Enhancing Behavioral Fidelity Within Distributed Virtual Environments

Sheila B. Banks, Eugene Santos Jr., and Martin R. Stytz
Artificial Intelligence Laboratory
Virtual Environments Laboratory
Department of Electrical and Computer Engineering
Air Force Institute of Technology
Wright-Patterson AFB, OH 45433
sbanks@afit.af.mil, mstytz@afit.af.mil, eugene@eng2.uconn.edu

Abstract

For a computer-generated force (CGF) application to be useful in training environments, it must exhibit complex, realistic behavior within the battlespace. To achieve this level of fidelity, it must operate at multiple skill levels and exhibit competency at assigned missions. CGF applications must also have adaptable decisions mechanisms and behaviors even when operating under uncertainty and the application must learn from past experience. Furthermore, simply correct performance of individual entity behaviors is not sufficient. Issues related to complex inter-entity behavioral interactions, such as the need to maintain formation and share information, must also be considered.

To achieve these necessary capabilities, an extensible software architecture, an expandable knowledge base, and an adaptable decision making mechanism are required. Our labs have addressed these issues in the context of the Automated Wingman (AW) project. The AW is based on fuzzy logic, the Common Object DataBase (CODB) software architecture, and a hierarchical knowledge structure. Decision making is founded on multi-layered, fuzzy logic controlled situational analyses combined with adversarial game tree techniques.

1. Introduction

Computer generated forces (CGFs) [5] are software agents that are computer representations of military forces that model human behavior and automatically execute a finite set of actions in response to actions and activities in their environment. CGFs of various complexity have been created for many different platforms [9] and CGFs can serve to augment friendly forces or populate enemy formations. In either role, they improve the fidelity of a distributed virtual environment (DVE) by increasing the number of entities participating in a simulation.

The need for real-time performance combined with realism for training and analysis drives a number of requirements for CGFs. CGF requirements include the need for modifiability, attainment of high fidelity human behavior representations, development of adaptable decision mechanisms and behaviors, and automated incorporation of past reasoning into the decision process. These high level requirements drive additional requirements such as multiple skill levels for classes of entities, graceful degradation of reasoning capability under system stress, an easily expandable modular knowledge structure, and adaptive mission planning. For example, the capability for multiple skill levels would provide pilot behaviors for aircraft entities at different skill levels, such as rookie, expert, and ace levels of pilots within the same battlespace. Additionally, the entity should have a complete set of behaviors for the type of missions it must perform, but not all behaviors for these missions need to be crafted to the same level of fidelity and quality. The CGF should be able to respond acceptably to unforeseen circumstances and deal with uncertain information. In addition to the individual entity requirements, issues related to complex inter-entity behavioral interactions must also be considered. The CGF should exhibit complex, realistic behavior patterns within the battlespace and be able to adaptively change its mission parameters during the course of a mission in response to events. Finally, these capabilities should be embedded in an extensible, explicable software architecture that has well defined locations for reasoning and knowledge storage. Current CGF applications lack the ability to achieve all of these requirements within a single, integrated system.

In this paper, we describe our progress toward achieving a CGF entity that satisfies the requirements presented above. Our CGF architecture is applicable to the development of *any* computer generated force to be operated in complex distributed virtual environments.

Our CGF application is presented by first introducing, in Section Two, the background topics that are relevant to the development of CGFs. Section Three identifies our CGF architecture and the design of an actual CGF, the Automated Wingman (AW). Section Four describes the decision making component of the Automated Wingman within our CGF architecture and Section Five concludes the paper with current research results and suggestions for future improvements.

2. Background

In this section we discuss Distributed Interactive Simulation (DIS), the Common Object DataBase (CODB) architecture, and current aircraft CGF projects in relation to the AW. Presentation of this material is motivated by the need to understand the CGF operational environment, the rationale for the CGF architecture we propose, and the decision-making methodology we employ.

