

Retrospective Theses and Dissertations

1996

Semantic correlation of behavior for the interoperability of heterogeneous simulations

Christopher James Dean
University of Central Florida

 Part of the [Computer Engineering Commons](#)
Find similar works at: <https://stars.library.ucf.edu/rtd>
University of Central Florida Libraries <http://library.ucf.edu>

This Masters Thesis (Open Access) is brought to you for free and open access by STARS. It has been accepted for inclusion in Retrospective Theses and Dissertations by an authorized administrator of STARS. For more information, please contact STARS@ucf.edu.

STARS Citation

Dean, Christopher James, "Semantic correlation of behavior for the interoperability of heterogeneous simulations" (1996). *Retrospective Theses and Dissertations*. 2897.
<https://stars.library.ucf.edu/rtd/2897>

SEMANTIC CORRELATION OF BEHAVIOR FOR THE
INTEROPERABILITY OF HETEROGENEOUS SIMULATIONS

by

CHRISTOPHER JAMES DEAN
B.S.E., University of Florida, 1994

THESIS

Submitted in partial fulfillment of the requirements
for the degree of
Master of Science in Computer Engineering
College of Engineering
University of Central Florida
Orlando, Florida

Spring Term
1996

ABSTRACT

A desirable goal of military simulation training is to provide large scale or joint exercises to train personnel at higher echelons. To help meet this goal, many of the lower echelon combatants must consist of computer generated forces with some of these echelons composed of units from different simulations. The object of the research described is to correlate the behaviors of entities in different simulations so that they can interoperate with one another to support simulation training. Specific source behaviors can be translated to a form in terms of general behaviors which can then be correlated to any desired specific destination simulation behavior without prior knowledge of the pairing. The correlation, however, does not result in 100% effectiveness because most simulations have different semantics and were designed for different training needs. An ontology of general behaviors and behavior parameters, a database of source behaviors written in terms of these general behaviors, and heuristic metrics are used to compare source behaviors with a database of destination behaviors.

This comparison is based upon the similarity of sub-behaviors and the behavior parameters. Source behaviors/parameters may be deemed similar based upon their sub-behaviors or sub-parameters and their relationship (more specific or more general) to destination behaviors/parameters. As an additional constraint for correlation, a conversion path from all required destination parameters to a source parameter must be found in order for the behavior to be correlated and thus executed. The length of this

conversion path often determines the similarity for behavior parameters, both source and destination.

This research has shown, through a set of experiments, that heuristic metrics, in conjunction with a corresponding behavior and parameter ontology, are sufficient for the correlation of heterogeneous simulation behavior. These metrics successfully correlated known pairings provided by experts and provided reasonable correlations for behaviors that have no corresponding destination behavior. For different simulations, these metrics serve as a foundation for more complex methods of behavior correlation.

TABLE OF CONTENTS

LIST OF TABLES	vi
LIST OF FIGURES	vii
CHAPTER 1 - INTRODUCTION	1
Simulation	2
Simulation Training	3
Distributed Interactive Simulation	4
Computer Generated Forces	8
Advanced Distributed Simulation and Interoperability	17
CHAPTER 2 - BACKGROUND	21
Behavior Representation for CGF.....	21
Intelligent Agents	29
Command and Control Behavior	32
Mission Planning	39
Case-Based Reasoning Behavior	44
Context-Based Reasoning Behavior	46
Reactive Behavior Architectures	47
Interoperable Linkages	52
Constructive-Virtual Linkages	52
Constructive-Constructive Linkages	60
CHAPTER 3 - PROBLEM DEFINITION	62
Contributions of Research	67
CHAPTER 4 - BEHAVIOR INTEROPERABILITY	68
Behavior Representation	72
Behavior Correlation Metrics	73
Parameter Correlation Metrics	78
Incremental Decomposition and Abstraction	79
Related Work	83
CHAPTER 5 - EVALUATION PROTOTYPE	88
CATT-SAF	88
ModSAF	90

Implementation of Approach	94
Correlation Implementation	98
Behavior Translation	109
Parameter Translation	112
 CHAPTER 6 - PROTOTYPE TESTING AND EVALUATION	 113
Comparison of CCTT and ModSAF Behaviors	116
Proof of Principle	117
CCTT and ModSAF Reactive Behaviors	119
Experiment 1	124
Experiment 2	129
Experiment 3	132
Experiment 4	136
Experiment 5	139
Experiment 6	142
Experiment 7	145
Experiment 8	149
Experiment 9	151
Experiment 10	154
Experiment 11	158
Experiment 12	161
Experimental Conclusions	164
 CHAPTER 7 - SUMMARY AND CONCLUSION	 166
 CHAPTER 8 - FUTURE WORK	 171
 APPENDIX	 174
Correlation Algorithm Pseudocode	174
Parameter Correlation Algorithm Pseudocode	182
 REFERENCES	 188

LIST OF TABLES

1. CCTT and ModSAF Tank Platoon Behaviors	117
2. CCTT-ModSAF Correlations	118
3. Summary of Experimental Results	165

LIST OF FIGURES

1. CCTT Assault Behavior in Terms of General Representation	97
2. General Parameter Representation	98
3. Partial Hierarchy for Tank Platoon Behaviors	102
4. Partial Hierarchy for Behavior Parameters	106
5. ModSAF Assault	111
6. ModSAF Assault in General Form	111
7. Example SAF Behaviors	114
8. ModSAF React to Enemy Contact Behavior	120
9. ModSAF React to Air Attack Behavior	121
10. ModSAF React to Indirect Fire Behavior	122
11. CCTT Actions on Contact Reactive Behavior	123
12. CCTT React to Indirect Fire Reactive Behavior	124
13. CCTT Assault An Enemy Position Behavior	125
14. ModSAF Assault Behavior	127
15. CCTT Attack By Fire Behavior	130
16. ModSAF Attack By Fire Behavior	131
17. CCTT Bounding Overwatch Behavior	133
18. ModSAF Overwatch Movement Behavior	135
19. CCTT Traveling Overwatch Behavior	137

20. ModSAF Traveling Overwatch Behavior	138
21. CCTT Tactical Road March Behavior	140
22. ModSAF Breach Behavior	141
23. CCTT Travel Behavior	143
24. ModSAF Travel Behavior	144
25. CCTT Consolidate and Reorganize Behavior	146
26. ModSAF Delay	148
27. CCTT Occupy Battle Position Behavior	149
28. ModSAF Passage of Lines Behavior	152
29. CCTT Platoon Defensive Mission Behavior	155
30. CCTT Platoon Fire and Movement Behavior	159
31. CCTT Hasty Occupy Position Behavior	162
32. ModSAF Hasty Occupy Position Behavior	163

CHAPTER 1

INTRODUCTION

Simulation interoperability can be defined in general terms as the ability of simulations to share a common environment and work together to support a common goal. Working together may involve resolving differences in communication protocols, system behavior, system timing, etc. Sharing a common environment may involve resolving differences in system fidelity, representation, databases, environment behavior, etc. It is these differences that cause the interoperability of simulations to be a major problem in the simulation community.

In order to fully understand the interoperability problem and all its aspects, several concepts need to be discussed first. The concepts of simulation and simulation training will be defined to provide the context for the discussion. Furthermore, foundational concepts such as distributed interactive simulation and computer generated forces will be discussed to set the stage for the problems that can occur when trying to connect dissimilar simulations. Finally, the problem of interoperability, i.e. the concept of advanced distributed simulation, and its corresponding issues will be discussed. The semantic interoperability problem is but one of the many interoperability issues.

Simulation

Simulation is a technique that allows the comprehension of reality by representing it using artificial objects and acting out scenarios with them. More specifically, the modeling of reality allows the understanding of time-varying phenomena. Computer simulation is a more specific simulation discipline which involves three basic steps: the designing of the model of a physical or theoretical system, the execution of the model, and the analysis of the results [Fishwick, 1995]. There are many ways to model systems: conceptual, declarative, functional (function-based approach, variable-based approach), constraint modeling (equation-based, graph-based), spatial modeling (space-based, entity-based), and multi-modeling (a combination of different models) [Fishwick, 1995]. The system in question can be modeled under varying levels of abstraction, whichever are necessary for the needs of the problem. In cases where no one model is sufficient, the system can be modeled using a multi-model of different models at different levels of abstraction connected in a seamless fashion. Simulation models can be executed in a serial or parallel fashion and varying kinds of execution analyses can be performed such as input-output analysis, experimental design, surface response techniques, data visualization, verification, and validation [Fishwick, 1995].

Simulation plays several roles in current research. Simulation plays a role in what is termed “computational science”, i.e. the visualization and simulation of large scale complex systems such as weather systems and molecular dynamics. Similarly, simulation is important in the study of chaotic and complex systems such as nonlinear systems.

Virtual reality is simulation taken to its maximum degree, the immersion of the analyst into the simulation itself. The potential of virtual reality is enormous and yet to be fully realized. Simulation can be used to experiment with artificial life which is a topic of much debate [Fishwick, 1995]. Finally, simulation can be used in physical modeling and computer animation. Typical physical models not only share the appearance of real-world objects but obey the same physical laws.

Simulation Training

Simulation is not only used to represent reality as a means of understanding it but also for the purpose of training. Simulation training can be used with any simulation under any role. Its use is an emerging field. Currently, simulation is being used to train military personnel (infantry, tank commanders, battalion commanders, etc.), nuclear reactor operators, aircraft pilots, radar operators, oil tanker pilots, etc. It should also be noted that simulation is also being used for the design of aircraft, circuits, computers and the like. As computing power becomes more widely available, other domains that can benefit from simulation such as medicine and economics will incorporate it into their operating environment which includes training, modeling, development, planning, and design [Cohen, 1994].

In industry and the military, computer-based simulation is increasingly being used in training because it is cost effective and is able to simulate real world conditions that would otherwise be impossible to duplicate. In the military domain, large scale exercises

can be simulated and used for training at all echelon levels from battalion down to the individual infantry. The worthiness of training as been shown many times, most recently during Kernel Blitz 95 [Neuberger and Shea, 1995]. The exercise showed that a Synthetic Theater Of War (STOW) created by simulation technology can provide valuable training for a variety of different roles, from actual combatants to support staff. The real-time aspect of the battlefield often makes the training of support staff difficult if not impossible. Simulation training enhances the training of support staff by making it less sensitive to the pace of the battlefield and introducing cost savings. This cost savings was suggested during Kernel Blitz 95 [Neuberger and Shea, 1995].

Distributed Interactive Simulation

To support military simulation training, the Department of Defense (DoD) has mandated the use of a framework and standard set of protocols to create a time and space coherent synthetic representation of the battlefield environment, known as Distributed Interactive Simulation (DIS). DIS is a entity-based simulation approach that allows large scale simulations to be built from independent simulator nodes which are linked via a common network protocol. Each simulator node independently simulates one or more entities and reports events over the network. A common terrain database is used to represent the shared environment. Because the simulator nodes are networked and the architecture is scaleable (within bandwidth constraints) many trainees can simultaneously participate in a training exercise and thus be effectively trained in team tactics [Petty, 1994]. DIS serves as the low level background for most military (and some non-military)

simulations. It is used to support the real-time interaction of autonomous simulations, manned simulators, and equipment in live arenas. Since its main purpose is to support military simulation, the world is most often modeled as a set of combat entities (tanks, infantry fighting vehicles, infantry, aircraft, etc.) that interact with each other via events that they cause. These events are in turn perceived by other entities causing other effects and so on. Some key DIS design principles include [Institute for Simulation Training, 1994]:

- No central system that controls event scheduling.
- Autonomous simulation nodes.
- Sending nodes emit “ground truth” data, receiving nodes are responsible for perception of that ground truth, i.e. their view of the real world with environmental effects taken into account.
- To decrease network traffic, each entity uses an algorithm known as “dead reckoning” that estimates the position of itself and other entities. When the difference between the actual position and predicted position of an entity surpasses a given threshold, that entity updates the other simulators with a position update. For more on dead reckoning, see [Fishwick, 1995].
- A shooting entity determines whether a target was hit, and the target determines the damage and effect.

Additionally, the large set of critical parameters that support DIS [Humphrey, 1994] include:

- Entity performance parameters
 - Speed
 - Acceleration
 - Angle of Attack

- Perceptual limits
 - Visual
 - Audio
- Rates of fire
- Capacities (fuel, ammunition, soldiers, etc.)
- Articulated Parts
 - Enumerations (DIS characteristic constants)
 - Range of motion
 - Rates
 - Limits
- Kinds of weapons
 - Warheads
 - Fuses

The initial focus of DIS application development has been on training of large, joint, or combined forces which is lacking in traditional training. [DIS Steering Committee, 1994]. The DIS mission is defined as:

“The primary mission of DIS is to define an infrastructure for linking simulations of various types at multiple locations to create realistic, complex, virtual ‘worlds’ for the simulation of highly interactive activities. This infrastructure brings together systems built for separate purposes, technologies from different eras, products from various vendors, and platforms from various services and permits them to interoperate. DIS exercises are intended to support a mixture of virtual entities (human-in-the-loop simulators), live entities (operational platforms and test and evaluation systems), and constructive entities (wargames and other automated simulations).

