MOVING ADVANCED DISTRIBUTED SIMULATION INTO OPERATIONAL TRAINING: THE DISTRIBUTED MISSION TRAINING INTEGRATED THREAT ENVIRONMENT PROJECT

Martin R. Stytz, Ph.D., Sheila B. Banks, Ph.D., Eugene Santos, Ph.D.
Virtual Environments Laboratory
Artificial Intelligence Laboratory
Air Force Institute of Technology
Wright-Patterson AFB, OH 45433
mstytz@afit.af.mil, sbanks@afit.af.mil, esantos@afit.af.mil

ABSTRACT

As the Air Force moves to the use of Distributed Mission Training for pilots, the adequacy of integrated threat systems and their ability to operate within a distributed virtual environment will be key factors in determining the success of this approach to aircrew training. For computer generated threats to be useful in training environments, they must exhibit a broad range of skills, display competency at their assigned missions, and comply with current doctrine. For cost reasons, a single computer host should be capable of inserting several threats into the environment, coordinating the threat activities with threats inserted using other systems, and of reusing scenarios developed at other host systems. Because of the rapid rate of change in Distributed Interactive Simulation and the expanding set of performance objectives for any computer generated force, the system must also be modifiable at reasonable cost and incorporate mechanisms for learning. The requirements pose an intricate set of challenges because the system must satisfy reasoning and fidelity requirements as well as performance requirements. To address these circumstances, we developed a set of general requirements for distributed mission training threat systems and used them to guide our specification of a generalized architecture for these systems. In this paper, we present a component-wise decomposition of the system and describe the structure of the major components of the distributed mission training threat system's decision mechanism.

ABOUT THE AUTHORS

Martin R. Stytz is an active duty Lieutenant Colonel in the U.S. Air Force serving as an Associate Professor of Computer Science and Engineering at the Air Force Institute of Technology. He received a Bachelor of Science degree from the U.S. Air Force Academy in 1975, a Master of Arts degree from Central Missouri State University in 1979, a Master of Science degree from the University of Michigan in 1983, and his Ph.D. in Computer Science and Engineering from the University of Michigan in 1989. He is a member of the ACM, SIGGRAPH, SIGCHI, the IEEE, the IEEE Computer Society, the IMAGE Society, AAAI, and the Society for Computer Simulation. His research interests include virtual environments, distributed interactive simulation, modeling and simulation, user-interface design, software architecture, and computer generated forces.

Sheila B. Banks is an active duty Major in the U.S. Air Force serving as an Assistant Professor of Computer Engineering at the Air Force Institute of Technology, Department of Electrical and Computer Engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH. She received a Bachelor of Science in Geology from University of Miami, Coral Gables, FL in 1984 and a Bachelor of Science in Electrical Engineering from North Carolina State University, Raleigh, NC in 1986. Also from North Carolina State University, Raleigh, NC, she received a Master of Science in Electrical and Computer Engineering in 1987 and her Doctor of Philosophy in Computer Engineering from Clemson University, Clemson, SC in 1995. Her research interests include artificial intelligence, intelligent computer generated forces, associate systems, distributed virtual environments, intelligent human computer interaction, and man-machine interfaces.

Eugene Santos Jr. is an assistant professor of computer science at the Air Force Institute of Technology. He received his B.S. in Mathematics and Computer Science (1985) and M.S. in Mathematics -- Numerical Analysis (1986) from Youngstown State University (1985) and subsequently completed a Sc.M. (1987) and Ph.D. in Computer Science -- Artificial Intelligence (1992) from Brown University. His research interests include automated reasoning, neural networks, natural language understanding, expert systems, machine learning, operations research, probabilistic reasoning, robotic planning, temporal reasoning, combinatorial optimization, numerical analysis and parallel processing. Member, IEEE, ACM, Sigma Xi and AAAI.

MOVING ADVANCED DISTRIBUTED SIMULATION INTO OPERATIONAL TRAINING: THE DISTRIBUTED MISSION TRAINING INTEGRATED THREAT ENVIRONMENT PROJECT

Martin R. Stytz, Ph.D., Sheila B. Banks, Ph.D., Eugene Santos, Ph.D.
Virtual Environments Laboratory
Artificial Intelligence Laboratory
Air Force Institute of Technology
Wright-Patterson AFB, OH 45433

INTRODUCTION

The Joint Synthetic Battlespace (JSB) proposed within the Department of Defense modeling and simulation master plan requires a distributed virtual environment (DVE) wide consistent threat environment to achieve a useful mission rehearsal, training, test and evaluation capability. To achieve this objective, all threats must appear at compatible levels of fidelity for all the entities operating in the DVE and they must interact with human-operated and computer-controlled entities in a realistic fashion. Achieving this goal is not currently possible for two reasons. First, each primary aircraft simulator training system developer has created their own threat system and made their own modeling decisions to support a specific user for a select few predetermined conditions. This traditional threat simulation approach is expensive and leads to ongoing difficulties in maintaining threat currency as intelligence updates are made, new weapons are introduced and new theaters of operation are identified. Second, the threat system interaction on a distributed network must be coordinated. The individualized nature of current threat systems precludes the possibility of introducing coordinated threats. The Distributed Mission Training Integrated Threat Environment (DMTITE) Project is developing an effective solution to these issues.

The DMTITE project is identifying the requirements for a distributed threat environment and building a demonstrator DOD High Level Architecture (HLA) compatible system that can provide realistic threats for pilots to train against. To be a suitable prototype, DMTITE must provide a distributed threat environment composed of surface threats, air threats and jamming systems. To achieve these objectives, DMTITE must have threat models, a knowledge base, and a ruleset for interaction with every appropriate entity in the DVE.