2.1 Distributed Interactive Simulation

Our CGF, the Automated Wingman, operates in a DVE [5] that employs the distributed interactive simulation (DIS) suite of protocols, IEEE 1278-1993 [3]. These protocols enable communication between individual simulator computer systems at distributed locations. DIS achieves an interactive representation of a virtual environment by the interconnection of the distributed hosts. Each host determines what is actually perceived of its entities and only communicates changes in the state of entities for which it is responsible. The DIS approach to DVEs requires participating sites to meet several requirements. These requirements consist of the following: (1) autonomous operation, (2) object and event based simulation, (3) state change information broadcast, and (4) dead reckoning capability. In the DIS protocols, the DVE contains actors and entities that interact through the asynchronous broadcast of event and state information. Stytz presents additional information concerning DIS and DVEs [5].

2.2 Common Object DataBase (CODB)

The Automated Wingman software architecture is founded on the Common Object DataBase (CODB) architecture [6]. The Common Object DataBase is a data-handling architecture that uses structured classes, data containers, and a central runtime data repository to manage and transmit data between application objects. This architecture reduces the coupling in a simulation by reducing the amount of information that a system component object class must maintain about other system component classes. As a result, an application component object must only access the container in the CODB where the information it needs resides.

2.3 Current CGF background and projects

The major structured components of an aircraft CGF include the following: vehicle dynamics, artificial intelligence (AI) reasoning, behavior modeling, and software architecture. CGF vehicle dynamics are important because the CGF should move through the virtual environment accurately whether it is human or computer-controlled. The vehicle dynamics for CGFs should not allow a human to identify it as a CGF by virtue of either exceptionally good or poor dynamic behavior.

The artificial intelligence reasoning component insures that the CGF pursues its goals, responds in a proper, human-like manner based upon its knowledge base, keeps the performance of the CGF within human and sensor limits, develops plans based upon its knowledge base, and manages other tasks. Fielded systems, like TAC-AIR SOAR [9] and ModSAF [1, 2] address these problems at different levels of fidelity. TAC-AIR SOAR, which builds upon the Soar architecture [4] for general intelligence and reasoning, is the most successful of the current aircraft CGFs. However, TAC-AIR SOAR does not accommodate uncertainty in its decision making process.

The central task in knowledge base construction is human behavior modeling. Human behavior modeling is the task of making the behavior and reactions of a CGF seem realistic by developing models that yield a reasonable analog of the output of the human decision-making process. Human behavior modeling requires the acquisition of domain-specific knowledge about human mental models and information brought to the decision-making process, as well as the key factors in the decision process. The human behavior modeling subcomponent of CGFs might normally be considered as part of the artificial intelligence component. However, separating it from the AI reasoning component serves to highlight its importance and the need to make behavior modeling design decisions independently of behavior reasoning mechanism design decisions.

A flexible CGF software architecture ensures that current CGF development efforts are extensible to future CGF requirements. The ability to modify the implemented CGF to include additional behavioral requirements directly depend upon software architecture flexibility.

3. CGF architecture

Our motivation for the development of a general CGF architecture for the AW was to provide a basis for the design of broad classes of CGFs. While each class of CGF has its own unique characteristics and performance requirements, there are many factors common to all classes and these can be successfully abstracted and reused across all classes of CGFs. Our goal is to provide a general architecture for CGFs, which naturally accounts for "variety" in a given type of CGF. Our architecture is based on highly modular components wherein interdependencies are well-defined and minimized and that allows us to pursue an evolutionary and exploratory approach to knowledge engineering.

3.1 CGF architectural requirements

The architecture we developed is based on several principles [7]. The first is that future architectures should focus on reducing programmer costs, even at the possible expense of marginal runtime processing inefficiencies. The second principle is that the development cycle for a CGF, as in most research and development projects, must include a series of revisions to the requirements. These changes range from changing data formats to introduction of new behaviors into the system. A third principle is that the push to attain improved performance and the strain of meeting delivery deadlines increases the entropy of any design, until the original design concept becomes blurred. The most obvious results of entropy are the use of global variables, global functions, and the disappearance of private data items. A fourth principle is that the components of the airframe model (aerodynamics model, avionics systems, and weapons packages) should be rapidly modifiable. Therefore, these components should be realized as separate objects that have a clean, robust interface to the remainder of the system. The final principle is that expanding system requirements will cause the knowledge base and reasoning system to be modified and adapted to new requirements throughout the life of the project and in the subsequently fielded system. As a result, the knowledge and reasoning components should be structured so that the knowledge base and reasoning system are clearly separated from the remainder of the system.