The DIS infrastructure provides interface standards, communications architectures, management structures, fidelity indices, technical forums, and other elements necessary to transform heterogeneous simulations into unified seamless synthetic environments. These synthetic environments support design and prototyping, education and training, test and evaluation, emergency preparedness and contingency response, and readiness and warfighting.” [DIS Steering Committee, 1994]

The protocol component of the DIS framework allows the simulated entities to communicate with one another and defines the various operations that can occur on the synthetic battlefield such as changes in entity state (damage, dead reckoning, etc.), firing weapons, weapons detonations, resupply, etc. [IST, 1994]. For example, the most common protocol packet sent during a DIS exercise is the entity state packet. Using it, simulated entities send location and damage information over the network which is used by other entities simulated by other simulators to generate the visual representation. Additional entity actions are communicated through collision packets, fire packets, radio communication packets, and radar/EM emissions packets. The DIS Vision defines these protocols as a:

“set of protocols that convey messages about entities and events, via a network, among various simulation nodes that are responsible for maintaining the status of the entities in the virtual world. The characteristics of the network are not important, as long as it can convey these messages to the interested simulation nodes with reasonably low latency (100 - 300 ms) and low latency variance. Within these constraints, the systems that generate entities that appear to be adjacent in the virtual world could be separated by thousands of miles in the real world.” [DIS Steering Committee, 1994]

The DIS protocol is defined over a set of protocol data units (PDUs) used for entity information, weapons fire, logistics support, collisions, simulation management, electromagnetic emissions, and radio communications. Specifically, some PDUs include:

- Entity State PDU
- Fire PDU
- Detonation PDU
- Service Request PDU

- Resupply Offer PDU
- Resupply Received PDU
- Resupply Cancel PDU
- Repair Complete PDU
- Repair Response PDU
- Collision PDU
- Create Entity PDU
- Remove Entity PDU
- Start/Resume PDU
- Stop/Freeze PDU
- Acknowledge PDU
- Message PDU
- Emission PDU
- Designator PDU
- Transmitter PDU
- Signal PDU
- Receiver PDU

DIS is meant to be the canonical paradigm for distributed interactive simulation.

Unfortunately, this is not entirely true as will be shown later. There are many issues involved in the support of DIS [Cohen, 1994] but these are not of concern here. What is of concern are the issues involving the use of Computer Generated Forces (CGF) and Semi-Automated Forces (SAF) for training in a DIS environment.

Computer Generated Forces

DIS simulations usually include special simulation applications known as CGF or SAF nodes. Because of the human resources needed to train upper and/or lower echelon personnel in large combat situations, CGFs are needed to provide a more robust training environment without additional manpower. CGF systems initially came into being as a result of the need to provide threat vehicles or supplementary friendly forces to train

personnel on simulators. CGF is able to provide realistic complementary forces and enemy forces. For example, at the Joint Readiness Training Center, CGF is used to represent one of three battalions and the corresponding OPFOR (opposing force) to provide realistic training of command and control for the regimental and battalion commanders [Jones, 1993]. In addition, the CGF can be used to represent any special force elements needed, joint force elements, and even coalition forces. Since 1990, CGF systems have been augmented to act not only as a simulator of threat and friendly vehicles but to act as a virtual experimentation environment. CGF permits experimentation with new doctrine and operational plans over diversified conflicts and experimentation with new equipment (this is known as a Battle Lab within the military) without the expenditure of the considerable time and money necessary to conduct a field exercise with geographically dispersed assets [Jones, 1993]. As an example, CGF was used in a scenario that involved using the Army Tactical Missile System (ATACMS) to destroy time-critical, high-priority targets such as air defense sites [Jones, 1993]. CGF was used to evaluate the length of time from target acquisition through the decision process and weapon-on-target in this time-critical operation. This usage has resulted in the various kinds of simulations that have been and are continuing to be developed to meet the diversified needs of the military. These additional features include the ability to support a virtual battlefield composed of not only vehicles and aircraft but three dimensional terrain and battlefield environment. In addition, these virtual simulators have been interfaced with live personnel and sensor simulators such as J-STARS to provide command and control

decisions. The CGF was used to provide J-STARS information (which was evaluated for usefulness), providing trainees with the opportunity to work with new systems before they are fielded and obtain otherwise hard to obtain data on behavior and response times [Jones, 1993]. An example from Jones [1993] includes the analysis of the timeliness of AWACS operators associated with controlling interceptors. The analysis provided important data on the survivability of tactical and strategic low observable vehicles. With the virtual CGF systems, new attack capabilities, force structures, tactics, equipment, and Tactics, Techniques, and Procedures (TTPs) can be demonstrated and analyzed. Typical examples of these experiments include the Advanced Warfighting Demonstrations (AWDs) and Advanced Technology Demonstrations (ATDs) [Pickett and Petty, 1995]. In addition to training, CGFs serve as a device for operations planning and mission rehearsal. Portions of a mission can be practiced or rehearsed as part of on-going training and readiness of a component of a force. This can identify problem areas or weaknesses of the planned mission. Military commanders can also use the results of CGF to gain experience and exposure to the various eventualities that may occur during the mission.

Currently, a military simulation may be virtual, constructive, or live. Each varies in the training resolution, timing mechanisms, and user interactions. Virtual simulations exhibit a high resolution representation of the battlefield, often simulating individual entities that make up larger units (tanks, dismounted infantry). They are typically real time and interact with the user in an asynchronous, time-driven manner. Because of their resolution, virtual simulations only provide a limited set of combat events. Constructive

simulations are lower resolution simulations designed to train upper level tasks such as battalion command and logistics (medical and supply, for example). Groups of entities are represented in an aggregate manner with individual entity actions and results simulated using force probability functions such as Lanchester equations [Taylor, 1983]. Lanchester equations are simple equations used to measure combat attrition, i.e. to decide which force is affected and how much strength is lost on an aggregate, not unit, level. Constructive simulations sacrifice entity and event detail for the breadth of operations that can be performed. Typically, constructive simulations run in faster than real time but can be adjusted to any time frame. They interact with the user in a synchronous time step fashion based upon events. Finally, live simulations can exhibit properties of both. High resolution is used for device specific actions and lower resolution for auxiliary actions. Live simulations may be made to interact with virtual and constructive simulations but since there is no computer simulated aspect, they will not be addressed further. Both constructive and virtual simulations make use of CGFs and SAFs to enhance training. CGFs are often used to represent the actions of opposing forces (OPFOR) as well as representing additional units (such as platoons) for friendly forces (BLUFOR). The only difference between constructive and virtual as far as CGFs are concerned is the resolution of representation and resolution of behavior. For an overview of the various constructive and virtual simulations see [Sandmeyer and Dymond, 1995]. For constructive simulations, the CGF component is concerned with the automation of command and control decision making. Command and control is the process of analyzing the situation and issuing orders

to subordinate units. For virtual simulations, the CGF is more concerned with reactive behaviors such as reacting to an enemy attack. SAFs are similar to CGFs in that SAF units contain CGFs at the lower echelons with a trainee at the higher echelons such as the company or battalion commander. Since the trainee controls the lower echelons via orders, they are termed semi-automated. The unmanned units still are responsible for their reactive and primitive behaviors (move, shoot, etc.). SAF arises from the need to have experienced soldiers in the simulation to interject behaviors into the system that are difficult to simulate automatically with present technology. Another advantage for CGF, aside from the human resource issue, is that using CGF for lower echelon units allows the pace of the exercise events to be tailored to the handling capacity of the echelon being trained without concern that the planning phases will generate delays in the action for lower echelon units [O'Byrne, 1993].

CGF systems are characterized by a set of standard objectives/benefits [Jones, 1993; Weaver, 1993; Picket and Petty, 1995]:

- CGF systems must support training, advanced technology demonstrations, and analysis which includes support for man-in-the-loop simulators at any echelon level, live interfaces at any echelon level, real-time or faster than real-time processing, and constructive-virtual interfaces.
- CGF systems must provide a realistic operating environment and varying scenarios.
- CGF systems should provide analytical summary information on the exercise.
- CGF systems must be DIS compatible and all that being DIS compliant implies.

- CGF systems must represent training from the individual vehicle/infantry up to the corps level.
- CGF systems should be able to interface with other service simulations that would be required in a joint exercise.
- CGF behaviors should be written in a modular and low coupled fashion to support verification and validation.
- CGF systems should be able to operate in any simulated environment.
- CGF systems can provide any fraction of an exercise.
- CGF systems should keep the number of required manned operators to a minimum. This is the focus of CGF behavior research.
- CGF behavior should be indistinguishable from the behavior of the human participants on the battlefield. This is also an important issue when considering behavior generation.

Implicit in the term CGF is the expectation of some form of intelligence and representation of decision making. Since CGF systems are used to represent OPFOR and supplementary BLUFOR forces, they must exhibit a degree of realism that allows trainees to receive positive training benefits [Petty, 1994]. CGF OPFOR units must fight like an enemy would and BLUFOR CGF units must cooperate with the trainee's force(s), i.e. they must react to the simulated situation and perform intelligent and doctrinally correct actions. The Institute for Simulation Training has researched the use of the Turing Test [Turing, 1950] as a criterion for CGF [Petty, 1994]. The Turing Test has many variants [Petty, 1994] but the most widely known formulation of the test is for an interrogator to determine, using a series of questions, if a respondent is a human or computer system. There is much controversy associated with the use of test as a criterion for intelligence,

but for CGF the basic question is “Can observers of simulated entities in the battlefield reliably determine whether any given entity is controlled by humans or a CGF system?” [Petty, 1994]. The Turing Test makes no restrictions on how the behavior is generated and as will be shown later, can vary greatly in implementation.

Passing the CGF Turing Test is easier than passing the original Turing Test since the “observer” is a trainee, not an expert on unit tactics. Also, combat operations are always conducted within the context of national and service doctrine. The doctrine is expressed at unit levels in the form of training, equipment and planning routines, operational techniques, and functional agencies that manage specific aspects of combat operations. Any commander’s decisions/plans, and thus any CGF decisions/plans, are constrained within this context. Thus, the set of possible interactions that could occur is limited to only that doctrinal set allowed by the simulation. The actions of the trainees become the questions and the CGF actions become the responses. Also, the trainee’s ability to observe other units is severely constrained by the fact that the trainees are only allowed to observe the portion of the battlefield that is visible from their location and with the visual equipment allowed by their simulated vehicle. Aggravating this situation is the fact that, in the case of OPFOR units, the units are trying to remain concealed. This restricted ability to observe the battlefield limits the ability of trainees to determine whether the units are human or computer controlled. Finally, [Petty, 1994] suggests that the trainee is probably more concerned with some activity that is pertinent to the current mission such as destroying opponents quickly as possible (missions usually have limited

time) rather than observing the opponent's behavior for signs of artificiality. See Petty [1994] for a description of some of the CGF Turing Test experiments. The experiments were designed to measure whether the CGF Turing Test is sufficient (a system passing it will surely produce positive training), necessary (system must pass it in order to be able to produce positive training benefits), or irrelevant (passing does not matter). The author concludes that the CGF Turing Test is irrelevant but does provide a useful heuristic to replace it. The experiments demonstrate that it is possible to produce positive training benefits from a system that does not pass the test and to not receive training benefits from systems that do. Passing the test only serves as evidence to the quality of the behaviors present in the system.

Thus far the discussion has centered primarily on CGF for virtual simulations. CGF also applies to constructive simulations at higher echelons, especially the joint task force (JTF) level. At this level, there are a number of agencies that can benefit from simulation training including the following [O'Byrne, 1993]:

- TACC Tactical Air Command Center (USMC)
- TACC Tactical Air Control Center (USAF)
- TACC Tactical Air Coordination Center (USN)
- TADC Tactical Air Direction Center
- TACLOG Tactical Logistics
- TAOC Tactical Air Operations Center
- DASC Direct Air Support Center
- FSCC Fire Support Coordination Center
- COC Combat Operations Center
- MTMC Military Transportations Movement Center
- DLA Defense Logistics Agency
- DIA Defense Intelligence Agency
- CRC Control and Reporting Center

- HDC Helicopter Direction Center
- TOC Tactical Operations Center

Each agency has resources assigned for management and logistics functions. These assets are balanced against the planned operations of the JTF commander. This relationship and the agency authority are specified again by doctrine. The important aspect of these agencies is that they require 15-75 specially qualified personnel to operate. CGF can play an important, and in some cases more difficult, role in training these agency personnel by replacing agency personnel with computer generated equivalents. The CGF can also train command post personnel by sending reports (situation reports, spot reports), starting at any echelon level, up the command hierarchy.

As mentioned previously, CGF serves a role in testing new equipment. [Jones, 1993] contends that this system acquisition and development is actually its greatest benefit. CGF can be used to support weapon requirements development (Cost and Operational Effectiveness Assessment -- COEA), early operational assessments (EOA), and development evaluation. The principal advantage is the ability to use a common methodology throughout the system acquisition and development process. An analyst can then focus on the results, rather than on the assumptions of the different system methodologies behind those results, because he/she better understands the system. Field test data allows the system to evolve and can serve to validate or adjust the CGF and its associated data bases, giving the analyst more confidence in the system and its results [Jones, 1993]. The CGF provides an operational context for requirements assessment and

provides important data that is frequently lacking when assessments are conducted. Using CGF, a virtual battlefield can be created with soldiers fielding experimental equipment against a realistic threat and thus deployment doctrine can be adjusted before any actual weapon construction is begun. The CGF can provide real time kill assessments with simulated test participants.