The DMTITE prototype will instantiate a variety of threats for use in distributed training scenarios. A key element of the system will be the provision of realistic behaviors for the threat systems. As a basis for this modeling, we incorporate realistic sensor models, aerodynamics models, and weapons models into DMTITE for each threat system. Decision making, using fuzzy logic, will be augmented with a scripting capability. We can readily expand the system to accommodate peer-to-peer communication and group tactics. This approach allows us to provide a range of threat skill levels for each threat modeled within the DMTITE system.

We based DMTITE on a general software architecture for computer-generated forces (CGFs) that naturally supports “variety” in performance for a given type of CGF and allows us to organize and build vastly different CGFs within the same architecture. Given the continuously changing nature of CGF requirements, an evolutionary and exploratory approach to knowledge engineering, such as our Rapid Evolutionary and Exploratory Prototyping (REEP) methodology is required. To support the REEP approach, our architecture consists of highly modular components where interdependencies are well-defined and minimized. To accommodate the instability of requirements and the accelerating change of pace in the underlying technologies, our software architecture is suitable for implementing and maintaining applications developed using a modified rapid prototyping approach. DMTITE will be operated using a graphical user interface that can be used to configure the threats that are required for a training scenario.

The next section presents a short discussion of background information for our project. Section three contains a description of the operational concept for the DMTITE system, its requirements, and the architectural implications of these requirements. Section four presents our solution to the system requirements that have been levied against DMTITE. Section five contains a summary and presents some suggestions for additional work.

BACKGROUND

This section discusses the topics of Distributed Interactive Simulation (DIS), current CGF background and projects, and the Common Object DataBase (CODB) architecture and the REEP approach to system development. These topics form the groundwork for the DMTITE architecture and its operational environment.

Distributed Virtual Environments

The most widespread use of network technology for distributed virtual environments (DVEs) relies upon the current DIS suite of standards (IEEE Standard 1278-1993) or upon the DOD High Level Architecture (HLA). DIS was designed to link distributed, autonomous hosts into a real-time distributed virtual environment via a network for exchanging the data that describe events and activities. DIS takes the concept of environmental distribution to its extreme; there is no central computer, event scheduler, clock, or conflict arbitration system. The HLA is a more comprehensive architectural approach, it describes the communication requirements, basic system requirements, and defines an object-oriented approach to defining a DVE. Stytz presents additional information concerning DIS and DVEs (7), as does Blau (1,2).

Computer Generated Forces

Computer generated forces (CGFs) that exhibit human-like behaviors are crucial to achieving large-scale DVEs. Approaches to achieving realistic CGF are described by Calder, et.al. (3), Edwards (4), Laird (5,6), and Tambe (10). CGFs are important aspects of a DVE because they enable a complex virtual environment with a large number of actors to be activated without the expense of involving large numbers of humans. The run-time challenges for a CGF lie in computing human-like behaviors and reactions to a complex dynamic environment at a human-scale rate of time. However, the computational challenge is eased somewhat because there is no need to replicate the human decision process, instead only the observable aspects of human decision making must be mimicked. However, the CGF behavior must be realistic and accurate enough so that other CGFs and human participants react to its outputs as though it were human-controlled. Advances in artificial intelligence are important to the development of realistic CGFs. The capability to construct large, complex reasoning systems and the development of large knowledge bases for use by the decision machinery combine to enable the implementation of CGFs of acceptable fidelity.

For our purposes, the major components of a CGF are the following: vehicle dynamics, behavior modeling, artificial intelligence, and software architecture. Vehicle dynamics are important because the actor should move through the virtual environment accurately whether it is human or computer-controlled. The vehicle dynamics for computer-controlled actors should never allow a human to identify it as a CGF.

Human behavior modeling addresses the task of making the behavior and reactions of a CGF realistic by developing models that yield a reasonable analog of the output of the human decision-making process. The human behavior modeling component should be modeled separately from the artificial intelligence component and it is the focus for certifying the accuracy of the performance of the CGF. Human behavior modeling requires the acquisition of domain-specific knowledge about the domain models that humans use and about the information that humans use in the decision-making process. For military virtual environment purposes, human behavior modeling involves incorporating doctrine, tactics, knowledge models, and training into the CGF.

The area of artificial intelligence addresses the problems associated with assessing and reacting to the environment based upon considerations like plans, assigned mission, the activities of other actors, the available domain knowledge, and the capabilities of the vehicle that the CGF must control. The artificial intelligence component insures that the CGF pursues its goals, responds in a proper, human-like manner based upon its knowledge base, develops plans based upon its knowledge base, and manages other tasks.

The vehicle dynamics, behavior modeling, and artificial intelligence system components are brought together within the CGF software architecture component. A flexible CGF software architecture ensures that current CGF development efforts are extensible to address future CGF requirements. The ability to modify the implemented CGF to include additional behavioral requirements is directly attributable to the software architecture's flexibility.

Common Object DataBase and Rapid Exploratory and Evolutionary Prototyping

The DMTITE architecture is based upon the Common Object DataBase (CODB) as described by Stytz (8). The Common Object DataBase is a data-handling architecture that uses classes, data containers, and a central runtime data repository to route data between the major objects in an DVE application. The CODB holds the entire current state of the DVE and all the public or exported information for each threat in operation within the DMTITE application. This architecture reduces the coupling in a simulation by reducing the amount of information that a class must maintain about other classes. To acquire public data from other DMTITE application objects, an object need only access the container in the CODB where the information resides. The World State Manager (WSM) portion of the CODB handles incoming and outgoing network traffic from a simulation application and also maintains the world state for all entities controlled outside of its host.