3.2 CGF component architecture

The essence of the architecture problem domain is the following: given that a need exists to build a CGF, what design architecture and methodology can be applied to take the CGF from concept to implementation, regardless of the type of CGF? Figure 2 presents the baseline architecture and key CGF components that we propose to answer this question. A CGF is essentially comprised of two types of components: a Physical Dynamics Component (PDC) and an Active Decisions Component (ADC). The PDC is made up of the components necessary to model the CGF's physical makeup, such as entity propagation models, sensor models, weapons models, and defensive elements models. The ADC is composed of the components that use the information from the PDC and the DVE to make decisions, and is broken down into three subcomponents: a Strategic Decision Engine (SDE), a Tactical Decision Engine (TDE), and a Critical Decision Engine (CDE). Each of the decision engines operates at a different level of the decision making process and are presented further in Section 3.4.

Although not considered a component equal to the ADC or PDC, the CGF architecture has another important structure. This is the CGF Router, which is the interface between the Distributed Virtual Environment (DVE) and the ADC and PDC.

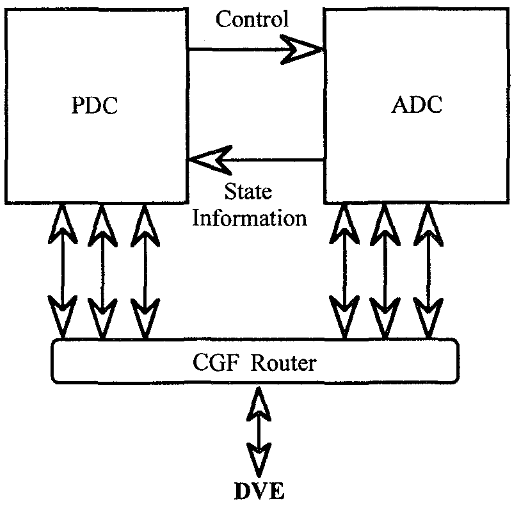

Figure 2. Generic CGF architecture: PDC, ADC, and CGF Router

3.3 Automated Wingman (AW) component architecture

Our AW system architecture (see Figure 3) used the principles in Section 3.1 to refine the CGF architectural definition in Figure 2. Within the architecture we use containers, which are data structures for moving large amounts of structured data between system components, to manage and control inter-component communication. The main CGF components are specified as objects. These objects are the Operator Skills Component (OSC), the Active Decisions Component (ADC), the Physical Dynamics Component (PDC), the Common Object DataBase (CODB), the World State Manager (WSM), and the Environment Database. Each of these