CGF also plays a role in pre-test analysis/test planning and post-test analysis. For pre-test analysis, CGF can identify critical factors to measure, scenario sensitivities, establish field test scenarios and event timelines, and predict outcomes, providing the “where” and “when” for the new equipment to be tested. In post-test analysis, CGF can be used to fill in missing information, extrapolate information, and translate those results to different locales. CGF can help determine the operational effectiveness and suitability of the new system under different robust, operationally realistic scenarios.

Advanced Distributed Simulation and Interoperability

As previously shown, CGF systems can span a wide range of fidelity and focus on many different aspects of training. Because no existing CGF system satisfies all the requirements for all users, CGF systems must interoperate with one another. All the various missions established as the vision for DIS (including civilian domains such as aviation command and control, disaster relief, distributed simulation games, and team training efforts) create specific challenges to simulation training. Attempts to meet these requirements with the flexibility and fidelity required leads to the interoperability of various CGFs, constructive and virtual. This interoperability is defined as Advanced

Distributed Simulation (ADS), and implies an ongoing evolution of simulation. The ADS environment, which may be synthetic or virtual, represents real world phenomena for the purpose of training, testing of developing systems, analysis, doctrine development, etc. It is the logical extension of DIS for the purposes of multiple heterogeneous simulations interoperating in a common environment. Also, over time, simulations will evolve and may require additional information and/or features not currently anticipated and defined, thus requiring interoperability adjustments. Interoperability can be simply defined as the

“set of explicit expectations (rules) and implicit expectations (assumptions) which are made by users in a simulation exercise “ [Riecken and O’Brien, 1994].

Expressed differently, interoperability can be defined as a measure of consistency between representations of the simulated environment. By one definition, interoperability has been achieved if the perception of the virtual space is sufficiently similar when viewed from different simulations [Altman et al., 1994]. By another definition, if the simulated outcomes match the desired training outcomes, then interoperability has been satisfied [Moskal et al., 1994]. Regardless of the definition, interoperability problems must be dealt with.

More specifically, ADS interoperability can be defined at two primary levels, the application and core level. The core level includes interoperability between network interfaces, software architectures, languages, and data representation. Development of standard interoperable software modules is an issue at this level. The application level is concerned with the interoperability between simulations and/or simulation components.

At this level, interoperable simulations can be defined as simulations using compatible protocols (valid in communicating what is being done), simulations using compatible algorithms (valid in determining how operations are being done), and in some cases simulations using compatible design requirements (valid in why operations are being done). [Smith, 1995] goes on to say that due to the limited understanding of complex processes, limitations of modeling fidelity, and increasing lack of process determinism, systems may be valid when run alone (meet the communication, algorithm, and design criteria) but invalid when combined with other simulations. The DIS protocol was an attempt to alleviate this problem. Unfortunately, the DIS standard does not provide for all types of interoperability. DIS was developed under the myth that the exchange of data would guarantee interoperability [Altman et al., 1994]. As mentioned previously, DIS does provide standards for interface definition, communication, environment representation, management, security, field instrumentation, and performance measurement. However, it does not specify entity representation standards, behavior standards, synchronization standards, or spatial coherence (correlation of terrain, resolution correlation and environment correlation such as ambient illumination, buildings, weather, etc.) standards and database standards. DIS can deal with limited forms of interoperability such as sensing interoperability, direct interactive interoperability, indirect interactive interoperability, associative interoperability, communications interoperability, and simulation management interoperability [Rush and Whitely, 1994]. Sensing interoperability is the ability for a battlefield element to sense another (fair fight) either

visually, thermally, with radar, etc. While DIS supports this, it does however have a problem with interoperating different kinds of simulations as will be discussed later.

Direct interactive interoperability is the ability for a battlefield element to physically interact with another such as moving over terrain or collisions. Indirect interactive

interoperability is similar, only an indirect method of contact such as shell fire is involved.

Associative interoperability is the ability for battlefield elements to act as though they were connected to another element such as vehicles moving in formation. Again, as will be

shown later, if the battlefield elements are controlled by different simulations, DIS does

not support complete associative interoperability. Finally, communication interoperability

is the ability for elements to communicate with one another and simulation management

interoperability is the ability to examine or control the parameters of battlefield elements.

CHAPTER 2

BACKGROUND

Behavior Representation for CGF

Behavior for CGFs have been loosely grouped into two categories: reactive and planning. Planning behavior is a traditional AI research area that offers the advantage that it is suited to general-purpose problem solving, thus allowing decisions to be made in unfamiliar situations or with unanticipated goals. Reactive behavior has an advantage over traditional planning in that CGFs using reactive behavior can react more quickly to a changing environment and thus operate more robustly in the dynamic, sometimes unpredictable, world of military simulation. As will be shown, both are used for CGF systems. It is also becoming clear that hybrid approaches using both types will be needed as the future of CGF increases in complexity. The various levels of complexity can be seen by analyzing the various kinds of behavior a CGF can be expected to perform.

CGF uses various forms of declarative modeling to generate behavior. CGF behavior can be defined on three levels: the individual level, crew level, and unit level [Pratt et al., 1994]. Individual level behaviors are characterized by decisions that are updated continuously by analysis of a priori alternatives, usually generated towards a specific goal. Typical individual behaviors include firing a specific weapon, scanning an

area, or seeking cover. The crew level is characterized by collaborative behavior. A crew commander coordinates the behavior of his soldiers to accomplish the assigned mission. The roles of each crew member vary from steering a vehicle, rotating a turret, loading a weapon, firing a weapon, etc. The crew commander has to consider more decision factors and alternatives than an ordinary individual. The unit level is characterized by the coordination of behavior. This becomes more difficult at higher levels of the military command hierarchy. The primary three functions of unit-level decision making are Command and Control (C2), route planning, and target engagement, all of which must be considered in order to provide realistic CGF. Command and control is characterized by tactical decision making, task assignment, target assessment, target assignment, fire control, and communication. Route planning is characterized by goal directed reasoning, terrain analysis, threat analysis, and vehicle/unit movement. Target engagement is simply characterized by operational decisions such as terrain assessment, sector scanning, target acquisition, weapons selection and firing. Unit level behavior is further characterized by many alternatives and difficult situational awareness above and beyond that needed for crew and individual behaviors. The lower levels focus on just the execution of a task but the unit level must also consider the selection of tasks, assignment of tasks, and coordination of those tasks. Typical tasks include movement in formations, assaults, occupying positions, etc.

Command and control behaviors consist of the planning and coordination tasks necessary for a combat unit to achieve its goals. Actions include directing the movement

and missions of subordinate units and monitoring these units for adjustments to the current executing plan. Command and control decisions occur at every unit echelon level, becoming more complex at higher echelons because of the large scope and number of units. Command and control decisions are made based upon the mission goals (offensive, defensive, recon), the battlefield situation (terrain, enemy presence and strength), and tactical doctrine as defined by the military. Based upon these factors, tasks are assigned to subordinate units, each of which in turn performs its own command and control decisions. Assignment can also be based upon unit distance to objective, support needed for other units, the priority of the assignment, and whether the unit is already in contact with the enemy.

Target assessment is based upon a doctrinal threat assessment which is based upon the unit size, unit location, firepower, mobility, armor, and intent. Typically, closer targets present a higher threat than those further away. Similarly, retreating targets are a lesser priority than advancing targets. Target assignment is performed at all echelon levels. Units are assigned targets based upon the unit's current responsibilities, distance to target, and potential threat.

Fire control concerns if and when a unit should engage and the weapon, ammunition, and rate of fire that should be used. At higher echelon levels, the coordination of fire becomes the overriding issue. At the lower echelon levels, factors for firing include target condition (damaged, destroyed, healthy, etc.), ammunition status, hit probability, synchronization objectives, and the actions of friendly forces.

Movement and route planning are characterized by selecting a path to follow, moving along the route in proper formation, and proper sector scanning for the enemy. Selecting the path is further characterized by the terrain conditions, enemy presence and cover and concealment. Route planning tries to find the optimal path necessary to meet the constraints of the given mission. Like command and control, it must perform situational awareness but usually on a more limited scale. If the enemy intent is taken into consideration, more complicated reasoning must be used when choosing a route. Complete (a priori) routes can be generated given the unit's location and objective, or the route may be generated incrementally as the unit moves along the route. Experimentation has shown that there are advantages and disadvantages to both, suggesting a hybrid approach combined with meta-level reasoning used to control the incremental and a priori methods [Pratt et al., 1994]. Unit movement along the route involves constraints such as the formation and spacing required, physical limitations of the vehicles, and terrain trafficability.

Target engagement involves sector scanning, target acquisition, and firing decisions. Constraints on engagement include limits on the degree and speed of turret rotation, gun elevation, and rate of fire. Coordination with other units must also be considered. Each vehicle in a unit is assigned a specific sector to scan for targets. Once a target has been identified, it is assigned to a vehicle or vehicles. Target acquisition uses the targets position, range, speed, and the mechanics of the weapon to determine if the target can be engaged. Aiming calculations include the target's velocity, the unit's

velocity, turret position and gun elevation. Once acquired, the target can be fired upon. As previously mentioned this is based upon ammunition status, hit probability, and number of targets assigned to the unit.

The execution of these CGF behaviors may seem straightforward but it is complicated by three fundamental problems [Gat et al., 1993]:

- missing or uncertain information about the environment and/or the enemy
- temporal constraints defined by the mission or the enemy
- the unpredictable and adversarial nature of the environment

Behavioral control in the presence of these problems has been extensively studied in the domain of autonomous robots [Gat, 1990; Gat, 1992; Wilcox et al., 1992].

Fortunately, solutions to these problems can be applied to the CGF domain [Becket and Badler, 1993]. The approach used in the Corps Battle Simulation (CBS) [Gat et al., 1993] avoids most of the technical obstacles that have arisen in robot control due to the fundamental view of robot control being primarily a planning problem [Gat et al., 1993] (recently this attitude has been changing [Arkin, 1989; Brooks, 1986; Payton, 1986]). CBS assumes a situated environment which is appropriate for the military domain and employs reactive behavior for control. Reactive behavior control is characterized by continuous decision making and improvisation and is now being utilized for robot control as well [Brooks, 1991].

In addition to the differences between reactive and planning behaviors mentioned previously, reactive behavior also differentiates itself from C2 behavior by being decomposed based upon task instead of function. C2 behavior often decomposes into functional modules such as modules for sensing, battlefield state, planning, etc. Reactive behaviors decompose into encapsulated, domain specific behaviors, also known as tasks. Example tasks include occupying positions, reacting to air attack, etc. These simple reactive behaviors are then combined to form more complex behaviors and form the basis for many virtual SAF systems. Reactive behaviors are often used for lower echelon units because they have two distinct advantages: [Gat, 1993]

- Response times are fast because reactive behaviors typically require minimal computational resources due to their specific task.
- Erroneous behavior is less likely since reactive behaviors do not store any information. Agents that store information can produce erroneous results if unexpected changes in the environment occur.

The specific focus that gives reactive behaviors their advantage can also be a disadvantage in that their scope is very limited and thus are not used for C2 behaviors. In reality, commanders must store information on the mission, current capabilities, enemy positions, and terrain.

As previously shown, behaviors for CGF can be applied at the various levels of the military command hierarchy. Most virtual CGF systems are concerned with behavior at the vehicle/infantry level, platoon level, company level, and battalion level. At the lowest level, behaviors are of a reactive nature in that vehicles and infantry simply react to events

(within doctrine) occurring in their environment as though they were not part of any organizational structure, i.e. no coordination or cooperation. The platoon level consists of a small number of lower level units cooperating via control of a platoon leader. Similarly, the company level consists of a small number of platoons cooperating under control of a company commander. There is also a chain of command that is necessary for commander replacement. The behaviors at the company level consist of a mix of the reactive behaviors present at the platoon level and some command and control behavior. The battalion level is also similar to the lower echelons in that the units are under one command and cooperate towards some common goal but many more units are involved each with varying functionalities. The battalion command level permits more complex coordination of subordinate units in that lower echelons can be ordered cooperate on tasks that do not benefit them directly but support different units. At this level command and control is the form of behavior required.

The Interactive Tactical Environment Management System (ITEMS) is an example of a CGF system that supports up to the battalion level [Siksik, 1993]. The behavior representation consists of a forward chaining expert system with rules describing military doctrine at the various echelons. The doctrine is divided into doctrine pertaining to the mission (goals, route-planning, contingencies, etc.), prime opponent selection, air combat (maneuver, weapons) and command and control. ITEMS is centered around the player concept which may represent any combat entity such as tanks, trucks, aircraft, SAM installations, etc. Individual players are also assigned command roles. Each player's

knowledge is encapsulated in an object-oriented fashion within frames. Included in each player's frame is not only the doctrine mentioned previously but reactionary behaviors concerned with opponents and other environmental stimuli. Parameters for the rules include player inventories, under attack status information, opponent detection, and weapon status information [Siksik, 1993]. At the company level parameters include reports/orders received/sent, threat positions and status information, company damage and weapon status, company position, and mission status. More complex behaviors can result such as withdrawing, alternate route planning, evasive maneuvers, and fire positions. The battalion level uses similar parameters as the company (company positions, company status, battalion order) and has similar behavioral outcomes (coordinated movement/attacks and mission control). The coordination and cooperation task however is more complex. Siksik [1993] describes a typical example of battalion operations: a hasty attack. The battalion commander assigns individual companies to be fire-base units, maneuver units, and directs the fire-base companies to move to a fixed location relative to the enemy and suppress enemy fire while the maneuver units outflank the enemy.