The Rapid Exploratory and Evolutionary Prototyping (REEP) methodology is a methodology that supports quick extraction and refinement of requirements, experimentation with alternative means for satisfying requirements, and rapid incorporation of the solutions developed by successful experiments into the application. Exploratory prototyping examines an implementation solution within the context of an operational solution. The intent is to minimize impact on the rest of the system while evaluating this new solution. Exploratory prototyping significantly accelerates our ability to assess potential implementation solutions. The intent of evolutionary prototyping is to allow successive revisions to the overall design and implementation without making major modifications to the system. The system can then evolve over time but always remains a complete solution. The REEP process begins with the construction of an initial prototype of the application that satisfies the baseline requirements.

Baseline DMTITE Architecture

The reasoning architecture for DMTITE has three components (Figure 1): a Skills Component (SC), an Active Decisions Component (ADC), and a Physical Dynamics Component (PDC). The PDC encapsulates all the physical attributes and properties of the individual CGF. This component includes the dynamics model, entity-specific properties, performance capabilities, weapons load, sensors, damage assessment, and physical status. The PDC also computes physical state changes, such as the new CGF position in the virtual environment. The SC consists of those portions of the CGF that vary between individual entities within a type and class. This component serves to model the skills and ability of the operator of an entity. The ADC contains the intelligent decision making processes and the knowledge required to drive them. The knowledge includes the overall mission, goals and objectives, plan generation, reaction time, and crisis management ability. The ADC has three reasoning engines, the Strategic Decision Engine (SDE), Tactical Decision Engine (TDE), and the Critical Decision Engine (CDE). These engines perform long-term, near-term, and immediate reasoning for the CGF. When a conflict between the outputs of these engines occurs, the Basic Control Module arbitrates a decision. These decision engines are described further in Stytz (9).

We separate these components from the remainder of the CGF architecture and from each other to insure that modifications are isolated and will not propagate throughout the entire system. The PDC is only responsible for the basic entity maneuver information, and functions 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 to it via the system software architecture. The SC 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

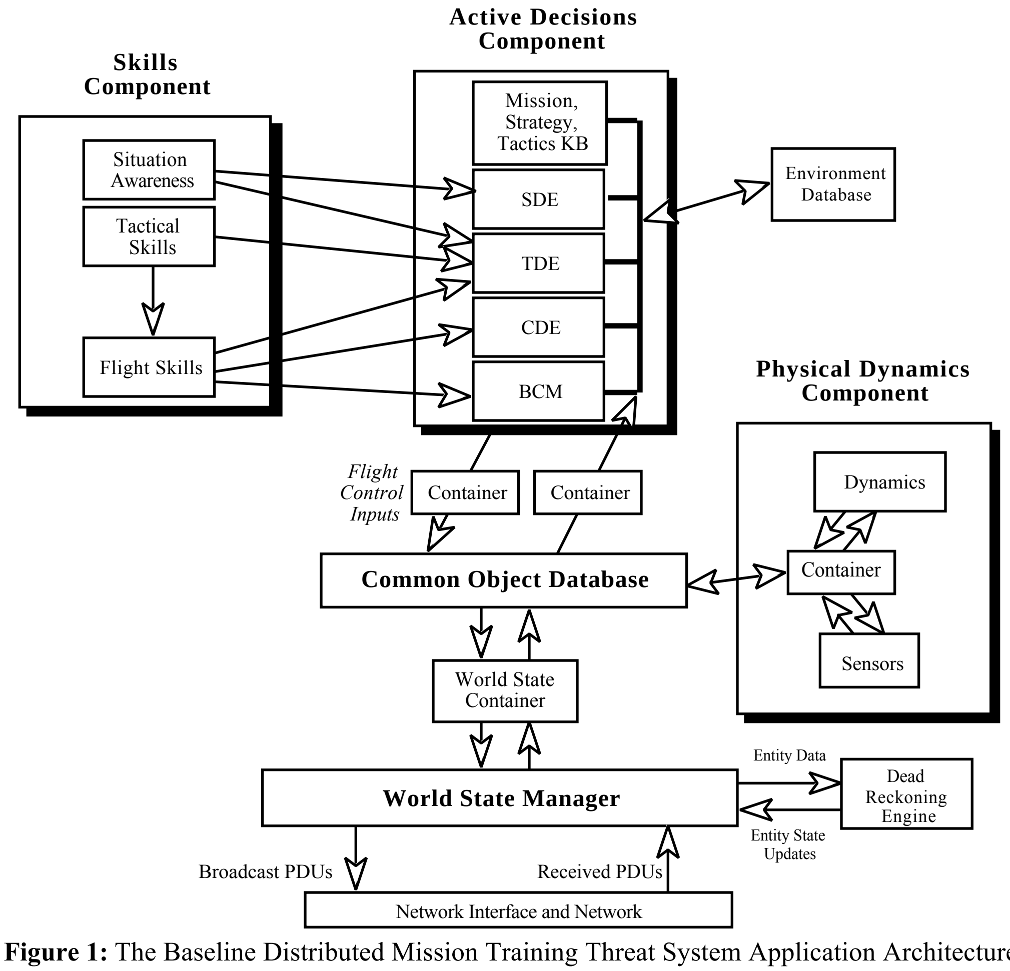

Figure 1: The Baseline Distributed Mission Training Threat System Application Architecture

modeled pilot's skills. The SC describes the pilot's ability to the decision making component so that the decision can be appropriately constrained by the simulated pilot's abilities.

Using the information described above and interviews with subject-matter experts, we defined the requirements for DMTITE in light of its concept of operations and the objectives for the application.