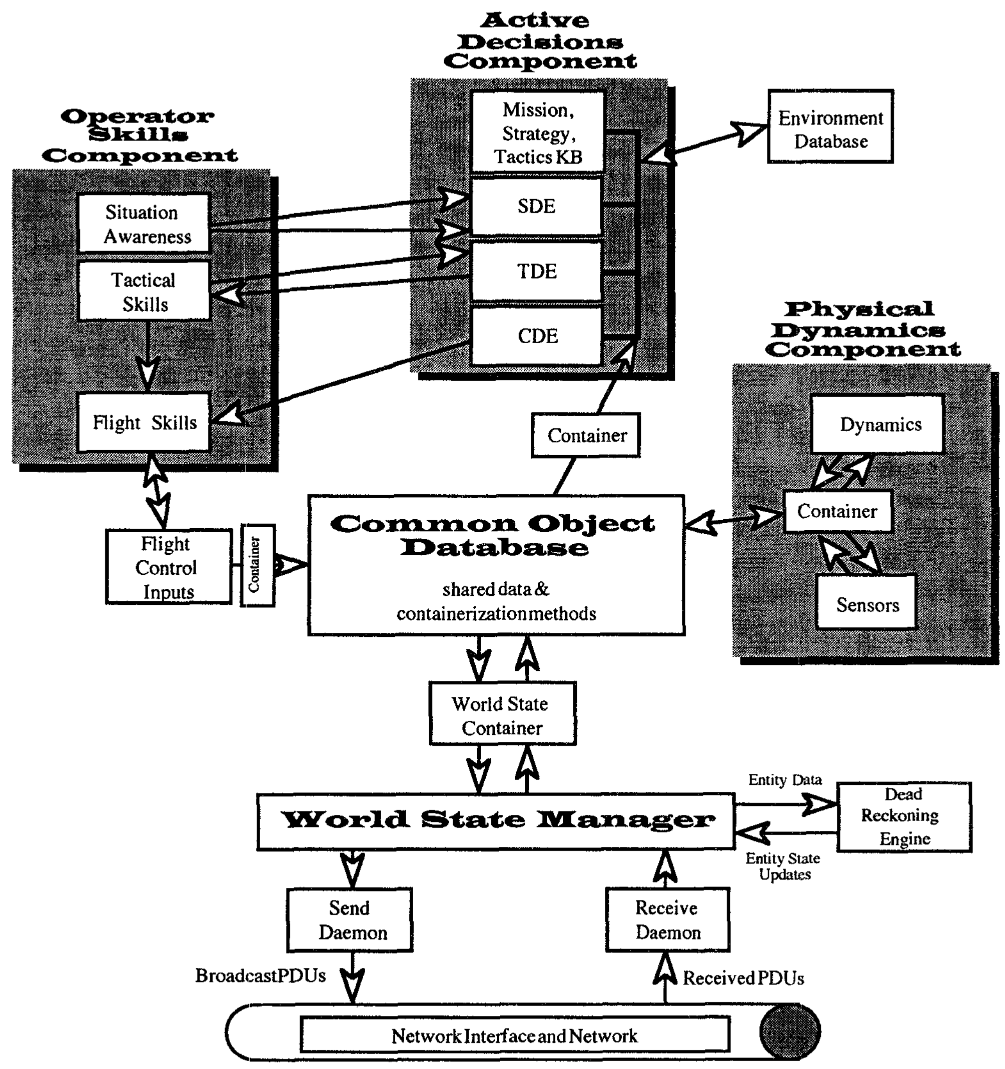

Figure 3. AW system architecture incorporating the Common Object DataBase

objects are, in turn, hierarchically defined as a set of objects that use containers to communicate. The CODB holds all public data and all system components may access the CODB for data from other system components. The World State Manager maintains a complete description of the state of distributed virtual environment as communicated using DIS-formatted PDUs.

3.4 CGF software and decision architecture

The AW architecture consists of three components (Figure 3): an Operator Skills Component (OSC), an Active Decisions Component (ADC), and a Physical Dynamics Component (PDC). The PDC encapsulates all the physical attributes and properties of the CGF. In the AW, this component includes the aerodynamics model, entity-specific properties, aircraft capabilities, weapons load, sensors, damage assessment, and physical status. In addition, the PDC computes physical state changes such as computing the current AW position in the virtual environment. The OSC consists of those portions of the CGF that vary between individual entities within a type and class, and models the skills and ability of the operator of the entity. For an aircraft entity, the components of the OSC include the simulated pilot's ability to maintain situation awareness and to execute tactical and flight skills. The ADC encompasses the intelligent decision making processes and the knowledge necessary to properly drive them, which includes the overall mission, goals and objectives, plan generation, reaction time, and crisis management ability. The ADC must accomplish many of its activities in real-time.

We chose to separate the components of the CGF architecture in this manner to insure that modifications are isolated and will not propagate throughout the system. The PDC only holds the basic entity maneuver information and is completely unaware of the status of the other system components. Likewise, the ADC is solely responsible for decision making and only knows about the physical component's status based upon the data communicated within the system software architecture. The OSC is more closely tied to the ADC than the PDC because the ADC is responsible for computing control outputs for the entity based upon the modeled operator's skills. The OSC describes the operator's ability for the decision making component so that the decision can be appropriately constrained by the pilot's abilities. The division of capabilities between these basic components lessens the system level impact of any changes in the PDC, OSC, or ADC.