Knowledge within ITEMS is stored in a relational database management system (DBMS). This database is composed of a rules database and tactical scenario libraries. A common database is also used to represent the current state of the scenario. This is used like a blackboard to allow the communication of knowledge between the various players and control functions of the simulation.

Intelligent Agents

The blackboard paradigm [Engelmore and Morgan, 1988] is an opportunistic process commonly used for diagnosis but plays an important role in many simulation behavior systems. The architecture is composed of three primary components: the blackboard, a set of knowledge sources (usually agents), and a blackboard controller. Each knowledge source is some form of opportunistic intelligent agent which knows about some restricted portion of the domain and thus can solve some "subproblem" independently from other agents (at the appropriate time during the problem-solving process, hence the term opportunistic). Together, these agents can "cooperate" to solve some larger problem. The agents may be expressed in a hierarchical fashion in which an upper level agent would have general knowledge about several subproblems and would defer the specifics to the appropriate lower level agents termed *specialists* [Gomez and Chandrasekaran, 1981]. When the current situation matches that which an agent knows about, it can contribute its knowledge to the blackboard. The blackboard serves as the medium by which the knowledge sources output their respective behaviors. The intelligent agents can obtain the current problem (or situation) from the blackboard and add or modify information on the blackboard to record the results of their reasoning. The agents can be heterogeneous in the sense that varying knowledge representation and reasoning schemes can be employed at various abstraction levels; from rules, semantic networks, scripts, frames, and case-based reasoning to more algorithmic and probabilistic schemes (a new research approach even uses cost functions for each agent whose

optimality determines behavior [Ge et al., 1995]). The current problem on the blackboard consists all known facts, partial solutions, hypotheses, etc. that can be used to identify the current problem solving state. This information may also be represented hierarchically so that different levels can use different agents and thus different representations of the problem. The output from one level serves as the input to the next. The blackboard input and/or modifications serve as events that drive the other intelligent agents in the system (under the appropriate context). The blackboard serves as the communication medium between the agents but in large scale systems can be a source of bottleneck. The blackboard controller can perform various functions from deciding which agents can contribute to the solution to deciding which resulting action to take. In this sense, the controller allows the agents to “cooperate” with the other agents to form behaviors. The controller has a set of complex behavioral goals that allow this control and cooperation [Gonzalez and Dankel, 1993]. Some goals include action determination, feasibility of actions, general versus specific actions, effects of problem-solving actions, changes of the problem due to solving steps, the use of global context information, and problem-solving actions versus actions to control the problem-solving.

Blackboard architectures have several features that are useful for generating CGF behavior. The blackboard paradigm provides both the event and context driven programming necessary to represent reactive and mission behavior, respectively. It also allows the integration of both object and functional decomposition for proper structuring of behavior. Each agent in the system can use the structure that most fits its specialty and

the reasoning process most appropriate for solving its subproblem at the proper level of abstraction. This is very important for CGF since most CGF solutions require a combination of algorithmic and heuristic tasks. Algorithmic tasks include dead reckoning, route planning, cover and concealment, line-of-sight calculation, etc. Non-algorithmic tasks include mission planning, situational assessment, decision planning, etc. Some of these tasks benefit from a knowledge-based system approach while others a procedural approach. A blackboard can integrate these different forms in a cooperative manner to produce realistic CGF behavior.

[Braudaway, 1993] describes a generalized combat model that provides the ability to simulate the group collective behavior and individual reactive behavior (local terrain and battle conditions) necessary for CGF. Under this model, the intelligent agents must perceive the environment, maintain a constantly changing model of the current tactical situation, plan actions, perform situational assessment, react to the changing environment, communicate and coordinate with other agents, and simulate the physical model of combatants such as tanks, infantry, air support, etc. In order to behave in a context-oriented and reactive fashion, a hierarchical model (from planning to entity movement) of behavior must be defined in terms of not only the kind of agent but also the battlefield operations these agents can perform. The blackboard paradigm supports this model. This model is also significant in that it reflects the natural operation of the military command structure. Orders are disseminated down the echelon hierarchies from the command levels to the cooperative entities. These orders are decomposed at each echelon level into tasks

appropriate for that level and at an abstraction level appropriate for the echelon. This combination of echelon behaviors and same level sub-behaviors produces the behavior defined at the echelon level that originated the order. The knowledge sources that the intelligent agents use follow the same military hierarchy. The knowledge sources are filtered based upon the echelon level of the entity (whether information can actually be perceived, i.e. "fair fight") into the simulation supervisor level, platform command level (battalions, companies, etc.), and platoon command level [Braudaway, 1993]. The supervisor level contains information about the status of the battlefield and the combatants, order information, and information filters. The platform level is the lowest level in the command hierarchy and contains information about the status of each controlled subplatform and information on how to control the platform itself as a single unit. Finally, the platoon level, contains information about the platoon's status, organizational structure, current order, and how to execute the doctrinal "primitive" behaviors such as assault, passage of lines, actions on contact, etc. The analysis and results in [Braudaway, 1993] suggest that the blackboard architecture is appropriate for representing the contextual command, control, and communication (C3) and reactive behavior necessary for units in the military domain.

Command and Control Behavior. Blackboard architectures are typically used to represent command and control behavior. Most C2 behavior is limited to the company level and below where typical behaviors consist of locating battle positions, planning routes, and engaging targets. At upper echelons, coordination and cooperation of units becomes a

complex problem that is not well suited to the lower echelon rules and finite state machine representations. Different combinations of events, changing contexts, and event sequences produce a combinatorial explosion of control possibilities that predefined control strategies such as finite state machines or rules cannot handle in an efficient manner [Braudaway, 1993]. Additionally, these strategies are static in that they cannot account for all the possible situations that can occur on the battlefield nor can they adapt to include them. At a minimum, C2 representations for upper echelons must include some representation of the battle plan and control measures which can be interpreted by the CGF command units, and a report and order structure for tactical state information and decision communication. Additionally and most importantly is the requirement for decision making capability that allows the upper echelon units to correctly and easily respond to battlefield events and adaptively respond within the given context of the battlefield and mission. There has been much research in the use of intelligent agents (usually synonymous with a blackboard architecture) for behavior and command and control, all of which are variations of the traditional theme [Pratt et al., 1994; Gagne, 1995; Nielsen, 1995; Rosenbloom et al., 1994; Jones et al., 1993; Kuokka, 1993; Harmon et al., 1994; Ge et al., 1995; Cox et al., 1994; Wittig, 1993; Chaib-draa et al., 1993; Becket et al., 1993].

The Judgmental METT-T (JMETT-T) project is one example of an intelligent agent system designed to perform C2 decisions based upon the military concept of METT-T [Bimson et al., 1995]. METT-T is a military method of decision making based upon

the analysis of the Mission, Enemy, Terrain, Troops, and Time. Decisions are made based upon the current situation and how it affects the mission, the locations and intentions of the enemy, the terrain constraints, cover and concealment positions, the status of the commander's troops and equipment, and any time constraints given by the mission. After the situation and mission are analyzed, potential courses of action (COAs) are analyzed, one is chosen and revised orders are sent. Unlike reactive behaviors, this judgmental behavior looks at the "big picture" instead of just the local situation.

JMETT-T uses an approach similar to a blackboard architecture to produce the judgmental behaviors necessary for SAF automation (increased CGF). The blackboard is represented by two different representations, a semantic network, and tactical map. These two representations provide two different views of the battlefield that are necessary for the tasks performed by the different agents. These two views are updated by reports, orders, and sensor data. The semantic network stores relationships among the various objects on the battlefield such as terrain objects occupied by enemy troops, battle positions to be occupied, friendly units and their organization, the mission execution matrix, etc. [Bimson et al., 1995] The semantic network stores declarative knowledge about the changing battlefield in a form that is easy to access and update. The network will continue to grow as new relationships and objects are added thus increasing the command entity's (CE) knowledge of the battlefield. The semantic network is appropriate for representing conceptual information but not spatial information. The tactical map stores geographical information about terrain, positions of obstacles, positions of units, etc. in a two-

dimensional grid. The cells may contain two-way pointers from map objects into the semantic network, thus establishing the unit's position relative to terrain features, enemy forces, friendly forces, etc. The map aggregates information on terrain trafficability, enemy threat, and cover and concealment into a 3-tuple value vector. This information is used by the terrain intelligent agent to perform intelligent route planning [Ourston, 1995].

The CE is the corresponding blackboard controller of the blackboard paradigm. The CE arbitrates agent suggestions and determines the course of action. The JMETT-T system is composed of a set of intelligent agents (IA) that correspond to each part of the METT-T analysis: a Mission IA, Enemy IA, Terrain IA, Troops IA, and Time IA. The CE is represented by several subagents: the alternative generator, course of action selector, consistency checker, and commander's vehicle. As in the traditional blackboard paradigm, each IA operates independently of one another, communicating via changes to the semantic network that another agent monitors. Unlike traditional blackboards, the agents can communicate directly if a specific service (such as a course of action) is required by another agent.

The Mission IA is concerned with the battlefield situation as it affects the mission. Changes to the mission may be warranted due to enemy presence, loss of equipment etc. The Mission IA may also change the mission of lower echelon units to complete the overall mission. The mission is simply a plan and it uses a goal-directed approach to achieve it. The Mission IA uses the concept of a subjunctive goal network that has been used successfully in the SOAR tactical air simulation [Jones et al, 1994]. Each node in the

network represents a goal or subgoal that needs to be accomplished to accomplish the mission. Horizontal arcs connect nodes that have a temporal relationship, i.e. one goal has to be accomplished before another. Vertical arcs represent a hierarchical relationship similar to the horizontal arcs. Instead of specifying a priority for goal acquisition, vertical arcs specify the subgoals that must be accomplished in order for a higher level goal to be accomplished. Each node has a set of dependency links into the semantic network and conditions on this dependency such as the size of an enemy object [Bimson et al., 1995]. If the size dependency cannot be met (a force mismatch), then one of the goals may not be able to be accomplished which in turn may cascade up the goal network causing the failure of higher level goals. Rules are used to control the operation of the goal network and services are used to return delays, mission critical goals, and goals that are in danger.

The Terrain IA maintains information about the condition of terrain objects (bridge out, for example). Using inductive rules that match (in a forward-chaining manner) on elements in the CE's knowledge base, the Terrain IA can identify obstacles, routes, and other terrain information that may be needed over the course of a mission.

The Enemy IA maintains information received via intelligence and spot reports about enemy positions, size, strength, and possible enemy intentions. This information is used by the CE when considering possible courses of action. Rules based upon enemy doctrine can provide guesses about the enemy's possible moves and how these affect the goals of the mission.

The Troops IA and Time IA are the most simplistic. The Troops IA provides information on the status of personnel and equipment and how these may affect the goals of the mission and courses of action. Similarly, the Time IA provides information on conflicts between the time required to execute a course of action and the time constraints on the mission.

The Alternative Generator is a component of the CE IA that provides alternative courses of actions for problem situations that may arise during the course of a mission. Examples include loss of critical equipment and obstacles. Case-Based Reasoning (CBR) is used to represent and choose previous solutions to situations similar to the current problem situation [Bimson et al., 1994]. Cases are matched on two levels: the general and specific. A problem situation is first matched based upon its general class such as an obstacle. Next, the choices are refined by matching upon its specific type and echelon required to handle the problem. For example, a general obstacle problem situation characterized by a bridge-out and platoon-level origin would return possible solutions as reroute and breach. Using the echelon type, the case base can be used at any echelon level and problems that cannot be handled at one level can be passed up the chain of command to higher levels for a solution [Bimson et al., 1994]. In the bridge-out example, the problem could be passed to a higher echelon where a bridge platoon solution could be used.

The Course of Action Generator receives solution alternatives from the Alternative Generator to select a specific course of action. It calls upon the other IAs to help

discriminate the choices with respect to material and personnel resources required and time constraints. Feasible alternatives are then further discriminated in a specific to general fashion based upon information about the enemy available from the Enemy IA, and impacts on the mission from the Mission IA. If no solution is found, the problem is sent to the next highest echelon. The COA generator looks at the different trade-offs available and attempts to select the best one. This is the focus of the JMETT-T effort in that it demonstrates how complex behaviors and decision making can be produced out of the interaction of several less sophisticated intelligent agents.

As seen with the Mission IA, any behavior system that plans to implement upper echelon CGF behaviors must include a planning component to automatically generate plans for a CGF or SAF. Planning is a classical AI problem that, with the advent of planning domains like CGF, has not only become more difficult but has been approached differently than in classical techniques. Classical systems like SIPE [Wilkins, 1984] typically viewed planning as being independent from execution and assumed only one active agent at one time. This gives the planner the ability to capture the complete state of the world and the all the cause-and-effect relationships. Unfortunately, this is an unrealistic approach for most real-world domains such as CGF. With the addition of multiple agents in the environment and uncertain information about the state of the world, the proper execution of a plan is no longer guaranteed. Other agents' actions cannot be guaranteed, so the complexity increases dramatically as more agents are added as in CGF.