OPERATIONAL CONCEPT, SYSTEM REQUIREMENTS AND THEIR IMPLICATIONS

As a prelude to describing our architectural solution and to support our design decisions, this section presents three essential elements of DMTITE-specific background information. The section opens with a discussion of the operational concept for the DMTITE system and describes its proposed use. After that, we present a discussion of the system requirements that DMTITE must satisfy. The section closes by examining the implications of the requirements for the DMTITE software and knowledge base architectures.

DMTITE Operational Concept

The DMTITE system will operate within a large-scale DVE and insert a variety of accurate and highly realistic threats into the DVE for pilot training. Each DMTITE system must operate autonomously and also be able to cooperate with other DMTITE systems for the DVE to portray a coordinated threat environment. Each of the training locations, and each participating system is connected to the other participants using a high bandwidth network. Each DMTITE system requires access to a common script for a scenario, but by virtue of using artificial intelligence techniques the execution of the scenario will vary each time it is run. Reusability of training scenarios must be maximized, so the scripts developed at any DMTITE location can be used at any other location. Figure 2 illustrates DMTITE use.

In Figure 2, there are three participating locations in the DVE, Eglin, Luke, and Tyndall Air Force Bases. Each location has two dedicated DMTITE systems that insert threats into the DVE. Four manned aircraft trainer systems for pilot training are also at each location. There are two additional DMTITE systems located at other locations that are responsible for inserting RADAR threats and jamming into the DVE. Each DMTITE can insert aircraft, RADAR jamming, RADAR detection, chaff, flare, SAM, and AAA threats into the DVE, and the performance of each system can be varied to portray a variety of operator skills and tactics. Because some threats require high-speed interaction with targets, and vice-versa, threats must be able to migrate between DMTITE systems undetected by any of the human pilot trainees.

Requirements and Their Implications

The requirements for DMTITE can be broken down into several categories and have been derived from those specified for aircraft CGFs by Stytz, Banks, and Santos (9). These requirements range from the software architecture that implements the CGF to the knowledge base that is used by the CGF to support its decision making. The source for these requirements is the need to support a wide variety of training scenarios at the lowest possible unit cost while simultaneously achieving a credible representation of the behavior of the modeled entity. This section contains a discussion of each requirement: 1) a wide variety of threat systems, such as AAA, SAM, jamming RADARS, acquisition RADARS, sound and infrared detection systems, aircraft, and Unmanned Autonomous Vehicles (UAVs), 2) modifiability for knowledge bases, software and hardware, 3) high fidelity representations, 4) adaptable decision mechanisms and behaviors, 5) hardware independence, 6) exterior software independence, 7) variety of threats at varying levels of fidelity depending upon the scenario, 8) variety of skills levels for both reasoning capabilities and manual skills, 9) terrain reasoning capability as part of operator emulation, 10) DIS and HLA compatibility, 11) a graphical user interface for DMTITE system configuration, and 12) capability to download scenarios from other DMTITE locations and reuse them. The next few paragraphs delve into a select few of these requirements and assess their implications for the DMTITE project.

A variety of threat systems. For the DMTITE to achieve its objective for operational training, it must be capable of inserting a variety of threat systems into the DVE. The project's goal is to remove the threat systems from the pilot training system and to distribute them across the network; therefore, every threat system must have at least a generic representative within DMTITE.

Modifiability. Modifiability is the ability to enhance existing CGF capabilities and includes the ability to rapidly expand a domain-specific knowledge base, a flexible software architecture, the capability to operate on a variety of hardware, and independence from external software. The requirement for knowledge base expandability addresses the need for the CGF to incorporate new strategies, tactics and maneuvers as ally and opponent concepts change. This requirement also addresses the need to maintain DMTITE in the field and supports improvement of the system's performance by permitting well-encapsulated changes to the knowledge base. A flexible software architecture likewise insures that DMTITE can readily adapt to meet new performance, interface, and communication protocol requirements. The architecture should be able to support the fielded system. While the software architecture should not change often once the system is fielded, there will be system upgrades and the software