The OSC and PDC contain all the information and status required to portray an aircraft model and its operator's ability. The PDC encapsulates the entity state information and the OSC contains a representation for all the operator skill variables. The key aspect of these two components is that these subsystems are completely parameterizable, and hence rapidly reconfigured and reused. We isolate entity control skills in the OSC because this separates the ability to parameterize the operator's capabilities from the decision making mechanisms. This parameterization allows any number of CGFs of a given type to be generated using a given ADC but yet each entity can have its own unique set of operator skills. The OSC models the operator's skills as a hierarchy of capabilities and allows us to compose more complex skills from elementary skills and to compose the higher level skills using a weighting of the appropriate elementary skills. The PDC and OSC do not, however, perform decision making based upon the information they store. The decision making task is solely the responsibility of the ADC.

The ADC contains a fuzzy planner, based upon fuzzy logic [8, 10], that allows certain subgoals to remain unsatisfied but yet allow the supergoal to be satisfied. This decision making flexibility permits a wide variety of possible behaviors and provides decision-making elasticity, allowing the CGF to achieve its mission in the face of uncertainty. Also, the fuzzy approach provides a method for optimization when various subgoals are applicable but only one is desired. This use of fuzzy logic adds another behavioral distinction that can be exploited to create a diverse mix of entities. The ADC is composed of the strategic decision engine (SDE), the tactical decision engine (TDE), and the critical decision engine (CDE).

The SDE is concerned with high level functions such as understanding and implementing mission level goals, communicating with other players, interpreting the surrounding environment, and revising mission goals and subgoals. The SDE handles strategic matters related to accomplishing mission goals by continuously re-evaluating the completion status of mission objectives and re-planning in a deliberative fashion to achieve the objectives. To execute its plans, the SDE then requests the TDE to carry out the near term (tactical) objectives.

The TDE's role is to make decisions about the moment to moment operations of the entity. This entails receiving specific taskings from the SDE (subgoals), assessing DVE state, and responding to these situations according to the knowledge base of the particular entity. These responses include activities such as determining which maneuvers to make in a given situation, determining when to employ ordnance, and when to implement other application specific elements such as electromagnetic counter measures.

The CDE supplies the survival instinct for the virtual entity and takes over control of the entity in emergency situations where termination of the entity is imminent. The CDE is a purely reactive reasoning system that deals with critical situations the AW might encounter. In operation, the CDE monitors the world state until a critical situation is detected. The CDE then assumes control of the AW until the crisis has passed.

3.5 CGF knowledge hierarchy

The essence of our approach to defining a CGF knowledge hierarchy lies in applying the concept of *separation of concerns* to the domain of CGFs. As defined in the practice of software engineering, applying separation of concerns in a system design means separating the "how" from the "what" in the system. We believe that this technique is useful in creating modular CGFs because it assists in separating an entity's decision making ability from its physical ability. The separation of concerns is also a useful technique for decomposing the entity's decision making process into manageable subcomponents.

We used the concept of separation of concerns to aid in defining and refining the roles of the decision engines. To accomplish this, semantic Areas Of Concern (AOC) were used. An AOC is a question that must be answered to understand the role of a decision engine. The purpose of the AOC is twofold: first, they help the knowledge engineer identify the kinds of knowledge and behaviors to be modeled at the different levels of decision making, and second, they help the software engineer organize the various decision making processes into encapsulated interchangeable modules.

The AOC were established as a result of modeling the considerations that confront a present day war fighter. For example, perhaps the single most fundamental consideration is the mission. From this flows a series of follow-on questions, such as "How do I perform my mission?", and "What is my current task within the

Table 1. Key questions for AOC determination

Decision Engine Component	Key Definitional Questions
SDE	• What's my mission? • How do I perform my mission? • Am I able to accomplish my mission? • What's my current task within the mission? • How do I communicate with other entities and actors? • How do I perceive the outside world? • How do I deal with a lack of appropriate information? • What are the target types that I'm designed for? • What are my responsibilities?
TDE	• What kinds of maneuvers am I capable of? • What kinds of weapons do I know how to use? • How do I choose which maneuver to apply in a given situation? • How do I choose which type of weapon to employ in a given situation? • How do I know when to employ ordnance? • How do I know that the weapon was effective? • How am I aware of my environment? • How does the local DVE state affect my decision making? • What is my tasking within the mission?
CDE	• Am I in any immediate danger? • If I am in danger, what maneuvers do I know how to do to escape the danger? • How would I decide which maneuver to make? • What kinds of countermeasures do I know how to use? • How do I decide when to use a countermeasure?

mission"? To be most useful, the AOC should be as domain independent as possible.