Mission Planning. New approaches to planning incorporate two phases: plan generation and plan selection. During plan generation, a set of candidate plans is generated based upon some initial constraints representing the current situation. Traditional AI techniques such as rules and case-based reasoning can be used to select candidate plans. Plan selection, involves choosing a priority on the remaining constraints, and choosing the plan that best meets those constraints. Each phase typically has the property of using temporal projection, a method of projecting actions into the future to predict what the results will be. Temporal projection is used to select the plan by using some evaluation function which measures the properties of the plan such as the probability of success and loss of resources.

One approach to plan selection for CGF is to use simulation within the simulation, i.e. simulate the execution of the plan within the simulation environment and choose the one with the best results [Lee and Fishwick, 1994]. Simulation training itself is a form of simulation planning. For CGF, simulation can be taken one step further, simulation within the context of the simulation can be used for planning. Simulations use some form of model to represent reality. The use of these models to formulate sequences of actions is central to planning since given this sequence, the model can be used to simulate the future [Dean and Wellman, 1991]. The simulation suggests modifications to the plan and in this way can be a useful tool for evaluating hypotheses.

[Lee and Fishwick, 1994] use a system composed of a terrain analyzer, world database, reactive module and planning module. The terrain analyzer can give routes,

tactical positions, terrain features, and line-of-sight determinations that satisfy given constraints. The world database keeps track of the state of the simulation such as locations of enemy and friendly units. These are used by the reactive and planning modules to derive behavior. The planner uses the reactive module initially and when the necessary conditions (triggers) arise new reactive behaviors can be initiated or the planning module can be activated for replanning. However, the planning module will be used initially when an order from an upper echelon is received. In either case, data from the world database and terrain analyzer are used to produce efficient plans for CGF. The planner can also generate orders for any subordinate units it commands.

The planning module contains several components necessary for simulation-based planning: a situation analyzer (SA), course of action (COA) generator, course of action simulator, and course of action evaluator. The SA is simply a collection of rules that fire on the data given in the world database and queries of the terrain analyzer to determine the current tactical situation. [Lee and Fishwick, 1994] focus on route planning so the SA provides various alternate routes for the situation. These alternate routes are represented in the form of a decision tree in which the branches determine different segments of the route(s) to take and which portions of the unit (company, platoon, section) follow which segment. Every possible combination of alternative choice or action is represented in the tree. This tree is created by the COA generator when given the alternate routes. Using another expert system, portions of the tree will be pruned. The remaining tree is sent to the COA simulator. The simulation is performed at a higher level of abstraction with units

limited in their intelligence and planning capabilities. A turn based approach is used in which friendly and enemy units receive their own time slice in which they can perform the primitive actions move, shoot, communicate, and observe. The system has to be able to do many small simulations as fast as possible in order to make a decision. The COA evaluator uses rules to measure the success or failure of the individual simulations based upon criteria such as the success of the mission, number of casualties and loss of equipment. This simulation-based planning approach has been added to a version of ModSAF in which the planner is represented as another behavior available to the units [Karr et al., 1995]

Simulation-based planning offers several advantages over other planning schemes [Lee and Fishwick, 1994]:

- It provides a consistent and uniform method for evaluating plans since simulation itself is uniform and consistent. Many other planning solutions use adhoc methods for determining the goodness of plans.
- Since distributed simulation is used as the planning method, simulation-based planning is also scalable to increasing number of entities on the battlefield.
- The simulation can easily play back the results of the COA simulator and COA evaluator to explain the plan selected.

Depending upon the computing power available, simulation can be used at all levels of planning at higher levels of abstraction. Practical compromises include using lower levels of abstraction for simulation or using a multi-model approach such as combining rule-based and simulation-based elements [Lee et al., 1993].

There are many other methods to automate planning. The CBS system [Gat, 1993] uses the Plans-as-Advice approach [Agre, 1990] in which the plan is used only as a resource to make situational decisions. The plan only constrains actions, not dictating them, and allows for the representations of incomplete and missing information since they are viewed only as information. Obviously, the more complete the plan the more efficient the resulting actions. The CBS system also includes a mechanism to detect and recover from failed plans due to incomplete or missing information. It attaches expectations to the steps of the plan, that when failed, dictate other courses of action. [Gat, 1993] provides more in-depth information on the representation necessary to implement this recovery mechanism.

The RAGE system [Tallis, 1993] is a constructive knowledge-based system designed to support high level C2 planning in an environment containing hostile and friendly reactive subordinate agents. Subordinate agents use search space trees to determine their next actions. The superior unit can control this searching and do planning when exceptions occur. The planning consists of making decisions based upon each subordinate's goals and constraints on those goals.

Robot control in many ways is similar to the CGF behavior problem. Case-based robot navigational planning is very similar to CGF route planning [Vasudevan and Ganesan, 1994]. Routes are chosen based upon previously experienced situations involving factors such as obstacles, terrain, weather, enemy presence, destination, etc.

[Ourston, 1995]. Other approaches to route planning that have CGF analogs can be found in [Warren, 1990; Carroll et al., 1992; Goel and Callentine, 1992; Chen, 1992].

The RPD architecture [Chaib-draa et al., 1993] is a hybrid architecture that combines reactive behavior, planning, deliberation. It uses a variety of problem solving methods from procedural reasoning for linking perception and action, rule-based reasoning for situation recognition in terms of goals, case-based planning for new situations, cognitive maps for beliefs about the state of the environment, and case-based planning for goal and route planning in unfamiliar situations. The approach is centered around the concept of social knowledge, i.e. the group adopts social rules that each agent obeys and assumes other agents will also. This embodies various rules such as coordination rules, cooperative rules, collective rules, and regulations. Coordination rules are rules such as staying in formation and in a specific order. Cooperative rules select behaviors that only make sense if other entities perform the same action such as attacking the same enemy. Collective rules are most easily characterized by the division of forces into separate tasks to support the overall mission. Regulations are doctrinal standards that the behaviors must comply with. The social knowledge approach is most appropriate for the military domain because it is driven by doctrine. The RPD architecture itself is quick to respond to changes in its environment (the reactive component), can plan activities to meet goals (planning component), and reason in the presence of unfamiliar situations taking into account the intentions of other agents in the environment (deliberation). The reactive component is characterized by linkages from environment perception to identification to

action. The planning component is characterized by linkages from environment perception to identification to planning to action. The deliberative component has not been addressed much in the CGF literature. It is concerned with the more complex problem of reasoning in an uncertain or unfamiliar environment in which there may be several ambiguous choices. It is characterized by linkages from environment perception to identification to decision-making to planning to action. The decision-making link must be able to choose between alternative goals, alternative actions, or alternative plans (the plan selection process mentioned previously).

Case-Based Reasoning Behavior. Up to this point most of the focus has been on higher level behaviors such as mission planning. The more common form of behavior as mentioned previously is the reactive behavior. Rules can be used to control reactive behavior like in ITEMS. CBR has also been used as a method of generating CGF behavior. The CAAT project [Keirse et al, 1995] uses cases of specific scenarios to generate behavior for CGF air combat. State variables lie along orthogonal axes in a Decision Space. The combination of unique state variables define a point in the decision space which defines the case. Example state variables include angle off target and range to target. The decision space defines regions such as avoid, pursue, or go home based upon the state variables. To retrieve a case, the current point in the decision space is calculated based upon the current situation's parameters and the case point in the decision space closest to the current one is the retrieved case. The authors state that through careful selection of the state variables that make up the multi-dimensional space, the

decision space can be invariant to rotations and translations thus allowing a particular decision point to map to all possible rotations and translations of a particular vehicle configuration [Keirseay et al., 1995]. This is a very important problem in using CBR for CGF. Cases, by their very nature, are specific, but due to the infinite variations in the environment they must be general enough to apply to a variety of similar situations. [Ram and Santamaria, 1993] use a similar approach they call continuous case-based reasoning for reactive robot navigation. Autonomous robot navigation involves combinations of four concurrent behaviors: goal seeking, obstacle avoidance, wander, and escape [Vasudevan, 1995]. Each behavior recommends changes to the vehicle parameters some of which are competing. Cases are used to select aggregations of these behaviors and their modifications. Also cases are used to switch aggregations of behaviors if the situation changes drastically. The cases are represented as analog vectors, each value representing the value of an input or output parameter at a specific time. Thus, the vector represents the history of this parameter over a given time-window. These vectors are associated with vectors describing the environment to form a case. A similar approach uses fuzzy logic rules that change the system parameters and select the appropriate behavior [Vasudevan, 1995]. Aggregations of these rules form cases. This CBR approach can be scaled up to a multi-agent environment (like the blackboard approach) where local reactive decisions can be coordinated based upon the actions of nearby vehicles. Much research in the reactive behavior area has come from the robot control arena [Arkin, 1989; Brooks, 1986; Payton, 1986]. Cooperative reactive behaviors have

been thoroughly discussed in [Arkin, 1992]. Some of these approaches require inter-agent communication and others do not. Most involve simple local rule-based strategies that, coordinated in a common environment, can yield organized behavior without the central control used in some blackboard systems.

An earlier use of CBR for CGF involved using case-based planning for intelligent agents [Castillo, 1991]. Cases were used in conjunction with scripts [Schank, 1977] and rule-based reasoning to generate intelligent agent behavior. Formal representations are used to represent plan goals, plan constraints, plan adaptation, plan generation, and plan failures. Each case is represented in terms of the goals they satisfy and problems they avoid. Rules are used to generate new cases if no closely related plans can be found. The case base itself is represented as a discrimination network with multiple entry points and three types of relations: logical (specialization ontology), structural (decomposition), and causal. This CBR approach removes the infinite variation problem by assuming all the cases and the inputs are presented in terms of abstract goals written in the form of a script.

Context-Based Reasoning Behavior. Most of the problems with using CBR for reactive behaviors arises from two of the properties inherent in CBR. One, the cases are designed to be specific, and two, a large case-base is needed for realistic behavior. Another approach, Context-Based Reasoning (CxBR) helps overcome these deficiencies. CxBR [Gonzalez and Ahlers, 1995] uses the concept of scripts [Schank, 1977] to provide tactical knowledge to intelligent agents on the battlefield. These scripts provide information necessary to perform situational awareness and the resulting actions (which may include

switching to another context). The use of contexts is based upon the hypothesis that tactical experts use only the relevant information necessary for the task at hand, there are limited number of events that occur under a given situation, and that the presence of a new situation requires a change in the course of action [Gonzalez and Ahlers, 1995]. The key point in context-based reasoning is that associating events and actions under a specific context eases the identification of such contexts and makes the behavior execution more efficient. In the domain of submarine warfare (where this work was developed) this is certainly the case. Only certain subsets of all possible situations are applicable under certain contexts. The contexts are more general than cases in CBR and can apply to a wider range of situations. Because the contexts are more general, the context base is usually much smaller than the case base. The script for a context supplies the steps required to carry out a specific action or recognize a new situation. Context-based reasoning can be used to define behavior at the mission level (mission contexts) and at the task level. Since the domain is narrow and specific, context-based reasoning can be used to generate realistic behavior for training purposes. Contexts also include any special instructions and constraints that must be satisfied throughout the execution of the context.

Reactive Behavior Architectures. Finite state machines (FSM) or finite state automata (FSA) and their variants are the most common representation of behavior for virtual simulations because of their simplicity. FSMs consist of a finite set of states, a set of state variables, the current state, and a transition function that maps from the set of states and current state to another state. If the FSM is augmented there may be a set of actions

associated with each transition. Changes to the state variables can occur synchronously or asynchronously for either event driven or time stepped simulations. States can be considered a snapshot of the current behavior at the current time interval. Transitions can be considered directed transition arrows that allow movement from one state to another or to the same state if appropriate. Predicates on each transition must be met in order for the transition to occur. When the FSM state needs to be updated (via event or time step) each predicate is evaluated in order on the updated state variables and the first one that evaluates to true is taken. Next, the current state is set to the new state that the transition points to and its behavior executed. FSMs provide a simple representation for representing the sequential behavior common to reactive CGF behaviors. CGF behavior states are adjusted based upon events entered into the machine and the results of actions performed by the machine. Refer to [Fishwick, 1995] for a more in depth discussion of the use of FSAs and other forms of declarative modeling (state-event graphs, Petri nets, finite event automata) for simulation modeling. The major disadvantage of FSMs is that they become impractically complex for large problems, requiring large numbers of states and transitions [Rumbaugh, 1991] and are ill-suited for some of the more complex C2 behaviors.

Several augmented forms of FSMs have been used in the CGF domain in an attempt to overcome the inefficiency of traditional FSMs. One such form is the Hierarchical Finite State Machine (HFSM), an object-oriented FSM representation. Object-oriented modeling offers several advantages for behavior representation.

