Subject Matter Expert-Driven Behavior Modeling Within Simulation

Jason R. Potts, Todd Griffith, Ph.D., Joseph J. Sharp, Dan Allison
Discovery Machine Inc.
Williamsport, PA
jpotts@discoverymachine.com,
tgriffith@discoverymachine.com,
jsharp@discoverymachine.com,
dallison@discoverymachine.com

ABSTRACT

Development of realistic entity behaviors within constructive training simulations presents many issues. Autonomous and semi-autonomous entities provide trainers with the ability to conduct large scale training exercises with limited resources, but the specification of these entities' behavior yield new issues that are not present when conducting traditional training exercises. Training requires that autonomous or semi-autonomous entities behave in well-defined ways that correctly mimic the behavior of their real-world counterparts. Failure to behave correctly can result in not only missed training objectives, but negative training if trainees learn to respond to constructive entities exhibiting unrealistic behavior. This paper presents work conducted under a combination of projects for NAVAIR Training Systems Division, the Office of Naval Research, and Joint Forces Command, demonstrating a customized behavior modeling framework for specification of autonomous entities. Behaviors were specified for entities within the Joint Semi-Automated Forces (JSAF) simulation environment, as well as an immersive 3D environment developed by Northrop Grumman called the Irregular Warfare Virtual Trainer. This paper presents a framework for enabling subject matter experts (SMEs) to develop behaviors using modular basic-level actions (BLAs) through an easy to use wizard-like interface. This interface assisted SMEs in representing their expertise regarding how entities within a training system should behave, in a manner which enabled execution within the simulated environment. We present a simulation-agnostic architecture for modeling fully executable entity behaviors through a visual task decomposition, as well as a methodology for enabling subject matter experts to directly develop and maintain entity behaviors without the assistance of a software engineering team.

ABOUT THE AUTHORS

Jason Potts is the Director of Intelligent Systems for Discovery Machine Inc., overseeing the software development of their core technology offerings, as well as the application of their technology to customer problems across a variety of domains. Since graduating with honors from WPI with a Bachelor's of Science degree in Computer Science, Mr. Potts has been designing and developing software for Discovery Machine Inc. since 2001. This work has focused on development of tools facilitating knowledge capture, and representation of human expertise in a machine executable form. In addition to behavior modeling of simulated entities within constructive military training environments, these applications also include representation and automation of business process intelligence within commercial sectors such as manufacturing or pharmaceuticals.

Dr. Todd W. Griffith is the founder and CTO of Discovery Machine. He is responsible for acquiring funding and managing the research and development of Discovery Machine's core knowledge acquisition products, the Discovery Machine Knowledge Service Modeler™ and Knowledge Service Engine™. He has been working in the area of intelligent systems research for eighteen years and has published papers in the areas of cognitive science, human-computer interaction, and intelligent systems. He has presented regularly at conferences and meetings since 1996 and has received a patent in the area of knowledge systems. Prior to founding Discovery Machine, Dr. Griffith

taught computer science and artificial intelligence at Georgia Institute of Technology and Bucknell University. Dr. Griffith's received his Ph.D. in computer science and artificial intelligence in 1999 from Georgia Tech with a focus on modeling the problem solving strategies of Subject Matter Experts (SMEs). A lack of available tools for knowledge acquisition, led to the founding of Discovery Machine and the commitment to build software that enables SMEs to encode their *own* problem solving strategies on the computer. In support of this effort, Dr. Griffith has been the principal investigator in research projects funded by DARPA, ONR, NWDC, NWSCDD, OSD, Army ERDC, NAVAIR, NASA, and NSF.

Joseph J. Sharp is a Software Engineer employed by Discovery Machine, Inc. He is a 2009 summa cum-laude graduate of Bloomsburg University of Pennsylvania where he studied and earned a Bachelors of Science degree in computer science. During his time at Discovery Machine, Joe has been involved in various behavior modeling projects, where he has played a significant role in developing technologies to aid in the process of using Discovery Machine tools to model complex human behaviors.

Dan Allison is the Director of Knowledge Engineering and Professional Services for Discovery Machine Inc. Mr. Allison has 20+ years of experience in system engineering in a variety of businesses including DoD SONAR and RADAR systems, commercial genetic analysis systems and FDA regulated medical devices. His technical emphasis has been system analysis, algorithm design and requirements management. His most recent work focuses on knowledge engineering using the Discovery Machine Knowledge Capture Methodology and building deployable knowledge models using Discovery Machine's Modeler software. Mr. Allison earned a BSEE from Penn State and MSEE from Syracuse University and is six sigma certified. He has received 2 patents on the topic of deconvolution algorithms and has been an active member of both IEEE and INCOSE.

Subject Matter Expert-Driven Behavior Modeling Within Simulation

Jason Potts, Dr. Todd Griffith, Joseph Sharp, Dan Allison

Discovery Machine Inc.