To make the AOC as simple and direct as possible, they are written in a first person perspective from the CGF's point of view. They are designed so that by answering these questions in the context of a specific type of CGF within a specified domain, the bulk of the work for identifying the CGFs behaviors will be accomplished. The key AOC are shown in Table 1.

4. CGF comparative decision mechanism

In addition to the primary decision engines that utilize fuzzy logic as the basis for decision making, the ADC makes use of a comparative decision process to perform "what-if" comparisons. One of the requirements for the Automated Wingman was to be able to choose the "best" maneuver to employ in a real-time engagement. This capability was achieved by modifying a two player, time stepped game tree [11] into a two player, asynchronous game tree (see Figure 4.) This approach allows the CGF to analyze the situation by determining opponent versus AW maneuver combinations, and then scores the resulting, relative orientations of the two aircraft according to what the algorithm expects to happen during the course of a fuzzily determined time period. The algorithm also operates within dynamically determined time constraints based on aircraft positions and velocities as well as expected weapon ranges and velocities. The algorithm analyzes, in a breadth first search, as many maneuver combination as possible within the given time constraints and presents the best possible choice for implementation. The process of selecting the maneuver combination is a discrete choice, not a fuzzy one. We adapted the game tree concept to a dynamic environment so that the CGF could compare and evaluate potential maneuvering decisions with regards to concurrent adversarial maneuvers over time. As a result, the complete ADC decision process requires the integration of multiple fuzzy decisions engines with a non-fuzzy comparative game tree. Our game tree approach is describedbelow.

Game trees are useful a decision mechanism when possible moves between opponents are known and resulting positions can be enumerated. The resulting positions can then be weighed, and decisions made based by selecting the path through the tree that results in the most advantageous weighting. Game trees are not naturally applied to situations where resulting positions between players are anything other than discrete. However, if sets of otherwise non-discrete possible player positions are described in discrete ways, then it is possible to use a game tree to make the best possible

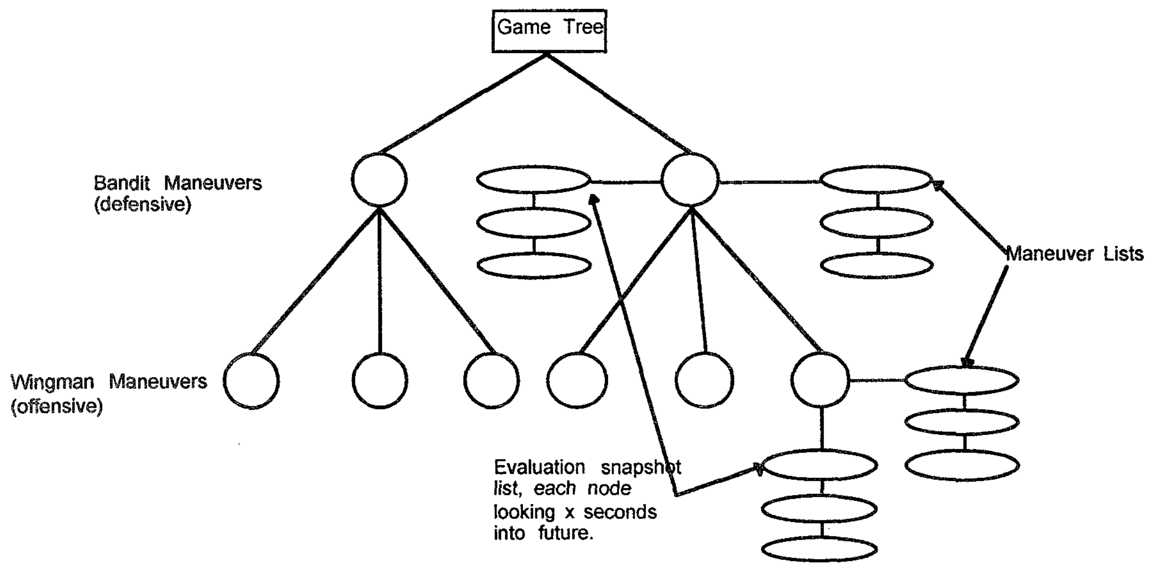
Game Tree
Bandit Maneuvers (defensive)
Wingman Maneuvers (offensive)
Maneuver Lists
Evaluation snapshot list, each node looking x seconds into future.