Behaviors can be recursively decomposed with finer levels of granularity at each level. More abstract behaviors are represented at higher levels, with specific behaviors at the lower levels. HFSMs models the traditional FSM states as objects. These states may contain state machines, algorithmic code, a rule-based system, procedural code, or any other mechanism needed to support the desired behavior. The behavior mechanism is fully encapsulated within the state and does not persist when the state is exited. HFSMs directly support the aggregation and decomposition of behavior, the traditional way tactical behaviors are considered. HFSMs offer distinct advantages in that they partition behavior into general and specific components and remove redundancies found in the traditional, flat FSM model. The HFSM states can also be concurrent to provide background behavior or concurrent behaviors for lower echelons. The use of HFSMs has been recommended as the representation of dynamic behaviors because of its less complex and more natural representation [Rumbaugh, 1991; Schlaer and Mellor, 1990]. The SimCore simulation framework [Aronson, 1994] allows the specification of CGF tactics and uses HFSMs for this specification. The use of HFSMs complies with the SimCore goal of providing a representation that is both extensible in breadth and depth in order to avoid the limits on the scope and fidelity of CGF behaviors [Aronson, 1994]. In SimCore, monitor variables are used to represent the view of the world and thus the state of the machine. These are used to determine tactical state transitions and tactical actions. The monitor variables form the communication interface between other entities. Other entities such as agents, sensors, or PDUs set values of the machine's monitor variables and thus

affect its behavior. Similarly, output communication is done through output monitor variables.

The Asynchronous Augmented Finite State Machine (AAFSM) is another FSM variant developed for ModSAF [Loral, 1995], the virtual SAF simulation used for experimentation in this research (the other virtual simulation used in this research is the Close Combat Tactical Trainer (CCTT) and it also uses FSMs for behavior generation but does not comply with any one strict representation. For more on ModSAF and CCTT, see Chapter 5 - Evaluation Prototype. The AAFSM is asynchronous because it responds to external events, not necessarily a fixed time step (which is also allowed). The traditional FSM is further augmented with many variables and functions that work with system variables and data that is not part of the FSM itself. A common example includes services to retrieve information about entities from the ModSAF Persistent Object database. In other words, the FSM portion of the AAFSM contains the traditional behavior control logic while the augmentations provides the behavioral embodiment and interfaces. Like HFSMs, recursiveness into and concurrency of sub-behaviors is also allowed. The AAFSM is more powerful in this regard in that it can not only spawn new sub-tasks, but stop or suspend them at any time and combine them with other subtasks to form new behavior combinations. Additionally spawned tasks can be combined with those already executing and these tasks can stop their parent task if necessary. The AAFSMs in ModSAF also monitor their original input variables continuously and can restart at the start state if key parameters like a route are changed in the middle of behavior execution.

The FSM behavior approach can be considered forward reasoning since the decision path traverses from some initial point to some goal, the objective of the behavior. However, a backward approach can be used that recurses from the behavior goals into simpler sub-goals and finally to some primitive goals. This approach may seem “backward” from the traditional way tactics are written in the field manuals. In field manuals the approach seems to be time-line based but it is really based upon prioritized goals. Examples include, when moving along a route, seek cover and concealment. This is really a sub-goal of the movement goal. In order for a backward approach to work, the domain must be well-known and organized. This is the case with military doctrine. It is one of the most highly optimized man-made systems and has had hundreds of years of military experience to draw upon when organizing tactics and behaviors [Kwak, 1995].

The Rational Behavior Model (RBM) [Kwak, 1995] is a backward reasoning model that is goal-directed like described above. It is a multi-level architecture composed of Strategic, Tactical, and Execution levels. The behavioral logic is contained in the Strategic level. The behavior goals and sub-goals are organized in this level as an AND/OR goal tree. This tree governs the control of the Tactical level. The Tactical level provides the embodiment of behaviors (like the augmentations of the AAFSM) by maintaining the attributes of the system and world view. The Tactical level also encapsulates the representations of internal behaviors, providing the recursive behavior abstraction provided by AAFSMs and HFSMs. The interface to the rest of the system is provided in the Execution level. The Rational Behavior Model has the distinct advantage

that while FSMs become clumsy at higher echelons, the Rational Behavior Model does not. It uses similar approaches to many planning components of upper echelon C2 systems and thus scales well. As future CGF behaviors become more complex and limitations of current approaches are reached, the RBM model may receive more attention and possibly lead to a canonical behavior form for all echelon behaviors. More on AAFSMs and RBMs and their comparisons can be found in [Kwak, 1995].

Interoperable Linkages

Constructive-Virtual Linkages

There is a desire within the military to allow constructive and virtual simulations (C-V linkage) to work together in a common environment to provide more robust training environments at different levels of fidelity. Fine-grained vignettes can be used for soldier training while lower fidelity areas can be used for upper echelon training. Under this multi-modeling concept, lengthy portions of engagements can be executed under a faster-than-real-time constructive system while battle vignettes can be executed by virtual systems to not only train upper level commanders but expose key strengths and weaknesses of the system, doctrine, or force structure being analyzed [Pickett and Petty, 1995]. Additionally, the penalty of high fidelity is kept only to those areas of interest while still maintaining a robust overall scenario.

One of the key problems with multi-modeling is the seamless connection. This is no exception for CV linkages. Unfortunately, as the difference between the overall fidelity of simulations increases so does the difficulty of interoperability. Similarly, the

interoperability difficulty increases as the fidelity of the simulations involved increases. This will become critical when discussing virtual-to-virtual linkages. CV linkages have the most extreme fidelity differences making this interoperability a crucial one. Simulations often sacrifice the fidelity of one environment to accurately reproduce another. Constructive and virtual simulations both have this property because constructive simulations sacrifice detail for breadth of operations and virtual simulations sacrifice breadth for detail. Discrepancies between the two modeling domains can severely effect the training value and skew the value of the exercise to the trainees when linked together. CV linkages suffer from "fair fight" and time/space coherence interoperability. "Fair fight" interoperability deals with the outcomes of one simulation being correlated with appropriate outcomes in the other. Thus, battlefield elements can engage one another without regard to the simulation that controls the elements. Since training in a heterogeneous environment is inherently more difficult than in a homogeneous one, some effort must be taken to ensure interoperability. Implementational or functional differences between interacting battlefield elements can yield unrealistic advantages for some while unrealistic deficiencies for others. Certain essential capabilities must be defined for all simulated modes to allow them to operate together fairly. Time/space correlation interoperability involves making sure that time is synchronized between differently simulated units so the outcomes are fair and realistic for all units. This can be a serious problem when dealing with CV linkages since constructive simulations usually execute faster than real time and virtual simulations do not. Most CV linkage research being done

at present is concerned with the aggregation/disaggregation of units problem that occurs when crossing Constructive-Virtual boundaries. [Stober et al., 1995] presents a survey of projects that fall under this definition of CV linkage.

The Eagle/BDS-D linkage [Franceschini and Karr, 1994; Karr and Root, 1994] involves Eagle, a corps/division level constructive combat model, that simulates ground combat only at the company and division level, and a DIS/SIMNET virtual environment using the IST CGF [Smith and Karr, 1992] testbed. The testbed provides a mechanism for testing CGF control algorithms. The issues that the Eagle/BDS-D linkage dealt with was:

- the aggregation/disaggregation of constructive units.
- the synchronization of the constructive time-stepping with the real-time clock of the virtual simulation.
- conversion of terrain coordinates.
- pseudo-disaggregation for representing aggregate units on virtual terrain.

Eagle simulated all aggregate units. These units were disaggregated when they moved into a high fidelity area (virtual area). The disaggregated unit would be send an operations order and be executed by the CGF testbed. Upon aggregation this process is reversed. When a unit is disaggregated, the CGF testbed would break the unit down into vehicles and place them according to a vehicle placement algorithm [Franceschini and Karr, 1994]. This constructive-to-virtual linkage uses special DIS PDUs (interoperability protocol) to control and support disaggregation/aggregation. A special appearance PDU is used to

pseudo-disaggregate a unit for appearance purposes on the virtual terrain so the unit can be seen properly by virtual units. Finally, combat was limited to indirect fire between constructive and virtual units. Direct fire was not supported due to the timing and database correlation problems [Stober et al., 1995].

The Corps Level CGF (CLCGF) research was undertaken to examine theater level exercises [Calder et al., 1995]. It linked Eagle to ModSAF. This linkage dealt with the same issues as the Eagle/BDS-D linkage. Like Eagle/BDS-D, the CLCGF used special Disaggregation/Aggregation PDUs. However, unlike Eagle/BDS-D, the Disaggregate Request PDU must be resent periodically or the unit will automatically be re-aggregated.

The BBS/DIS linkage's goal was to integrate the Brigade/Battalion Battle Simulation (BBS) with a DIS network [Hardy and Healy, 1994] SIMNET simulators were used to represent DIS entities since, at the time, there were not DIS CGF simulators. Again, this linkage dealt with the same issues and problems as the previously mentioned linkages, only involving different simulations.

Like Eagle/BDS-D, the Janus/BDS-D linkage's goal was to provide a validated simulated force for armor scenarios [Pratt et al., 1994] This research effort is different from the others in that the two simulations complement one another. Some of the linkage problems still exist but they are not the focus of their effort. The Janus scenarios have been developed and accepted over many years. Now, these scenarios can be brought into DIS to provide CGFs without using extensive computer resources. The DIS portion provides Janus with man-in-the-loop capability and reactive behavior [Pratt et al., 1994]

The Janus linkage involved many changes to the system to make it DIS-compatible. Analytic algorithms (night vision detection, hit/kill probabilities, aircraft play, dead reckoning, engagement arbitration) had to be changed to meet the DIS world. Filters had to be developed to translate or extrapolate data to and from DIS. This data includes DIS PDUs, terrain data (visual terrain databases versus terrain files), coordinates, enumerations, velocity vectors, etc.

Whatever the linkage, constructive-to-virtual linkages still have not solved the issues of Combat Results Correlation Error (outcome of battles should be the same regardless of the representation of the units) [STRICOM, 1994], Spreading Disaggregation (domino effect for disaggregation of units close to virtual units) [Petty and Franceschini, 1995], Unit Formation on Disaggregation [Stober et al., 1995], and Direct/Indirect Fire between constructive and virtual units [Stober et al., 1995]. The Aggregation/Disaggregation problem and its PDU(s) representation is still a highly researched topic as shown in [Petty and Franceschini, 1995; Franceschini et al., 1994; Clark et al., 1994; Cox et al., 1995; Foss et al., 1995; Generazio et al., 1995].

Smith demonstrates through an example involving three units of different representation, the problems that can occur between constructive and virtual simulations [Smith, 1995]. One unit, Unit A, is a pure constructive aggregate unit that uses a single location and orientation. Damage and strength is based upon the unit as a whole and fuel supply is not considered. Unit B is still an aggregate unit, but its vehicles are assigned offset locations from the center of mass, and fuel consumption is distributed to the

vehicles. Unit C is a pure virtual unit, where each vehicle has its own unique location and moves and consumes supplies independent of the unit it belongs to. Damage and strength is based upon the individual vehicles. Individual vehicles also may experience fatigue and loss of morale if they are representing manned simulators. The location of the unit may be based upon the lead vehicle or a center of mass (centroid) calculation.

When these representations meet on the synthetic battlefield, several problems can arise. When Unit A fights it can continue until it has 100% damage without regard to individual unit damage (damage is distributed equally), their bearing on the enemy, or fuel consumption. This gives it a distinct advantage over the other units which must worry about bringing their individual vehicles to bear, supplying them, and keeping them undamaged (its equipment will be more damaged, since a disproportionate amount is directed towards a smaller amount of equipment). Thus two units of equal strength, one aggregated and one not, will not be equal and not experience a fair fight. With Unit C, it must also deal with fatigue and morale degradation due to combat losses and combat duration which tilt the balance even more in Unit A's favor. Unit C, also has to spend more effort on bringing vehicles to bear and moving them into advantageous positions not allowed with Unit B. Unfortunately, this has no bearing on conflicts with Unit A. Additionally, Unit C may use electronic warfare which has no effect on Unit A. With additional fidelity (chemical/biological/nuclear warfare, dynamic combat groups, logistics, intelligence, communications saturation, weather, etc.) the discrepancies between the units is magnified. Smith states that any discrepancies in the either the virtual or constructive

domain models will effect the training environment and skew the results in favor of representation that is more basic [Smith, 1995].

In an effort to overcome the constructive-virtual problem and reduce the redundancy is solving the same problems for different combinations of simulations, standard interoperability architectures are being developed. An early architecture was the IRIS architecture developed at the Naval Air Warfare Center [Kazarian and Shultz, 1994]. The goal of the project was to provide portable components that can be used to adapt existing constructive, live, and virtual simulations to DIS with the least modification. The IRIS focus, like the Janus/BDS-D linkage, was at the DIS level. The IRIS core consists of a Simulation Interface Unit (SIU) that is composed of a simulation specific and simulation independent component. The Simulation Specific Component (SSC) must be written for each simulation that will use the system. The independent component is meant to be reused each time. Since the IRIS effort is more on a DIS level, these components are more concerned with sending, receiving, filtering, and translating PDUs, and isolating the simulation from these DIS protocols. Depending upon the simulation connected, the SIU will need "intelligence" to allow the simulation to be an equal participant in the exercise [Kazarian and Shultz, 1994]. The IRIS architecture serves mainly as an architecture to bring constructive and live simulations into DIS. Their research has not addressed inter-entity communication and interaction, or the aggregation/disaggregation problem. They recognize some of the issues such as fidelity differences, but offer no solutions. One

interesting aspect, however, is their work on interfacing live training into the DIS environment.