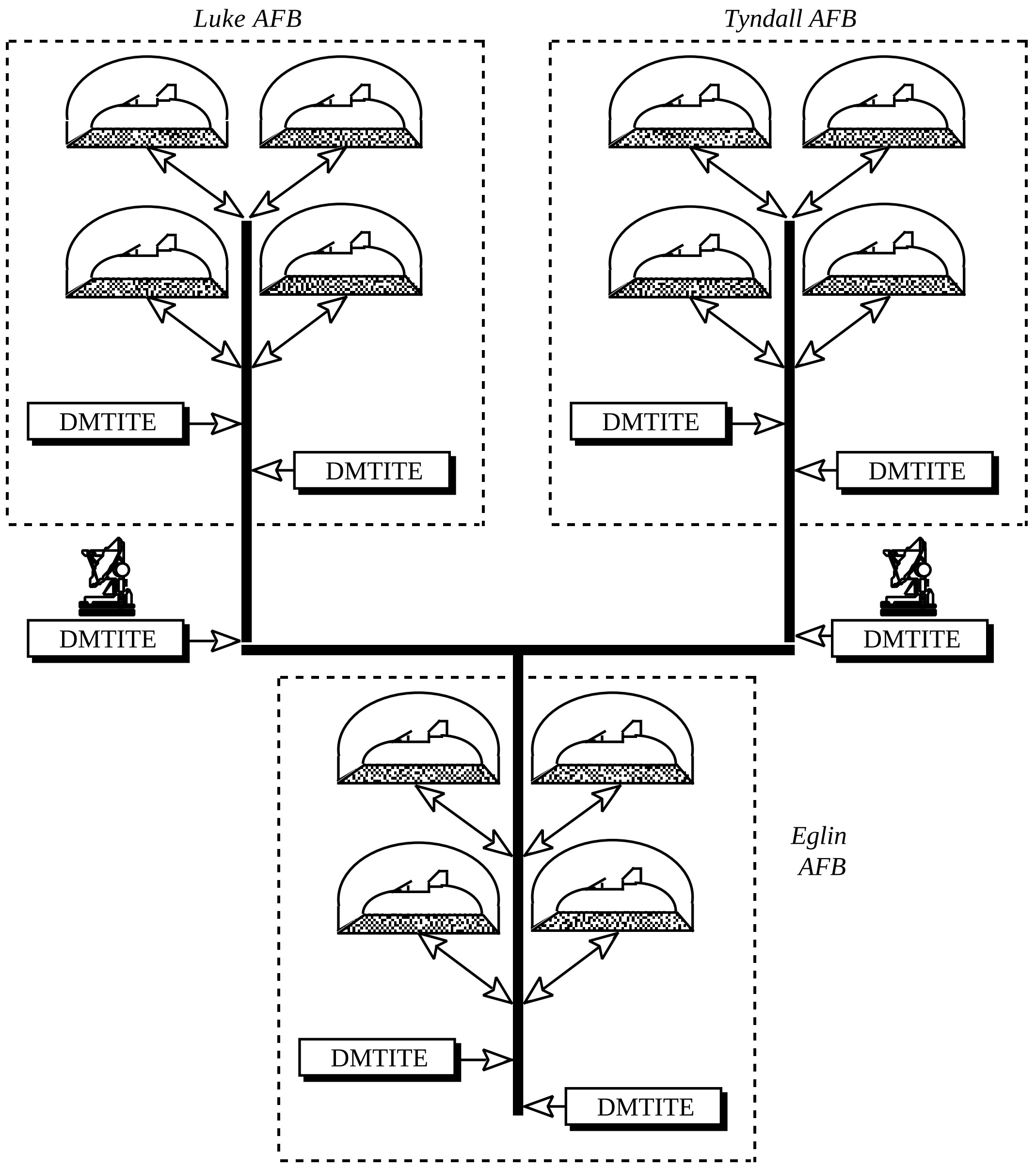

Figure 2: DMTITE Operational Concept

architecture should be able to accommodate these changes with limited impact upon the rest of the system. Hardware independence addresses the need to be able to move DMTITE to new, more capable computer hardware with minimum rewrite of the application code. The need to remain independent from external, non-DMTITE software supports the need to remain independent of hardware and to allow the system to take advantage of developments that can improve its performance. DMTITE must be independent of any software encumbrances that would impede porting it to a new hardware platform or operating system.

High fidelity representations. High fidelity representations in DMTITE are achieved by enabling CGF operation using accurate: 1) world representation, 2) dynamics for vehicle motion, 3) sensor and weapons models, and 4) models of human behavior. The world representations are based upon surface representations composed from primitive data elements organized within a hierarchical representation of the terrain data. However, since CGFs do not operate in isolation, their world representation must have a high fidelity counterpart for manned systems as well as for other CGF systems. The issue of implementing correct dynamics for vehicle motion is another aspect of achieving a high fidelity representation. Correct vehicle dynamics insures that the vehicle only moves according to its capabilities and does not achieve a level of performance that is unrealistic given the terrain, weather, and atmospheric conditions. Likewise, the weapons and sensor models must use the same sensitivity, field of view, and range as its real-world counterpart. This level of fidelity must span the variety of sensors from the eyesight of the operator of the CGF to the RADAR and Infrared sensing systems of the CGF. This requirement is discussed further in Stytz (9).

Adaptable decision mechanisms. Adaptable decision mechanisms allow the CGF to exhibit a degree of flexibility in dealing with situations that occur in the virtual environment. The decision mechanisms must adapt to the amount of information that is available. Adaptable decision mechanisms permit the system to maintain robust, credible behavior for the DMTITE at run-time under a variety of external circumstances and at different levels of operator skill. The sub-requirement for robust, credible behavior is necessary so that each threat instantiated by DMTITE can act and react even when confronted by conflicting or incomplete information and when under system stress. If the threat does not exhibit robust behavior, then the threat will fail or have a scripted pattern of behavior.

Threats at varying levels of fidelity and a variety of skills levels. The first component of this requirement speaks to the need to conserve computational power. Threats should be available to the exercise designer at multiple levels of fidelity so that only those threats that require a high fidelity representation are represented that way and are allowed to consume a correspondingly greater amount of computational resources. Irregardless of the level of fidelity, each threat should also be available in a range of skill levels. Multiple skill levels allow the training to be tailored to the abilities of the human participants and provide a more realistic training situation because the opponents and allies exhibit a variety of capabilities to train against; therefore, the training environment is more realistic. The skills can be realized by varying manual skills, by varying the range of options available to the decision making component, by varying the knowledge about friendly and enemy tactics available to the decision making component, and by permitting the decision making component to forecast the impact of each available option on its ability to perform its mission.

DIS and HLA compatibility. The requirement for DIS compatibility stems from the need to be backward compatible with existing training systems, which are largely DIS compliant. The HLA compatibility requirement is driven by a DOD mandate. However, since each DMTITE can be rapidly reconfigured and its data output can vary, multiple Federation Object Models will be required to accommodate the variety and timing of output data from DMTITE based upon the threats being instantiated, their fidelity, and the skill level of each threat.

Graphical user interface and scenario download capability. These two requirements stem from the need to reduce the expense of configuring DMTITE to participate in an exercise. The use of a graphical user interface will help structure the available options for the system and allow the user to readily assemble and examine the number, type, fidelity, and skill for the threats a DMTITE system must provide. The scenario download capability requirement addresses the desire to reuse scenarios across DMTITE systems.

Implications Of These Requirements

The requirements for modifiability, high fidelity representations, and adaptable decision mechanisms have implications for system complexity, real-time performance, knowledge engineering, and scalability. We discuss these implications below.