Williamsport, PA

jpotts@discoverymachine.com,
tgriffith@discoverymachine.com,
jsharp@discoverymachine.com,
dallison@discoverymachine.com

INTRODUCTION

As virtual training environments using constructive entities play a more central role in developing warfighter capabilities, a number of issues arise. Autonomous and semi-autonomous entities enable large scale training exercises with limited resources, but the specification of these entities' behavior yields new issues that are not present when conducting traditional exercises. Training requires that autonomous entities behave in well-defined ways that correctly mimic their real-world counterparts. Failure in this regard can result in not only missed training objectives, but negative training if trainees learn to respond to constructive entities exhibiting unrealistic behavior.

This paper presents a behavior modeling framework for constructive entities which enables subject matter experts (SMEs) to develop complex entity behaviors using modular basic-level actions (BLAs) through an easy to use wizard-like interface. This interface assists SMEs in representing their expertise regarding how entities within a training system should behave, while preserving that expertise in an easily understood form through the entire development process and during runtime execution within the simulation environment. Elements of this behavior modeling approach have been applied in two domains, each of which presents advantages of this technique.

We present work funded by NAVAIR-TSD and the Office of Naval Research, demonstrating a customized behavior modeling framework for specification of constructive entities within the Joint Semi-Automated Forces (JSAF) simulation environment. This application of our approach enables the rapid development of new behaviors for subsurface vehicles as well as rotary-wing aircraft, which in turn enables the development of customized scenarios by trainers without the need for additional software development or additional manpower in the form of JSAF operators. Thus, through use of the techniques presented here, training exercises can be increased in their scale and complexity, more directly achieving their training objectives, without increasing the resources required.

Additionally, we present work conducted for Joint Forces Command in conjunction with Northrop Grumman. In this effort, we applied elements of our behavior modeling technique to represent non-player characters (NPCs) within the Irregular Warfare Virtual Trainer (IWVT), an immersive 3D simulation built upon the Delta-3D platform. The domain of irregular warfare presents unique challenges for behavior modeling, due to the rapidly evolving behaviors required for NPCs representing populations in different parts of the world. By enabling SMEs to directly define behaviors for use within virtual training simulations, we facilitate development and customization of those simulations to match changing situations in the real world. Users can adapt insurgent tactics within the simulation to match observed adaptations by their real-world counterparts. SMEs with expertise in the culture of a specific region (i.e. cultural anthropologists, warfighters with experience in the field, etc.) can directly modify NPCs to behave in a manner consistent with that specific region, enabling trainees to develop cultural awareness and the intra-personal skills necessary to effectively interact with a local population.

JSAF Behavior Specification

Under projects funded by NAVAIR-TSD and ONR, customized behavior modeling environments were developed for specification of JSAF behaviors driving subsurface vehicles and rotary wing aircraft. These modeling environments enabled creation of new missions by experts through modular composition of BLAs developed for a particular domain.

Using the process presented in this paper, we built 26 new BLAs for helicopters and 36 BLAs for subsurface vehicles within JSAF. All of these BLAs are functional and tested in the console. Many of them have been used in missions we constructed to test our behaviors in JSAF. The effort spent on knowledge acquisition, development and testing these 35 new BLAs was approximately 70 man-days. This results in about 2 man-days per BLA for BLAs which are functional for demonstration. More testing is done on each BLA to develop them for deployment, but the total time per BLA is still no more than 5 man-days from start to finish on average. It is important to note that this is an average time. More complex BLAs may take longer and simple BLAs will be quicker to develop and test. This level of effort is several orders of magnitude less (from months to days) than what it would take to develop this much new functionality by writing and debugging code to add new task frames to JSAF.

After development of the behavior modeling environments, called *consoles*, and the initial basic level actions for each domain, new missions for a particular entity type can be created by a user with no software engineering expertise in a matter of minutes. They can then be deployed directly from the modeling environment into JSAF while the system is running, *without needing to recompile source code or even stop the simulation*. While these behaviors can be rapidly constructed, they can display extremely complex and nuanced behaviors as a result of the knowledge encoded within the BLAs.

Missions developed using this approach utilize complex cognitive modeling of the entities' environments, while containing transparent representations of expertise captured directly from SMEs. Some of the more complex aspects of the behaviors captured include communication between autonomous helicopters and human crews and Anti-Submarine/Anti-Surface Warfare Tactical Air Controllers (ASTACs), as well as making decisions based upon mental models of the entity's opponents.

Irregular Warfare Virtual Trainer Behavior Specification

As part of an effort funded by Joint Forces Command, behavior models were developed to represent complex behaviors in a 3-D training simulation is the Irregular Warfare Virtual Trainer developed by Northrop Grumman. The goal of this simulation was to train soldiers to be able to maintain effective situational awareness in a complex and crowded environment, in this case, an Afghani bazaar. We developed behaviors to control all entities within the training simulation which encompassed virtual characters ranging from vendors and villagers, to possible insurgents, a personal security detail, and the center piece of the simulation, a village elder that could interact in a complex cultural interview with the trainee.

The approaches described in this paper allowed for the development of very complex and culturally interesting interaction between the trainee and the virtual characters and between individual virtual characters. The trainee was presented with a crowded marketplace, and it was his/her responsibility to navigate through it effectively and safely. Certain behavior models that were designed to stand out to the trainee would cause characters in the simulation to display body language that was culturally representative of disrespectful and indignant attitudes. At points in the training, the trainee is instructed to point out suspicious characters, at which point they have to rely on clues such as body language to determine possible suspects.