The ACTORs architecture uses knowledge-based modules to adapt results from all Battlefield Operating Systems (BOS) to the appropriate training level whether it be theater, battalion, company, platoon, etc. [Mastaglio et al., 1993]. All simulations components interact at the entity level and their behavioral actions are filter/modified for presentation to other ACTORS in the system. Also, the entities themselves may be adjusted. For example, an ACTOR can adjust units for fatigue or posture, thus reducing its effectiveness. Entity level actions, via ACTORs, can be generated from intelligent agents, manned simulators, tactical engagement simulations, or constructive simulations. The architecture also provides “portals” into these systems by which soldiers can participate at the various levels. Thus, there are ACTOR systems at every level of simulation from a brigade commander, to his/her staff, all the way down to the men in the vehicles. The ACTORs themselves also allow the simulation to increase proficiency over the course of the exercise using machine learning techniques [Mastaglio et al., 1993]. However, it should be noted that the architecture is only conceptual in nature and should support seamless integration of different vertical and horizontal simulations. The architecture has not been implemented and is primarily concerned with integrating new simulations, not existing ones.

SPARTA has developed a different architecture to support the integration of simulations [Bartel et al., 1994]. They use an object-oriented approach to develop

“teams” that are hierarchically composed of object components and can interact with other “players” via UNIX sockets. The architecture supports varying levels of fidelity and the plug-n-play concept. With the object-oriented approach, players can be developed at any level of fidelity and plugged into an existing team. A simulation controller is used for temporal arbitration to schedule events for the teams and coordinate their time steps via its master clock. A team controller is used to create, initialize, and update players on a team. As mentioned previously, the architecture uses sockets for communication between the teams and the controller and every other team allowing one-to-one (point-to-point), one-to-many (broadcast and multicast), many-to-many, and many-to-one communication between entities. Like the ACTORs architecture above, the SPARTA system is designed more for simulation construction than simulation interoperability.

Constructive-Constructive Linkages

Not only do CV linkages experience interoperability problems. Interoperability problems may exist when trying to integrate two complementary constructive simulations. For instance, the integration of CBS and TACSIM [Smith, 1995] illustrates the problems that can occur with simulations designed for different aspects of training. Both are validated simulations but have several discontinuities when integrated. They both have a common communications protocol but this does not overcome the lack of a shared modeling framework. CBS, as mentioned previously, is a typical constructive simulation with aggregate units. The aggregate units are assigned a single location and move over aggregated terrain. TACSIM, on the other hand, is a simulation that supports

the collection and distribution of intelligence about the posture and intention of enemy forces. TACSIM relies heavily on the deployment of individual pieces of equipment in a unit to not only identify the unit but determine its activity. When integrated with CBS, TACSIM adds a doctrinal deployment pattern to the aggregate unit. This pattern consists of an assignment of a unique location to every piece of equipment relative to the position of the aggregated unit. This assignment is based upon several factors such as the unit's type, size, activity, and operational characteristics. This deployment pattern is then reflected in intelligence reports sent to CBS.

Unfortunately, the deployment pattern is not supported by CBS. Thus, when the deployment pattern is used to provide targets for artillery there is a problem since the individual piece locations are not locations associated with any CBS aggregate unit. Artillery fire upon the unit in question may be a complete miss even if it hits the aggregate unit exactly because of the dispersal of equipment in the deployment pattern. This illustrates the main problem of interoperability, whether it be constructive-to-constructive, constructive-to-live, constructive-to-constructive, virtual-to-live, or virtual-to-virtual; the problem of using different simulation models to represent the same type of entities, events, behavior, etc.

CHAPTER 3

PROBLEM DEFINITION

As mentioned previously, the DIS standard does not specify behavior standards for battlefield elements. Because of this characteristic, associative interoperability can be difficult for elements simulated by different simulations. In addition to the CV linkages already discussed, there is growing interest in virtual-to-virtual linkages (SAF-SAF) because the military wants to conduct large-scale theater of war training exercises. Additionally, there is interest in joint task force operations. Both of these goals require that for virtual simulations, CGF units must be able to be composed of entities that are owned and simulated by different simulations but must act properly under the specified task organization, i.e. each unit must coordinate with every other unit even if they are simulated by different simulations. Entire missions need to be correlated, a behavior at a time, regardless of how the final behavior choices were generated (reactive, intelligent agents, CBR, etc.). Unfortunately, the higher the fidelity of the simulation, the more functionality and encompassing semantics are present, thus increasing the complexity. In many cases there is a discrepancy in the encompassing semantics of the simulations. There are usually differences between the behaviors that each simulation is capable of or in how they are implemented. Behavior addresses the performance and interactions between different simulations to produce the appropriate effects. The main characteristic of performance correlation is how the different simulations interact with one another [Spuhl

and Findley, 1994]. Behavior correlation is one aspect of performance correlation. Simulations may possess the same behavior but do not perform the behavior in the same manner. This may occur for several reasons. Either one of simulations has development errors, was developed with a different objective in mind, or is a research system and is not concerned with 100% correct behavior. Missing or incorrect behaviors are especially a problem when combining forces from different countries.

Smith demonstrates the associative interoperability problem using a simple automobile on the highway analogy [Smith, 1995]. Each automobile is controlled by some autonomous entity (usually a driver) and shares the same environment (roadway) with other entities (vehicles). Each vehicle must react to the operations of the other vehicles in their vicinity. A communication protocol similar to the DIS protocol is used between entities and communicates:

- changes in vehicle state (speed, direction, turn signals, brake lights, etc.) needed by other vehicles.
- accident notification.
- emissions (headlights, horn, exhaust).
- signal (radios, car phones).

A simulation can be created which will adhere to the above communication standards. However, each vehicle may be controlled by an independently developed model of operations. This model may use different algorithms and/or rules for operating on the highway. Not only does this not build a realistic training environment (all drivers in the

real world usually adhere to the same set of rules), but it can cause serious problems when interoperating. These problems can include variations in the understood speed limit, which side of the road to drive on (left or right), yield behavior, right-on-red behavior, and driving surface (which you can drive on, paved roads, dirt roads, sidewalks, gravel, fields, lawns, etc.). So, even though the simulations may generate their PDUs correctly, when playing together the results are not fair or representative of the entities involved. In reality, the method used to standardize the vehicle operations goes beyond the automobile design and communication protocols specified by a DIS-like architecture. A driver's manual is used as a common behavior framework which is required to be learned by a driver before he or she is allowed to participate. The regulations of the framework are enforced by the law to ensure the coordination of various vehicles, with vehicles that do not adhere to the rules (dangerous drivers) being removed from the highways. This maintenance of the modeling framework allows the distributed heterogeneous simulation of automobiles to operate properly. In simulation, a similar modeling framework is needed to ensure the operation of heterogeneous distributed systems. Similarly, maintenance of the framework in the form of modifications or added components to the simulations are necessary to ensure continued interoperability. This example illustrates the problems that can occur in simple automobile simulations. Warfare is many orders of magnitude more complicated than this example, with rules that are not always understood nor agreed or adhered to. The military does have its doctrine but simulation designers, not military experts, are used to create the simulation models and thus differences appear in

implementations. Also, since some simulations may have a different focus than others, the robust implementation of all behaviors may not be a priority. The official rules of warfare becomes the modeling framework that must be adhered to. Since there are no complete and official set of rules for warfare, military doctrine and knowledge of enemy tactics must be used instead.

While few have recognized the semantic interoperability problem, no one has yet to provide a satisfactory solution. Smith [1995] has suggested that common modeling frameworks are necessary for complex cases of interoperability where full integration is required. These common frameworks allow simulations to interact using the same “language” and share functionality. Typically a modeling framework must describe the composition of objects in the scenario, what their capabilities are, what they are affected by. Also it must address the events in the system and effects to be considered by these events. In object-oriented terms, simulation entities, events, etc. can be converted from their specific form to a general form and then to the specific form required by the destination simulation. The extra step of converting to the general common model provides flexibility in that it allows interoperability between different combinations of simulations without having to know the exact combination beforehand. For the correlation of behavior, not only is a common behavior framework necessary but some form of correlation of the behavior is required that can allow simulations to execute the

behavior specified by a another simulation. Correlation has been defined by Spuhl and Findley [1994] as:

“Correlation exists when a simulation imitates all the necessary attributes of an event such that the effect experienced by the observer is appropriate or equivalent to the same effect experienced in another simulator or experienced in the real world.”

For the correlation of behavior, all the necessary attributes of a behavior must be imitated by both simulations. However, due to the differences in simulation behaviors, the behavior correlated for the destination simulation may not be exactly the same as that of the source simulation, i.e. they do not share a common framework. Thus, the best match or correlation must be found. Only major changes to the destination simulation's architecture (to support a common framework) would allow the total correlation specified by the above definition. The common framework approach mentioned by Smith [1995] is not a practical solution for existing systems, and thus a new approach is needed. This research describes the first approach to the correlation of behavior for heterogeneous simulations. The correlation of behavior will be defined as semantic correlation, specifically the semantic correlation of behavior between two heterogeneous simulations. Semantic correlation for behavior needs to not only correlate the best “match” between simulation behaviors, but also correlate the parameters associated with the behaviors. If the parameters of the source simulation behavior cannot be correlated with the destination simulation behavior then the behavior cannot be executed.

Contributions of Research

The intent of this research is to develop and test a methodology that promotes interoperability of behavior among simulations using as much common representations as possible, along with heuristic metrics to correlate behavior. To satisfy the problem of interoperable SAF simulations, this research makes two important contributions :

- The development of a general framework for behavior and behavior parameters that facilitates the correlation between simulations. The structure of this framework is domain independent with the actual contents common to the domain in question. This can provide any combination of interoperating simulations as opposed to creating new point-to-point simulation translation code every time a new simulation is added or a new combination is desired.
- Most importantly, the development of a middleware component (in this case a SAF-to-SAF) that supports interoperability by enabling the correlation of heterogeneous simulation behaviors. The automatic correlation of behavior from a source simulation to an equivalent form for a destination simulation may involve closest fit forms for unknown or different behaviors. To satisfy this requirement, a set of closeness heuristic metrics shall be defined for both behaviors and their parameters that will be used to determine the destination behavior with the best “semantic closeness” to the given source behavior. In addition, a methodology for parameter conversion shall be defined to support run-time correlation of the selected behavior correlation.

CHAPTER 4

BEHAVIOR INTEROPERABILITY

The semantics of a simulation define what the model does or means. From previous discussions on interoperability and the problem at hand, it can be concluded that:

“The problem of interoperability between dissimilar simulations is related to the inherent complexity and is really an issue of semantics” [Altman et al., 1994]

For the problem of semantic correlation, the semantics are the behavior that each simulation’s entities exhibit. Closely associated with the semantics are the structure (syntax) that the implementation uses to describe how the model performs its function [Altman et al., 1994]. Furthermore, Altman et al. [1994] contends that the simulated battlefield lends itself towards hierarchical decomposition and that abstractions are necessary to create a useful hierarchy. Abstraction allows the essential differentiating characteristics of an object to be defined within a given context and thus provides strict conceptual boundaries that make it easier to understand what the object in question represents. While Altman is speaking in general terms about simulations as a whole, this reasoning can be applied to simulation behavior. The approach taken in this research uses abstractions to create behavior hierarchies that can be used to compare the similarity of behavior. Behavior is not only composed of sub-behaviors but many times these sub-

behaviors represent more general cases of the behavior in question. These abstract behaviors will be considered to be primitives at the lowest level of decomposition.

The decomposition of semantics into more general forms is not a new concept. Roger Schank used the primitive concept as a means of describing the semantics of language (i.e. concepts) for natural language understanding [Schank and Kirby, 1993; Schank, 1975]. His conceptual dependency (CD) theory allowed sentences to be expressed in notation that is independent of the source language and facilitated the drawing of inferences from the knowledge present in the sentences. Schank defines a set of 10 primitives into which all sentences are composed [Schank and Abelson, 1977]:

- ATRANS: Transfer of an abstract relationship (e.g. give)
- PTRANS: Transfer of the physical location of an object (e.g. go)
- MOVE: Movement of a body part by its owner (e.g. kick)
- GRASP: Grasping of an object by an actor (e.g. clutch)
- INGEST: Ingestion of an object by an animal (e.g. eat)
- EXPEL: Expulsion of something from the body of an animal (e.g. cry)
- MTRANS: Transfer of mental information (e.g. tell)
- MBUILD: Building new information out of old (e.g. decide)
- SPEAK: Production of sounds (e.g. say)
- ATTEND: Focusing of a sense organ towards a stimulus (e.g. listen)

Schank also defines other constructs analogous to English such as actions, objects, modifiers of actions, modifiers of objects and a set of 14 rules which describe how all the constructs can be combined. The CD representation is quite involved and has not really been used as it was originally intended. Other simpler variants have been created and used in the area of natural language understanding [Allen, 1995]. Fortunately, simulation behavior is less complex than language concepts and can be expressed in simpler terms than natural language. Therefore, many of the constructs and primitives are not necessary.