The requirement to be able to instantiate a *wide variety of threat systems* within a single computer host indicates that the system must use a set of general purpose reasoning mechanisms and threat specific knowledge bases to enable reasoning about the DVE. Therefore, the architecture must enable multiple threats to share a single, shared representation of the DVE state, permit sharing of knowledge bases between like-threats, and allow each threat to transmit its data to the other DVE participants.

The requirement to achieve a *modifiable* system indicates that the system should be structured so that components are isolated from each other and so that there is loose coupling between components of the system. The benefits of isolating the system components are that it minimizes the system-wide impact of changes to the software or knowledge base and also serves to retard architectural entropy. Data movement between components should be carefully managed within the architecture and the programmer should be constrained to remain within the system's architectural approach when performing maintenance. An additional architectural consequence of the need to provide modifiability is that the control flow for the system must be a visible and separate component of the architecture. The task of knowledge engineering is complicated by modifiability because the knowledge acquisition effort must be more extensive than would otherwise be the case and can complicate the design because there must be a clean separation between the knowledge representation and the decision mechanism.

The need for *high fidelity* representations within DMTITE affects its architecture, knowledge base and information flows. The architecture must support multiple levels of fidelity in the representations of terrain, airframe, and human behavior. The data flows must insure that the information available to the decision making mechanisms accurately models the type and quantity of information available to the human operator in the real world. Additionally, DMTITE must have access to different levels of detail of information so that the it is not burdened with

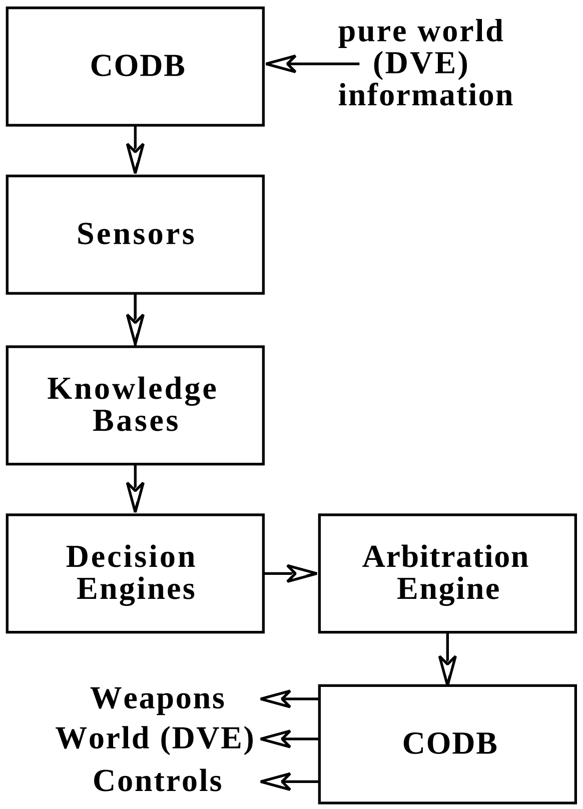

Figure 3: Data Flow Through the Application for an Individual DMTITE Threat

reasoning about high detail terrain features that are beyond its sensor range. For the knowledge base, the design should encapsulate related items of knowledge within a single access unit and insure the separation of unrelated knowledge components. This permits the decision mechanisms to atomically access the information they require and also permits the designer to update the knowledge bases with minimal impact upon other information in the knowledge base.

A key aspect of the DMTITE operational environment is that each system has perfect knowledge about the state of all of the entities in the DVE, which is an inaccurate portrayal of the real-world operational environment. As a result, each threat operating within a DMTITE system must have its information restricted so that it operates only upon a realistic model of the information that would be available to the real-world counterpart. Figure 3 portrays how this may be accomplished within the baseline architecture. As information enters the DMTITE system, it is stored within the CODB. When a threat application requires information, the information is extracted from the CODB and then passes through a sensor filter. The sensor filter functions to restrict the available information to match that of the real-world system. After sensor processing, the data is placed in the knowledge base, where the Decision Engines can access it for their decision making computations. The Arbitration Engine takes the results of the Decision Engine's work and decides which outputs should be acted upon. The outputs from the Arbitration Engine are then forwarded to the CODB where other threat components, such as weapons or system controls, use the outputs as part of their computations for emulating a threat.

The requirements for *adaptability*, *multiple skill levels*, and *multiple levels of fidelity* affect the knowledge base and the decision making components in several regards. Firstly, the decision making component must robustly deal with incomplete information and uncertainty. The decision mechanism must be structured so that the amount of information considered when making the decision can be adaptively varied and so that additional possibilities can be considered as time and circumstances permit. Secondly, because the system requires general-purpose decision mechanisms the knowledge base component must contain enough information to allow the decision mechanism to satisfy the requirements for multiple skill levels and multiple levels of fidelity. As a result, the knowledge base architecture must support partitioning of the knowledge by level of fidelity and level of skill and allow access to the database accordingly. Finally, the need for multiple levels of fidelity affects the decision making component in that one means of achieving computational savings is to alter the type of reasoning performed. Therefore, the decision making component of the DMTITE architecture must support the use of different reasoning systems for a given threat without requiring changes to the threat's knowledge base.

A SOLUTION

In this section we describe our proposed solution to the DMTITE system requirements presented previously. This section opens with a discussion of the software architecture solution we developed. The section concludes with a discussion of the DMTITE knowledge architecture.