Other behaviors in the simulation included common villager behaviors, which while less interesting on the surface, enabled realistic interaction between the virtual villagers. The villagers had the ability to approach vendors and converse about various products for sale and prices of those products. These conversations were accomplished by sending information from one entity to another in coordination with playing the accompanying audio. Entities, upon receiving an incoming message, could parse the message and determine an appropriate response. The response was then, in the same manner, sent back to the initial entity, that could then parse and respond as appropriate. This conversation would last until an entity decided to leave the conversation. The overall result was an interesting and realistic Afghani marketplace filled with virtual characters that acted in a culturally appropriate manner.

The behavior models used in this training environment produced after action review (AAR) materials during execution. Player interactions with the various NPCs were recorded to an external source (in this case, Microsoft Excel spreadsheets), as well reported to the game environment for display during a review upon completion of the mission.

Advantages

This approach to behavior modeling offers the following advantages:

- Rapid development of new behaviors for customized training scenarios

- Expert domain knowledge captured within the modeling tools to assist building more intelligent behaviors
- Transparency of behaviors for subject matter experts during and after development
- Transparency of behaviors at runtime, easing the debugging process and exposing decision making process of entities to operators

BACKGROUND

Behavior modeling, according to our approach is not achieved through finite state machine approaches, nor through rules applied to dynamically create problem spaces, as one might find in cognitive architectures (Newell, 1990) (Anderson, 1982) but by the storage of a set of process models from the outset. We call these process models "strategies" and rules are stored within these strategies. The pedigree of our approach comes from two places. The first is the Case-Based Reasoning (CBR) community (Kolodner, 1993), which treats storage and retrieval as the primary mechanisms of intelligence. Like the CBR community we believe that patterns are often stored and retrieved. The idea that we store both conceptual knowledge as well as process knowledge in the form of patterns should not be surprising to anyone who studies cognition.

The storage of such knowledge in production rules, although convenient from a computational perspective, seems quite improbable from a cognitive perspective. Patterns of process are captured in the form of task decompositions, which leads to the second place of origin for our ideas. Chandrasekaran (1986) proposed that knowledge patterns pertain to processes as well as concepts. He called these patterns of process "generic tasks". The notion that one weak method (i.e., heuristic search through problem spaces) could adequately account for human cognition did not seem reasonable to him, so he suggests the idea of patterns of process which would apply across domains. Heuristic search and case-based reasoning are simply two examples of generic tasks used by intelligent systems.

Our task-method decomposition hierarchies (strategies) represent patterns of behavior that can be applied to specific kinds of problems. Experts know these patterns and are able to recall and apply them. The process is not a search for a rule but is matching a pattern and executing a strategy. The system can simply execute the strategy to solve the problem; the strategy can be composed from more basic task-method hierarchies as well as other representations. The BLAs described in this paper are stored patterns which we present to SMEs for modification and reuse.

One of the most significant differences between our approach and past efforts in modeling human behavior is the way we elicit knowledge from subject matter experts. Our methodology involves the direct capture of strategic knowledge from experts in a visual form that the subject matter experts can understand from inception to deployment. Other cognitive modeling languages capture rules that are dynamically applied to enable the entity to make decisions. Our strategies are more case-based in nature. Although there are rules throughout the hierarchical representation, those rules are generally localized to a specific context which makes them significantly simpler to acquire and adapt.

Rule formation and modification have proven to be difficult problems since their inception nearly 50 years ago. Our research has determined that strategy capture in the form of hierarchical task networks (HTNs) in the task, method, knowledge (TMK) language can be easy through a visual/diagrammatic interface. The real advantage is that the visual representation provides context to the subject matter expert, which enables him to focus on the right questions leading to better elicitation of knowledge. The knowledge engineer can then ask better questions since the expert is focused on the same issue. In a rule based system the knowledge engineer must lead the expert through a set of examples and rules, neither of which provide the expert with the context necessary for uncovering the exceptions that are crucial to intelligent behaviors. In a strategy-based approach the expert walks through a set of situations providing the expertise based on the situation.

In contrast to a script or lisp-like representation; the visual, hierarchical nature of our models makes them transparent to the expert which enables them to evaluate the actions in context. The hierarchy provides a benefit to the experts in that it enables them to visualize their processes, which helps them to understand those processes better.

Our approach provides both an effective way to acquire important domain specific strategies, and also to connect to existing task-frames and actions within SAF systems. Once you have the expert, the next problem is how to discern what the expert knows and how the expert will use the knowledge in a problem solving scenario. Using our approach this is all transparent. We obtain the patterns and strategies of experts and we require it to be easy for the experts to understand what the model says. Thus, we provide a more explicit approach to matching the entity's behaviors to what a

SME would do, and further, we enable the SME to iterate over the representation until it is correct.

