, 06134 (electronic copy)
1	MS 1004	Patrick G. Xavier, 06134 (electronic copy)
1	MS 1138	Raymond Trechter, 06130 (electronic copy)
1	MS 1161	K. Terry Stalker, 05447 (electronic copy)
1	MS 1188	Eric P. Parker, 06114 (electronic copy)
1	MS 1188	Richart O. Griffith, 06130 (electronic copy)
1	MS 9406	Donna D. Djordjevich, 08116 (electronic copy)
1	MS 0899	RIM-Reports Management, 9532 (electronic copy)
1	MS 0161	Legal Technology Transfer Center, 11500 (electronic copy)
1	MS 0359	D. Chavez, LDRD Office, 1911 (electronic copy)


Sandia National Laboratories