The architectural solutions presented in Figure 4 is based upon the architecture outlined in Figure 1. The main architectural components of the generic CGF architecture are maintained, with a few key differences. We use one CODB that is shared among all the entities instantiated by a single DMTITE system host computer. The CODB and WSM combination serves to place data into the DVE and receive data into DMTITE. The CODB is also used to route data between threats instantiated within the individual DMTITE system. Each threat type within a DMTITE system shares a single knowledge base, the type of data accessed from the knowledge base is determined by the fidelity level and skill level of each threat instantiation. Currently, the knowledge bases are read-only at run-time. The decision mechanisms within each threat are

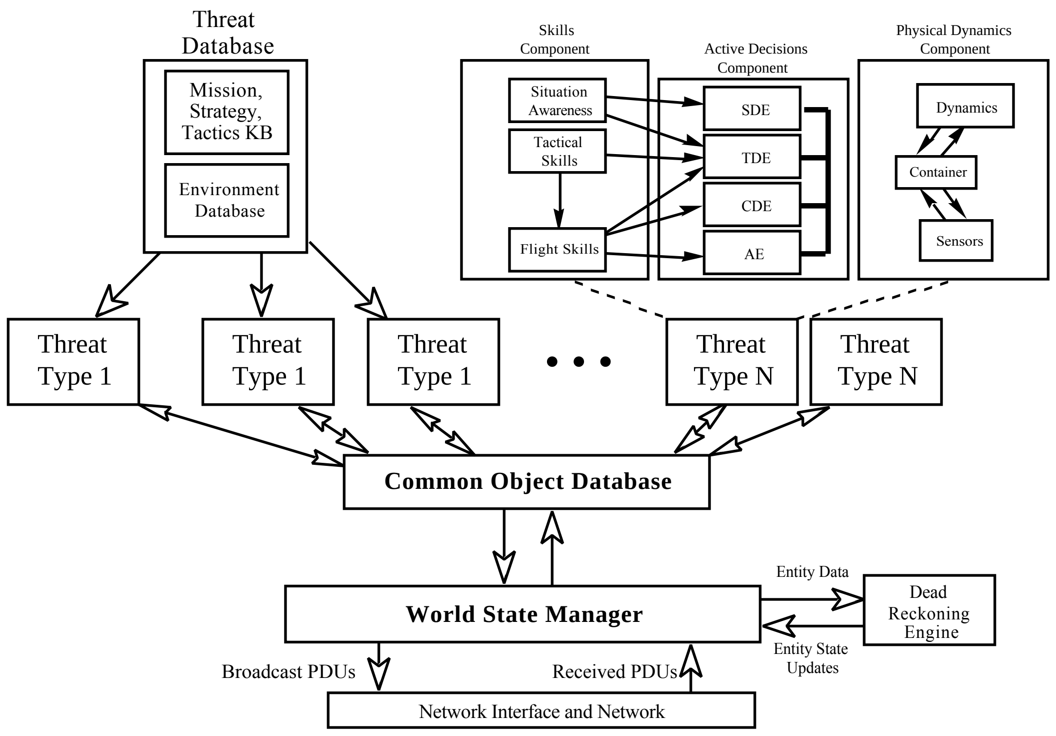

Figure 4: DMTITE System Architecture

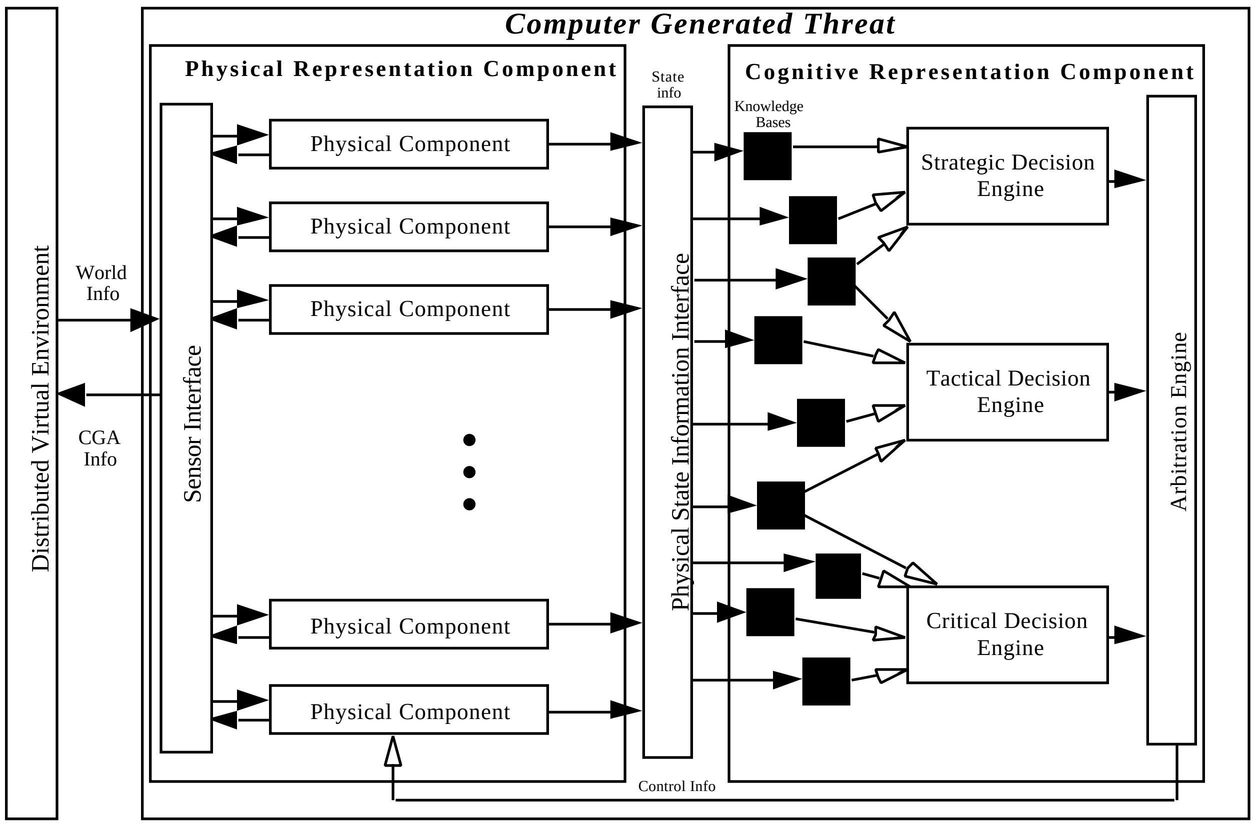

Figure 5: Data Flow Through the DMTITE Architecture

instantiated along with the threat, they are not shared between instantiations. The chief difference between the DMTITE reasoning mechanisms and the generic system in Figure 1 is that the knowledge base is removed from the reasoning mechanisms. As in the baseline architecture, we rely upon the CODB approach to achieve loose coupling between system components while also enabling the best possible system performance. By virtue of sharing knowledge bases between threats, the difficult and expensive knowledge base construction process must only be accomplished once, the burden of achieving different levels of fidelity and skills falls upon the decision mechanisms and its access mechanisms to the knowledge bases.

Figure 5 presents another view of the architecture, but in this instance we are examining the data flow through the system. This view of the architecture illustrates how the architecture supports the data flow presented in Figure 3. We view this processing as consisting primarily of two stages, modeling of the physical world state and then reasoning upon the state. The reasoning outputs are then used to control the threat and to generate outputs for the DVE. After the DVE data enters DMTITE via the WSM and CODB, it is passed to each threat after being appropriately processed by a sensor modeling unit. The Sensor Interface is a component of each individual threat type and models the type and fidelity of the data available to the threat operator in the real-world system. The Physical Component then combines its state with the Sensor Interface output and passes this summary of the DVE state to the decision making component. Upon entry to the Cognitive Representation component, the incoming state data is operated upon by the SDE, TDE, and CDE in conjunction with the threat knowledge bases. The SDE, TDE, and CDE place the outputs of their computations into the Arbitration Engine (AE), which selects the actions that are fed back to the physical component to be acted upon. This view of the architecture highlights the activity of the sensor models and also illustrates how information and knowledge bases are shared between types of threats and between instantiated threats in a single DMTITE system.