ARCHITECTURAL OVERVIEW

Figure 1 presents an overview of the system architecture used to enable construction of autonomous entities for a simulation developed here. It consists of four major components:

- The simulation used as the training environment
- A thin interface layer providing a set of functions for behavior models to use in communicating with the simulation
- A module for managing the various behavior models developed by SMEs, as well as their instantiation within the simulation
- A framework for developing knowledge capture environments (KCEs), or consoles, specifically tailored for development of behavior models within a particular domain

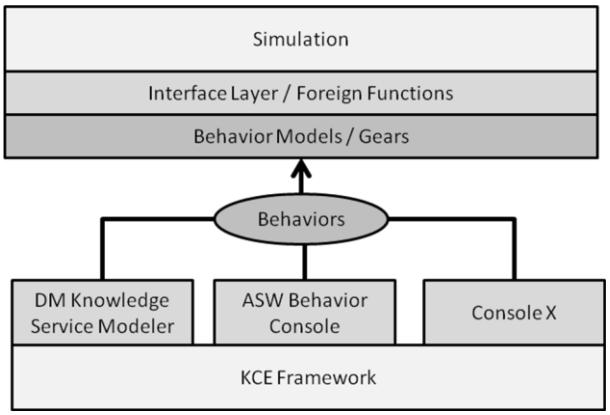

Figure 1: High Level System Architecture

Simulation Interface Layer

Between the simulation and the behaviors lies a thin interface layer that provides the following core functionality:

- Action functions – functions called by the behavior model to cause the entity to perform actions in the simulated environment, such as movement to a particular location, turning sensors on/off, or firing a weapon.
- Query functions – functions called by the behavior model to request information about the simulation, such as the current state of the entity driven by the behavior model or what sensory stimuli are currently being observed by it. The key differentiator between queries and actions is that queries are expecting a response from the simulation.
- Data structure specifications – a specification of data types passed back and forth between the simulation and the behavior models, used by the functions described above.

In addition to these three core components of the interface layer, a mechanism is required to synchronize the behavior execution with the rest of the simulation environment. In practice, many operations performed in the behavior models are internal to the models themselves, and so synchronization needs only to occur at the point where communication between the behavior model and the simulation takes place.

Figure 2 shows a diagram of the communication that takes place between the simulation and the behavior manager at runtime. This particular diagram shows the communication mechanism used during integration with NPCs in the IWVT, but the mechanism can be adapted to use a variety of different methods to communicate between the behavior models and the simulation. For instance, our integration with JSAF leveraged a very similar overall communication mechanism, but it took place over the Java Native Interface, passing the same types of calls directly though function calls to dynamic libraries, sending Google Protocol Buffers in the place of XML packets to achieve improved efficiency.

The key point to note is that the system is *simulation-agnostic*, and can be adapted to integrate with various simulation environments. If multiple simulations support the same set of sensory input and actions required by a behavior's model, that model can be deployed in both simulations without modification. Additionally, the TMK structure of the behaviors supports failure-based decision making, so that a given task can have multiple methods of completion, each attempted in series until one succeeds in achieving the goal of the task. Thus a given task may specify less efficient methods of completing its goals, which the system can call back upon if the integration with a particular simulation environment does not support all of the functionality required by the preferred method.

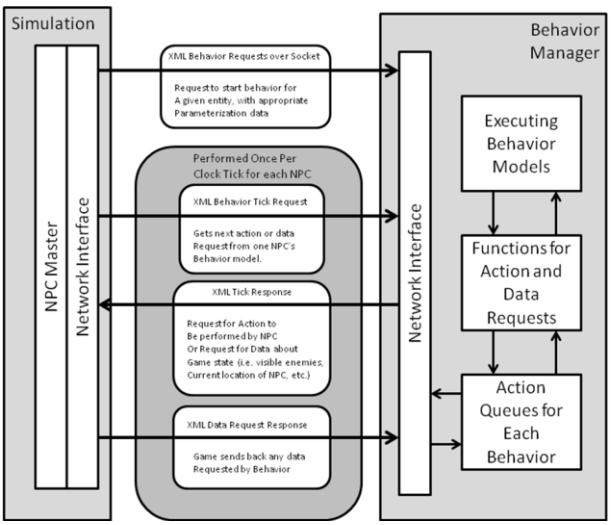

Figure 2: Communication between Behaviors and the IWVT Simulation

Behavior Model Structure

Figure 3 shows the top level structure of a typical behavior model represented in TMK. The slashed background of the subtask group indicates that the three subtasks (Process Situational Awareness, Perform the Main Mission, and Monitor Sensors) execute concurrently.

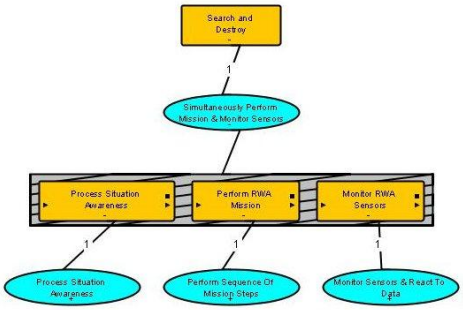

Figure 3: Top Level Structure of a Behavior Model

These three subtasks represent the following three primary components of a behavior model:

- Situational Awareness Processing – This subhierarchy processes raw data about the entity and the simulation environment, and uses it to generate higher level mental models of the world in which the entity exists. This allows the behavior to be defined using concepts that a human would take for granted, without having to delve into the details of how to derive knowledge of the simulated world from low level data. This section of the behavior model is specified in more detail below, in the section titled "Enabling Situational Awareness".
- Mission Execution – This subtask consists of a series of BLAs representing the primary mission to be performed by the behavior. These subtasks are executed sequentially, with the possibility for iteration over the set of BLAs. This is the section of the model with which the SME most directly interacts at behavior composition time, building up a mission by choosing which BLAs he wants to combine.
- Reaction to Events – This subtask allows the behavior to interrupt execution of the primary mission, and react to unscheduled events in the simulation. For instance, if an entity comes under fire, it will need to react to that event in a timely manner, most likely delaying the execution of its current mission. This branch uses a blackboard system combined with listeners to enable the SME to define layered reactions within a subsumption architecture.