Figure 4. Asynchronous game tree example

choice. We used this technique for the Automated Wingman.

A CGF-type independent knowledge representation for the game tree was developed to represent possible player moves. This representation allows the knowledge to be separated from the algorithm used to act on the knowledge. As a result, we can use the same game tree algorithm for any CGF regardless of its CGF type. We represent the moves, or maneuvers, as directional graphs within a three-dimensional space. We represent the maneuvers in this space by taking the fundamental shape of the maneuver and then rotating and scaling the maneuver using standard formulas, which allows us to achieve any desired permutation of the basic maneuver. We specify the maneuver using x, y, and z coordinates and the values for CGF heading, pitch, and roll. Given these six attributes, it is possible to describe the location and orientation of a CGF with regards to the coordinate system origin. By considering CGF velocity, applying a specific maneuver representation, and then rotating and scaling the maneuver based on the velocity, we can predict the location and orientation of the CGF at a future time.

Given that we used this approach to model maneuver decision making in the highly dynamic and time critical domain of combat aircraft operations, we needed a mechanism to determine the maximum elapsed time that the game tree algorithm could consume while searching for a solution. To establish the maximum time, we calculate the minimum time it would take the AW to intercept the target from the current position. This minimum intercept time is then divided by a user specified value, the result being the maximum allowable time the algorithm can use to make a decision. The user specified divisor prevents the algorithm from consuming the entire time to intercept to make a decision. By giving the user control of this value, it allows flexible scaling of the performance of the CGF's decision making. In addition to the division value, an absolute minimum timing value is also specified in case the calculated decision time is very short (less than one second), this allows the algorithm to override the calculation and use the minimum specified decision time.

The game tree is constructed by using the target's known maneuvers for a defensive approach as the first tier nodes, and then combining all the wingman's maneuvers for an offensive approach as the second tier nodes. As a result, the ply of the basic tree never exceeds two (not including the depth of the evaluation branches discussed in later). Since the algorithm was written to accept any number of maneuvers when constructing the tree, the names of the known maneuvers as well as their data files are established at program startup by an initialization file. The CGF does not accept dynamically defined maneuvers after program start.

To permit it to operate on fuzzy logic variables, the game tree algorithm uses the values of variables defined over fuzzy term sets as iterative multipliers to determine how far ahead the CGF can accurately predict both it's and the opponent's position and orientation. For our tests, the evaluations of the fuzzy values resulted in look-ahead values of one, two, or three seconds. As a result, on each breadth-wise pass through the game tree, an AW with a look-ahead value of three seconds would get to look three seconds into the future of the current maneuver combination. Hence, after two passes of the tree, an AW with a look-ahead of three would be six seconds into the maneuver, whereas an AW with a look-ahead of one would only be two seconds into the maneuver. Thus we are able to establish better foresight performance for an AW with a higher operator skill rating.

By knowing the maneuvers and the aircraft velocities, we can create "snapshots" of future player positions and orientations calculated using the look-ahead value to determine the number of passes through the game tree. These snapshots are then used to represent the discrete player positions required for evaluation of the game tree. Moving breadth-wise through the game tree, player positions are evaluated one maneuver pairing at a time, so long as there is a combination remaining and sufficient time to make the evaluation. A single evaluation involves fuzzifying the coordinate locations, and then allowing the inferencing engine to evaluate all the applicable rules. We use a standard, center of gravity defuzzification technique to derive a player positional score.