SUMMARY

In this paper we presented an architectural solution to the requirements for a distributed mission training threat system. The requirements that we presented were, in turn, derived from a consideration of the operational concept for the DMTITE as well as from conversations with threat system subject matter experts. The architectural solution we developed is based on the CODB architecture and permits the use of REEP, insures the isolation of reasoning components from knowledge base components, permits multiple threats to be instantiated within a single DMTITE system, restricts available information to a model of the information available in the real-world, and allows a DMTITE system to insert multiple threats of many different types into a DVE.

The first DMTITE system is under development at the time of this writing. The first demonstrations are scheduled for Fall, 1997.

REFERENCES

1. Blau, B.; Hughes, C. E.; Moshell, J. M.; & Lisle, C. (1992) "Networked Virtual Environments," *Proceedings of the 1992 Symposium on Interactive 3D Graphics*, Cambridge, MA, 157-160, 29 March - 1 April.
2. Blau, B., Moshell, J. M., & McDonald, B. (1993) "The DIS (Distributed Interactive Simulation) Protocols and Their Application to Virtual Environments," *Proceedings of the Meckler Virtual Reality '93 Conference*, San Jose, California, 19 - 21.
3. Calder, R.B., Smith, J.E., Courtemanche, A.J., Mar, J.M.F., & Ceranowicz, A.Z. (1993) "ModSAF Behavior Simulation and Control," *Proceedings of the Third Conference on Computer-Generated Forces and Behavioral Representation*, Orlando, FL, 347-356.
4. Edwards, M. & Stytz, M. R. (1996) "The Fuzzy Wingman: An Intelligent Companion for DIS-Compatible Flight Simulators", *The SPIE/SCS Joint 1996 SMC Simulation Multiconference: 1996 Military, Government, & Aerospace Simulation Conference*, vol. 28, no. 3, New Orleans, Louisiana, 77 - 82.
5. Laird, J. E., Newell, A., & Rosenbloom, P.S. (1987) "SOAR: An Architecture for General Intelligence," *Artificial Intelligence*, vol. 33, 1-64.
6. Laird, J. E., et. al.. (1995) "Simulated Intelligent Forces for Air: The SOAR/IFOR Project 1995," *Proceedings of the Fifth Conference on Computer Generated Forces and Behavioral Representation*, 27-36, Orlando, FL.
7. Stytz, M.R. (1996) "Distributed Virtual Environments," *IEEE Computer Graphics and Applications,* vol. 16, no. 3, pp. 19-31.
8. Stytz, M. R., Adams, T., Garcia, B., Sheasby, S. M., & Zurita, B. (1996) "Developments in Rapid Prototyping and Software Architecture for Distributed Virtual Environments," *IEEE Software*, vol. 12, to appear.
9. Stytz, M. R.; Banks, S. B.; & Santos, E. "Requirements for Intelligent Aircraft Entities in Distributed Environments," *18th Interservice/Industry Training Systems and Education Conference*, Orlando, Florida, 3 - 5 December 1996, publication on CD-ROM.
10. Tambe, M., Johnson, W. L., Jones, R.M., Koss, F., Laird, J. E., Rosenbloom, P. S., & Schwamb, K. (Spring 1995) "Intelligent Agents for Interactive Simulation Environments," *AI Magazine*, vol. 16, no. 1, 15-40.