Mission Sub Hierarchy Structure

Figure 4 shows an example of the mission section of the Search and Destroy behavior, constructed for use by a rotary winged aircraft. The objective of the Search and Destroy RWA mission is to get a contact on an enemy submarine with a high level of confidence that the contact is valid and the contact details are accurate, move to within torpedo range of that contact, and deploy a torpedo against that contact. An example application of this mission would be in a defensive search and destroy mission where and SH-60 is tasked with finding an enemy sub and destroying it before it is able to launch an attack on a friendly high-value unit (HVU).

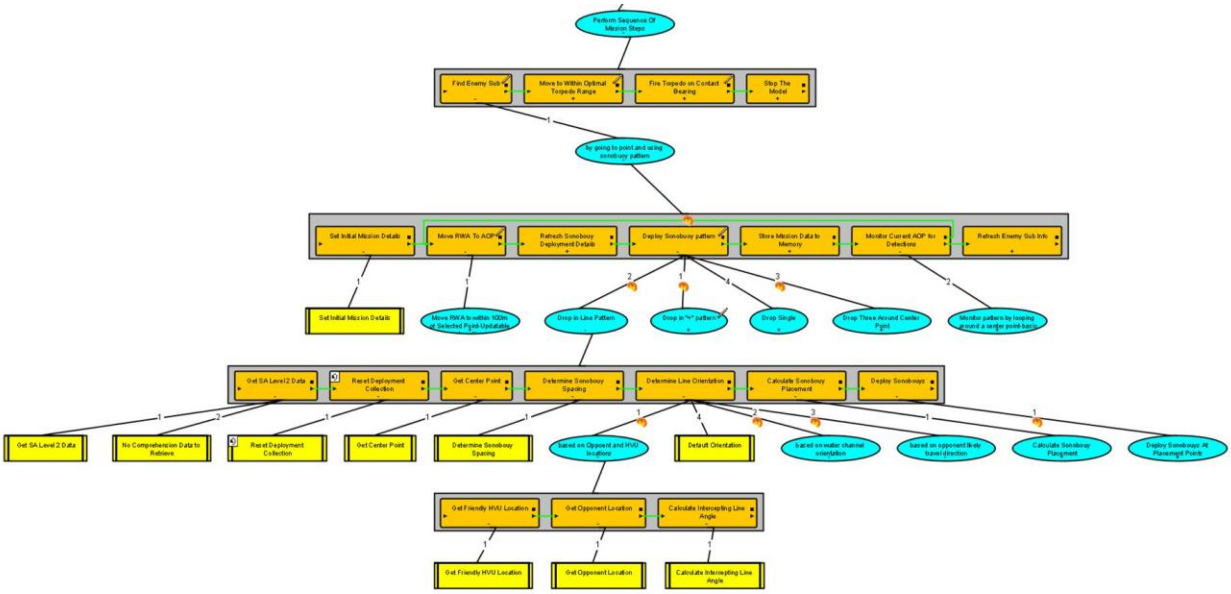

Figure 4: Example Mission Hierarchy

This mission is constructed from several basic level actions that achieve the mission objectives: Find Enemy Sub, Move to Optimal Torpedo Range, and Fire Torpedo. The second two BLAs are fairly straight forward, and perform the task of moving to within striking distance of and deploying a weapon against the enemy submarine. The more interesting BLA is the Find Enemy Sub BLA which is broken down into several major steps that allow the BLA to execute in a flexible manner depending on the situation.

Initially the Find Enemy Sub BLA reports mission details to the situational awareness module (SAM) to customize how the SAM will process and report the situation. The main details to report are the type of contact (opponent) that this BLA is interested in and the type of mission this BLA is performing. This allows the SAM to determine an appropriate area of probability (AOP) to start the search, taking into account likely opponent tactics.

With an initial AOP determined by the SAM, the BLA can proceed to the AOP and deploy a pattern of sonobuoys. This sonobuoy deployment is dependent on data processed by the SAM in that the type (passive or active) and pattern are both determined by the current situation. The factors that determine these deployment details include type of mission, level of confidence in any current contacts, probable enemy tactics, and EMCON state. An example sonobuoy deployment method is the placement of a defensive line through which the enemy will have difficulty passing. This method will determine the spacing between sonobuoys based on effective ranges of the sonobuoys to be deployed, and will calculate the orientation of the line of sonobuoys based on the mission and current situation. In this case, the mission is to defend a friendly HVU against a suspected enemy sub and the SH-60 has not been authorized to deploy active sonobuoys. The spacing of sonobuoys will be determined by the effective range of passive DIFAR sonobuoys, and the orientation will be determine by the location of the HVU and suspected sub location. The sonobuoys will be deployed in a line that cuts off the sub from the HVU. The deployed sonobuoys are reported to the SAM and this method concludes.