5. Future work and conclusions

The AW currently presents a believable but limited portrayal of pilot behaviors within a DVE. Future work should expand the range of behaviors available to the AW. One aspect of this effort should be the design, implementation, and testing of the parameterization of multiple skill levels. Another aspect of the skill levels development activity must be to expand the AW's knowledge base by employing techniques for learning and knowledge-base modeling to provide a capability for achieving multiple levels of CGF ability/skills. However, simply possessing the capabilities for multiple skill levels within the decision component is insufficient for our purposes. A theory for designing entities with multiple skill levels, techniques for reusing skills among multiple skill levels, techniques for degrading skills without reaccomplishing the knowledge-base design, and an approach for modeling skill levels in humans are all required. Based on our results with the AW software and knowledge base architectures, a promising area for research is the investigation of the capability of these architectures to instantiate multiple and differing classes of CGFs.

The Automated Wingman has achieved a basic capability for believable operation within military DVEs and has demonstrated that fuzzy logic in combination with an adversarial game tree approach to decision making can achieve an acceptable range of behaviors for a CGF. The AW is based upon a generic CGF knowledge base and software that permits us to scale the software components and the knowledge base in response to new requirements as well as allowing us to undertake a rapid prototyping approach to CGF development. We have demonstrated the ability to adapt a static game tree to the highly dynamic, asynchronous environment typically found with combat fighter aircraft computer generated forces. The positive results of our initial tests showed that such an asynchronous game tree could in fact perform well in such an environment when the maximum processing time per decision cycle is constrained.

References

[1] Calder, R.B., Smith, J.E., Courtemanche, A.J., Mar, J.M.F., and Ceranowicz, A.Z. "ModSAF Behavior Simulation and Control," *Proceedings of the Third Conference on Computer-Generated Forces and Behavioral Representation*, Orlando, FL, pp. 347-356, 1993.

[2] Ceranowicz, A. "ModSAF Capabilities, A Progress Report," *Proceedings of the Fourth Conference on Computer Generated Forces and Behavioral Representation*, Orlando, FL, pp. 3-8, 4 - 6 May, 1994.

[3] IEEE Std 1278-1993. IEEE Standards Coordinating Committee of the IEEE Computer Society. "IEEE Standard for Information Technology - Protocols for Distributed Interactive Simulation Applications". Technical Report IEEE Std 1278-1993, March 1993.

[4] Laird, J. E., Newell, A., and Rosenbloom, P.S. "SOAR: An Architecture for General Intelligence," *Artificial Intelligence*, Vol. 33, pp. 1-64, 1987.

[5] Stytz, M. R. "Distributed Virtual Environments," *IEEE Computer Graphics and Applications*, Vol. 16, no. 3, pp. 19 - 31, May, 1996.

[6] Stytz, M. R., Adams, T., Garcia, B., Sheasby, S. M., and Zurita, B. "Developments in Rapid Prototyping and Software Architecture for Distributed Virtual Environments," *IEEE Software*, Vol. 14, no. 5, pp. 83-92, Sep. - Oct., 1997.

[7] Stytz, M. R., Banks, S. B., and Santos, E. "Requirements for Intelligent Aircraft Entities in Distributed Environments," *Proceedings of the 18th Interservice/Industry Training Systems and Education Conference*, Orlando, Florida, publication on CD-ROM, 3 - 5 December, 1996.

[8] Schwartz, D. G. "A System for Reasoning with Imprecise Linguistic Information," *International Journal of Approximate Reasoning*, Vol. 8, pp. 463-468, 1991.

[9] Tambe, M., Johnson, W. L., Jones, R.M., Koss, F., Laird, J. E., Rosenbloom, P. S., and Schwamb, K. "Intelligent Agents for Interactive Simulation Environments," *AI Magazine*, Vol. 16, no. 1, pp. 15-40, Spring, 1995.

[10] Zadeh, L. A. "The Concept of a Linguistic Variable and its Application to Approximate Reasoning," *Information Sciences*, Vol. 8, pp. 199-249 and 301-357, 1975.

[11] Giarratano, J. and Riley, G. (1994) *Expert Systems: Principles and Programming*, PWS Kent, Boston.