Having deployed a pattern of sonobuoys, the SH-60 proceeds to monitor that pattern for possible enemy sub contacts. This involves simply hovering or circling in the general vicinity of the sonobuoys and maintaining awareness of the contacts being reported by those sonobuoys. These contacts are consolidated in the SAM to combine all known data about any given contact into one contact which can then be used to determine validity of the contact and maturity of the contact details. The SH-60 will continue to monitor the pattern until it is determined that either a valid contact has been reported or that the confidence in the current AOP is too low at which point the SAM will analyze the situation and report a new AOP for the SH-60 to evaluate. If given a new AOP that is outside of the detection range of the

current sonobuoy pattern the SH-60 will repeat this process of deploying and monitoring sonobuoys as specified.

This process repeats until a valid contact is found, the SH-60 is told to abort the mission, or it runs low on fuel and needs to return to base. This basic level action is considered to be successfully completed if a valid enemy submarine contact is acquired. At this point the "Move To Optimal Torpedo Range" BLA will put the SH-60 into optimal firing position, and the Fire Torpedo BLA will deploy an MK-54 torpedo against the enemy.

Behavior Development Console for SMEs

At the heart of the behavior modeling architecture we've developed here lies the KCE Framework, which provides a base system for developing modeling environments tailored to specific domains. Figure 1 presents some examples. The DM Knowledge Service Modeler represents a general purpose knowledge capture tool, suitable for capture of process and structure knowledge in any domain. The ASW Behavior Console offers a similar environment, custom tailored for development of behaviors used in anti-submarine warfare (ASW) training scenarios. Other consoles could be developed for any variety of specific domains.

All of these consoles share the fundamental representation of processes as task decomposition hierarchies built with the TMK Language. Thus, models developed with one console are able to be viewed using other consoles. The primary reason for development of a specific console is that the interface and user experience can be modified to suit a specific set of users attempting to develop a specific set of models.

In the case of the ASW Behavior Console, the interface has been tailored to specifically assist the user in the creation of autonomous entities for deployment in ASW training exercises. This customization takes the form of various additions to the user interface to provide the user with guidance in creation of new behavior models, as well as removal of interface elements that are not needed by that particular domain.

One particular example of this customization is presented in Figure 5, which shows a wizard-like interface called an *advisor*. This particular advisor is displayed by the system when the user begins creation of a new RWA behavior. This advisor itself represents knowledge and expertise regarding how to construct executable behavior models within a simulation, which are areas that the target audience (a training instructor, or an ASW expert) may not be familiar with. By encapsulating this general knowledge of behavior modeling in the form of an advisor, we are enable the SME to turn their expertise regarding operations and their specific training requirements into fully functional entity behaviors for deployment in a simulated environment.

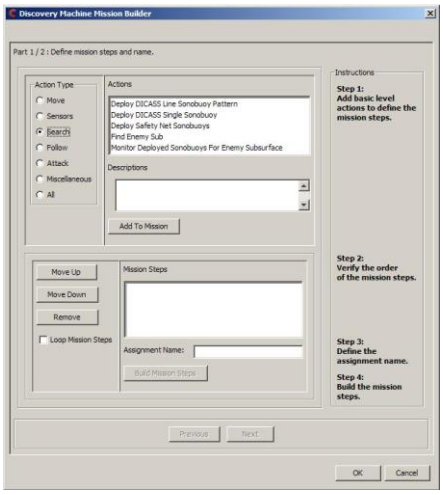

Figure 5: RWA Mission Builder Advisor

Basic Level Actions (BLAs)

This advisor guides the user in creation of behavior models comprised of modular sub-hierarchies called Basic Level Actions (BLAs). These BLAs encapsulate independently operational modules containing human expertise about how to perform a specific operational task, such as *Search for Blue Surface Vessel* or *Deploy Sonobuoy Pattern X*.

Development of a set of these BLAs followed a formalized knowledge capture methodology which was designed to extract domain specific knowledge from SMEs. Once captured, these BLAs form building blocks that any user can plug together to construct complex missions for autonomous entities, without having to delve into the complex nuances of the underlying operations that take place during execution. This enables a user such as a training instructor to construct intelligent behaviors for an RWA, even though they may not be an expert in the specifics of those types of operations.

Additionally, the user is able to construct new BLAs which comprise combinations of existing BLAs through another advisor, which will automatically identify parameters which are required or mapped throughout the novel BLA's sub-hierarchy, exposing the appropriate

parameters on the top level of the BLA for use within larger mission hierarchies.

BLAs are stored in a library which is made available to the user in the console, indexed by category (i.e. Movement, Sensors, Search, etc.) with text describing the functionality represented by the BLAs hierarchy. These are available for manual injection into mission hierarchies, as well as being available through the mission builder advisor.

BLA Development Methodology

Development of BLAs follows a formal methodology designed to facilitate introspection and articulation of subject matter expertise. This methodology for knowledge capture allows the design process to proceed from highly abstract levels of representation to concrete operational levels. The methodology consists of multiple phases. The initial phases consist of a structured interview process with the SME. The structure within the interview encourages the SME to think about problem solving within the BLA from many points of view. This helps to uncover the tacit knowledge that makes the SME expert in their particular field. The later phases of the methodology are integrated with the TMK modeling software such that BLAs can be constructed rapidly in a manner which is both transparent and executable. The BLAs are easy to understand since the TMK representation is visual and its inherent structure encourages each section of the hierarchy to be autonomous and understandable within itself. In the final phases of the methodology, the individual BLAs are tested to make sure that they perform the specified goals identified in the earlier conceptualize phase. Additionally, the library of BLAs is compared against the requirements captured in the initial phase to make sure that the library achieves coverage of the overall project's goals.

ENABLING SITUATIONAL AWARENESS

Our research effort has developed an implementation of Mica Endsley's model of situation awareness (Endsley, 1995) as shown in Figure 6. This implementation enables constructive entities to process low level sensory data into higher level abstract understanding of their environment for the purpose of enabling SMEs to define decision making at a level they are comfortable with. Our approach allows the experts to define complex, nuanced behavior that depends upon advanced situational awareness, without having to become entangled with the complexities of the cognitive processing that takes place.

Levels of Information Used in Situation Awareness

SA data is represented as two separate pieces, corresponding to the Level 1 and Level 2 data in Endsley's model. These two data structures represent perceived data and comprehended data, respectively.

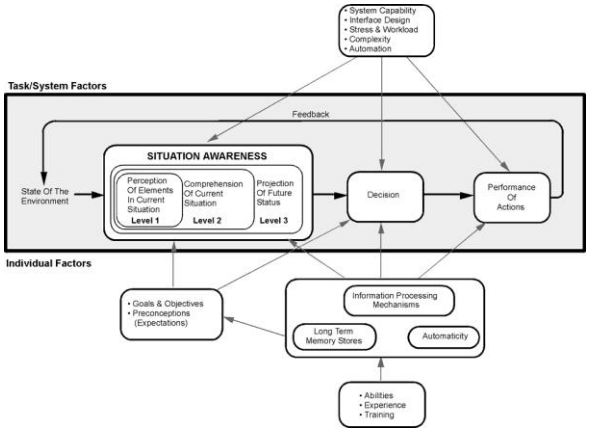

Figure 6: Mica Endsley's Model of Situation Awareness

Level 1 data consists of raw sensory perceptions of the entity within the simulation that requires no inference. Examples of this data might include things such as the state of the entity itself, or contacts that show up on that entity's sensors. This data provides a basis upon which higher level understanding of the environment can be derived.

Level 2 data consists of higher level data inferred from the Level 1 data, representing abstract mental models of the simulated environment. Examples of this data might include knowledge about a particular entity that exists in the world, based upon all available sensory data. This data provides the behavior model with an understanding of the simulated environment that can be used in the decision making of BLAs while separating out the complex cognitive modeling that takes place in processing the raw data into useful concepts.

In addition to the Level 1 and 2, projection of future states (Level 3 data in Endsley's model) as well as the resulting decision making and performance of actions within the simulation are represented implicitly within the BLAs comprising the TMK hierarchy representing the mission execution.

Processing Information

Processing of Level 1 perceived data into comprehended Level 2 data takes place within the Situation Analysis Module (SAM), executing in parallel alongside the primary mission. The SAM consists of a series of subtasks repeatedly executed in series throughout the lifetime of the behavior. Each individual task processes some piece of SA data. For instance, one of the tasks within the SAM processing loop analyzes all of the raw contact data received through the simulate RWA's sensors, as well as the contact data received by sonobuoys deployed by the aircraft throughout its mission. This data is then combined in order to develop a mental model of the entities which are generating the sensor contacts. An example of this would be merging low level data from two passive sonobuoys, which may only be providing a bearing to a target, into a single representation of an entity at a particular location derived through triangulation.

Creating Behaviors that Leverage High Level Knowledge

Data processed by the SAM is made available to the BLAs used in the construction of the behavior's mission. This data exists in a form which allows SMEs to define decisions within the behavior model using concepts they are familiar with, without having to concern themselves with the low level details involved in processing raw sensory inputs into abstract mental models.

For instance, instead of having to manually deal with a memory structure detailing past actions taken by another entity in order to determine what their goal is, the mental model representing the other entity is stored in the SA data and contains a representation of its likely mission (e.g. find and destroy the highest value unit in a group). This enables mission construction by the SME to take place on a level where they are able to build decisions around concepts that they are familiar with, without having to deal with the intricacies of complex data structures or processing. They are able to define behaviors such as, "If the enemy is performing an offensive action, place yourself between his suspected location and his likely target and perform defensive maneuvers."

In addition to enabling SMEs to leverage complex SA processing techniques within their behavior specifications, this approach has the added benefit of modularizing development of the BLAs. As the processes used to derive abstract SA data are improved, BLAs which have been previously constructed enjoy seamlessly improved performance. Multiple methods of SA processing can be employed to represent various levels of competence on the part of the autonomous entity, even allowing for human-in-the-loop injection of observations into the processing at run time.

DEPLOYMENT AND EXECUTION

Once knowledge is captured in the form of BLAs, and deployed as behaviors within the simulation, we have developed a graphical viewer for displaying the runtime execution of the behaviors' models.

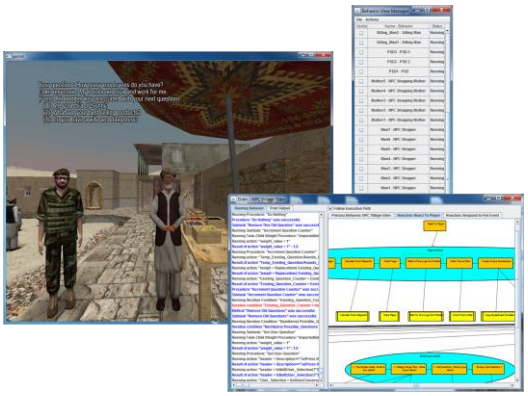

Figure 7: IWVT Shown with Behavior Manager and Viewer

Figure 7 shows the Irregular Warfare Virtual Trainer (left) running alongside the behavior manager (top right) and a particular behavior (bottom right). At runtime, the behavior manager displays all of the behaviors being run within the simulation, allowing the user to select a model for viewing. When selected, the model representing that entity's behavior model is displayed, providing the user with information about the behavior as it is being run. This view assists in debugging a behavior model by providing detailed information about all aspects of the execution, including parameter values and decisions made by the entity.

This ability to graphically view the execution of the behavior for a particular entity at runtime is crucial in preserving the transparency of the behavior for the SME. Experiences at fleet synthetic training exercises showed that JSAF operators often encountered issues when attempting to use traditionally developed tasks within the simulation. Documentation for specific tasks was often out of date, and existed only as an abstract description of the intended behavior. When the expected behavior did not match with the execution at run-time, this created frustration for the operators. In some cases, they simply chose to never use those beha-

viors again, since they were unable to understand what the behavior was doing or why.

By exposing the decision making of the behaviors, we allow operators to more easily map the observed actions taken by entities within the simulation. This allows them to not only understand why a particular behavior results in particular actions, but also enables them to directly modify those behaviors to suit their specific needs. Instead of dealing with changes to high level requirements, operators can see exactly where in a model changes need to be made to achieve the desired behavior.

Through use of a graphical view of the behavior, we unify elements of documentation with the execution. This eliminates the possibility of documentation falling out of sync with a deployed model, since any changes to the behavior model will result in a change to the graphical representation, which can be observed at runtime.

ACKNOWLEDGEMENTS

The efforts described in this paper have been sponsored by NAVAIR-TSD, Joint Forces Command, and the Office of Naval Research.

REFERENCES

Anderson, J.R. (1982) Acquisition of Cognitive Skill, *Psychological Review, 89*, 369-406.

Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C., & Quin, Y. (2004). An integrated theory of the mind. *Psychological Review, 111*, 1036-1060.

Archer, S. G. & Adkins, R. (1999). *Improved Performance Research Integration Tool (IMPRINT) Analysis Guide.* Army Research Laboratory Technical Report, Aberdeen Proving Ground, MD.

Carbonell, Jaime (1983) Learning by Analogy: Formulating and Generalizing Plans from Past Experience, In *Machine Learning*, Michalski, Carbonell, Mitchell (Eds), Tioga Press.

Chandrasekaren, B. (1986) Generic Tasks in Knowledge-Based Reasoning: High-Level Building Blocks for Expert System Design, *IEEE Expert*. Fall, 1986.

Endsley, M.R. (1995). Toward a theory of situation awareness in dynamic systems. *Human Factors*, *37(1)*, 32-64.

Garcia, C.J. & Griffith, T.W. (2005) A Composable Behavior Modeling System for Rapidly Constructing Human Behaviors, *Interservice/Industry Training, Simulation, and Education Conference (I/ITSEC) 2005*.

Gray, W. D., John, B. E., & Atwood, M. E. (1993) Project Ernestine: Validating a GOMS analysis for predicting and explaining real-world task performance. *Human-Computer Interaction, 8,* pp. 237-309.

John, B. E., Prevas, K., Salvucci, D. D., Koedinger, K. (2004) Predictive human performance modeling made easy. *Proceedings of CHI 2004* (Vienna, Austria, April 2004) ACM New York.

Kolodner, J. (1993) *Case Based Reasoning*, Morgan Kaufman, San Mateo, CA.

Lebiere, C., Archer, R., Warwick, W., & Schunk, D. (2005). Integrating modeling and simulation into a general-purpose tool. *Proceedings of the 11th International Conference on Human-Computer Interaction*. July 22-27, Las Vegas, NV.

Newell, A. (1990) *Unified Theories of Cognition*, Harvard University Press.

Royce, W. (1970). "Managing the Development of Large Software Systems". *Proceedings of IEEE WESCON, 26(August)*, 1-9

Discovery Machine, Inc. Patents Awarded:
US Patent 7,257,455 B1 – System and Method for Collecting and Representing Knowledge

Discovery Machine, Inc. Patents Pending:
US Application 10/326,386 – Software and Methods for Task Method Hierarchies
US Application 10/969,617 – Computer Interchange of Knowledge Hierarchies