The idea of common primitives for behavior agrees with various sources in the CGF community. [Smith, 1995] suggests that a common modeling framework is needed to solve the interoperability problem. Similarly, [Altman et al., 1994] contends that a set of unifying semantics are necessary. A set of common behaviors and primitives can provide the unifying semantics necessary for the semantic correlation of behavior for heterogeneous simulations. However, since simulations can only interoperate to the extent that they share common semantics [Altman et al., 1994], the more behaviors and primitives in common, the better the correlation of behavior and thus the simulation interoperability.

Behaviors for military simulation are often expressed as higher level behaviors written in terms of four primitives [Ourston et al., 1995]:

- MOVE: Describes the sequence of steps necessary to move units or platforms, taking into account direction, platform positioning, platform orientation, unit spacing, speed parameters, etc.
- SHOOT: Describes the target priorities, fire distribution, rates of fire, ammunition to use, etc.
- SEARCH/OBSERVE: Describes weapon orientation and search sectors, search techniques (such as seek cover and concealment), search ranges for various target types, etc.
- COMMUNICATE: Describes reports and orders to be sent based upon specific battle conditions, control measures encountered, observations made, etc.

In addition to being expressed in terms of these primitives, the behaviors have associated with them a set of parameters, situational triggers for behavior changes (reaction to enemy contact, for example), and in some cases initial and termination conditions. In some situations, the situational triggers may be considered a specific form of a primitive such a REACT. These primitives and some additional ones can be found in several CGF models and simulations [Landweer, 1993; Ceranowicz, 1994; McEnany and Marshall, 1994].

To measure the interoperability, and thus the commonality of semantics, metrics should be developed based upon the type of entity and task it is to perform [Altman et al., 1994]. For behaviors, metrics can be created that heuristically evaluate their closeness based upon several conditions.

Behavior Representation

The common approach to interoperability between two systems is to create specific “wrappers” around appropriate simulation components that allow them to interoperate [Altman et al., 1994]. The problem with this approach is that it is very specific and does not deal with the general association problem nor the problem of unifying semantics. Given n simulations, there are n^2 interoperability combinations that could possibly be desired. Without some set of unifying semantics the problems will get worse as more and more simulations are developed in isolation. While a complete set of unifying semantics cannot always be guaranteed, a middleware component can be used that will enable any combination of simulations to be connected together. In the terms of behavior interoperability, a middleware component can be used that will translate specific behaviors (from a source simulation) into more general ones which can then be translated into specific destination (destination simulation) behaviors for execution. Once the source behavior is translated into its general form, it can be translated into any of the remaining $n-1$ simulations without prior knowledge of the pairing. This is much more scalable than coding specific point-to-point connections every time a different combination is needed or a new simulation is added. In order to accomplish this, a generic, simulation independent representation of the behaviors was developed. Specific simulation behaviors are translated into behaviors written in terms of general domain behaviors and primitives. An ontology of behaviors and parameters is used to support the similarity metrics. The parameter decomposition and ontology must be completely common between both

systems in order for correlation of parameters to be possible. Since the simulations are in the same domain and parameters are not as sensitive to interpretation as behaviors, this is acceptable. This behavior representation allows simulation specific behaviors to be translated to any of the n-1 simulation systems. The general behaviors comprise a set of behaviors that is sufficient for correlation. Sufficient is deliberately vague in this context. What constitutes sufficient is dependent upon the simulations involved. The more behaviors and primitives that are in common between simulations, the better the correlation.

Behavior Correlation Metrics

Behaviors are usually represented in an aggregate fashion. Higher level behaviors are represented in terms of lower level behaviors until the primitive level is reached. Behaviors may be represented in terms of more general behaviors or the aggregate of lower echelon behaviors. In the case of aggregate lower echelon behaviors, different behaviors may be assigned to different units. This is not a problem since the higher echelon behavior can still be considered to exhibit these behaviors even though not all lower echelon units exhibit all the behavior. Because there is an infinite number of ways the same behavior can be represented a simple comparison is not sufficient. When trying

to compare and correlate behaviors several metrics can be used to determine how similar they are:

- A source behavior can be found at a lower or higher level of decomposition of a behavior than in the destination behavior. This is defined as the WHERE-IS metric.
- A source behavior can be decomposed into its sub-behaviors which can then be correlated. This is defined as the HAS-A metric.
- A source behavior can be related to a more general or more specific behavior present in the destination behavior. This is defined as the IS-A metric.
- A source behavior can be related to a similar behavior of the destination. This is defined as the SIBLING-OF metric.

Any combination of these metrics can be used at the various levels of decomposition to determine the semantic closeness of two behaviors. In this context, semantic closeness is defined as the percentage that the destination behavior will perform the desired behavior. There is no guarantee that the chosen behavior will execute the same behavior as the source, only that it will be the best match possible among the available destination behaviors. Many times, behaviors may be essentially the same but are organized differently. There are five major cases that illustrate the various ways differently structured behaviors can be correlated. The five cases use contrived examples of behavior from the military domain for the sole purpose of illustrating the possible metrics. The behaviors of interest in each case are represented in italics.

The first case illustrates a source behavior that is found one deeper level of decomposition on the destination side. If the behavior is not found, then its subcomponents can be used as a means of correlation. An example of the first case is:

CASE 1: Lower Level WHERE-IS

Behavior A:

TRAVEL
 MOVE
OCCUPY_POSITION

Behavior B:

CAUTIOUS_MOVE
 TRAVEL
 OCCUPY_POSITION

Case 2 illustrates a similar situation but in reverse, the behavior is found two levels of decomposition higher:

CASE 2: Upper Level WHERE-IS

Behavior A:

ASSAULT
 TRAVEL
 OCCUPY_BP
 TRAVEL
 TARGETER
 OCCUPY_POSITION
CONSOLIDATE

Behavior B:

ATTACK BY FIRE
 TARGETER
 TRAVEL
 OCCUPY_POSITION

CONSOLIDATE

Case 3 illustrates the situation where the behavior is decomposed into its sub-behaviors and correlated:

CASE 3: HAS-A

Behavior A:

ASSAULT
 TRAVEL
 OCCUPY_BP
 TARGETER
 TRAVEL
 OCCUPY_POSITION
 CONSOLIDATE

Behavior B:

TRAVEL
 OCCUPY_BP
 TARGETER
 TRAVEL
 OCCUPY_POSITION
 CONSOLIDATE

Case 4 illustrates both the general-to-specific and specific-to-general IS-A correlation.

When correlating from behavior A to behavior B the more specific

HASTY_OCCUPY_BP can be used in place of OCCUPY_BP. When correlating from

behavior B to behavior A, the more general OCCUPY_BP can be used in place of

HASTY_OCCUPY_BP. Case 4 is as follows:

CASE 4: IS-A

Behavior A:

ASSAULT
 TRAVEL
 OCCUPY_BP
 TRAVEL
 MOVE
 SHOOT
 CONSOLIDATE

Behavior B:

ASSAULT
 TRAVEL
 MOVE
 TARGETER
 HASTY_OCCUPY_BP
 OCCUPY_POSITION
 CONSOLIDATE

Case 5 illustrates the SIBLING-OF correlation. Here BOUNDING_OVERWATCH is correlated with TRAVELING_OVERWATCH since they are inherited from the same parents, and hence similar:

CASE 5: SIBLING-OF

Behavior A:

ASSAULT
 BOUNDING_OVERWATCH
 TRAVEL
 OCCUPY_POSITION
 OCCUPY_POSITION
 CONSOLIDATE

Behavior B:

ASSAULT
 TRAVELING_OVERWATCH
 TRAVEL
 OCCUPY_POSITION
 OCCUPY_POSITION
 CONSOLIDATE

Extra behaviors may also be present on either the source behavior or destination behavior.

Extra behaviors on the destination behavior do not affect the closeness as it has been defined.

Extra behaviors only mean that the destination behavior does more than needed which is acceptable. Only if the extra behaviors drastically cause the behavior to conflict with the source behavior will there be a problem. There may also be some ambiguity if more than one destination behavior share the same subset of behaviors that match the source behavior. As far as the semantic closeness is concerned the behaviors are equal. A modification to the algorithm could be made that would choose the behavior will the least amount of extra behavior but that is no guarantee that behaviors will not be ambiguous. Extra behaviors on the source behavior do decrease the closeness since the destination behavior may be missing some important functionality.

Parameter Correlation Metrics

In addition to performing metrics when correlating behaviors, metrics must also be calculated for correlating the parameters associated with that behavior. Parameters either are necessary for the corresponding behavior to perform its function or modify how the behavior is executed. Common parameters for military behaviors include speed, formation, platform, route, etc. The metrics define how close the parameters between the two behaviors match. Parameter correlation is only performed for the top level source and destination behavior. The parameters of sub-behaviors are not really significant since as long as the initial parameters correlate, the behavior can be executed. In addition, many times the sub-behavior parameters will be derived internally and have no explicit relationship to the top level parameters.

There are three metrics that apply to parameter correlation, the IS-A, PARENT-OF and HAS-A metrics. The IS-A and PARENT-OF metrics both determine the closeness along an inference path between a source parameter and destination parameter. The IS-A metric determines if a destination parameter is a child of one of the source parameters. The metric determines the inferential distance between the two. Similarly, the PARENT-OF metric determines if a destination parameter is a parent of one of the source parameters. Unmatched (Additional) parents in a PARENT-OF metric also do not affect the closeness for the parameter. This just means that the parameter is more complex than the source parameter being correlated which is satisfactory. These two metrics can be combined to generate a correlation path from a specific source parameter to a more

general parameter and then back to a more specific destination parameter. For example, an ASSAULT_POSITION can be correlated to an OBJECTIVE by following the inference path from ASSAULT_POSITION to POSITION to AREA to OBJECTIVE, where OBJECTIVE is a specific type of AREA. The HAS-A metric determines the closeness along a decomposition path between a source parameter and destination parameter. For example, suppose a ROUTE can be decomposed into a START_POINT and END_POINT. Then, a source ROUTE parameter can be correlated with START_POINT and END_POINT parameters of the destination behavior. The IS-A and PARENT-OF metrics can be combined with the HAS-A metric so that the sub-parameters of parameter may also be matched with destination parameters.

Incremental Decomposition and Abstraction

The correlation algorithm uses incremental decomposition and abstraction of behaviors to determine the closeness. Each source behavior is recursed into and is compared (via recursion again) to the levels of the destination behavior. Each behavior is decomposed into its sub-behaviors which are also correlated down to the primitive level. The correlation algorithm uses the following high level steps when correlating a source behavior:

- 1) Check for the presence of the source behavior at the given level of decomposition in the destination behavior.
- 2) If the behavior is not present, apply the WHERE-IS, IS-A, HAS-A, and SIBLING-OF metrics, using the maximum closeness result.

- 3) Recurse into the source behavior, performing these steps on each sub-behavior. Combine the results of the sub-behavior correlations and multiply the result by the closeness value determined in one of the two previous steps.
- 4) Repeat steps 1-3 on the next behavior at this same level of decomposition.

The parameter correlation algorithm follows the same basic steps, with the parameter metrics being applied instead. It is important to note that behaviors can increase the closeness if they match, but behaviors that match in name are not necessarily equal. The closeness must be determined down to the primitive level to determine an accurate correlation (hence the presence of step 3 above). The correlation algorithm uses the semantic closeness metrics defined earlier to determine the behavior closeness value. This value is calculated using closeness factors (decreases in closeness) for each metric along with a few others. These factors may need to be adjusted for a specific destination system to guarantee proper correlation.

Regardless of the actual values, intuitive (fuzzy) values can be assigned to the following adjustments to the closeness of two behaviors:

- **MEDIUM** The decrease in closeness for finding a behavior at an extra level (+/-) of decomposition (WHERE-IS metric).
- **LOW** The decrease in closeness for using one inheritance level of generality or specificity instead of the exact behavior (IS-A metric).
- **MEDIUM** The decrease in closeness for using the sub-behaviors instead of the behavior itself (HAS-A metric).
- **MEDIUM** The decrease in closeness for using a sibling behavior (SIBLING-OF metric).

- HIGH The decrease in closeness for not correlating a source behavior at all (although this may vary depending upon the total number of source behaviors at the given level of decomposition).
- LOW The closeness fraction assigned to the contribution of the parameters as a whole.
- VERY-LOW The closeness fraction assigned to the contribution of reactive behaviors.
- LOW The decrease in closeness for using a more general or specific parameter (PARAMETER IS-A/PARENT-OF metrics).
- VERY-LOW The decrease in closeness for using the sub-parameters instead of the parameter itself (PARAMETER HAS-A).
- MED-HIGH The decrease in closeness for letting a parameter default.

As each source behavior is correlated, the metric that produces the best closeness value is combined with the aggregate closeness value of its sub-behaviors. The value is then combined with the other behaviors at the same level of decomposition and filtered up to the upper levels of decomposition. At the top-level, the correlation of the behaviors is combined with the parameter correlation to obtain a final correlation for the behavior in the range between 0 and 1. Each sub-behavior (except reactive behaviors) are equally important in the closeness determination. Reactive behaviors count for less since they do not define the behavior, only their presence helps determine the closeness. The algorithm makes sure that it does not recurse into reactive behaviors when looking non-reactive source behaviors since this would drastically throw off the correlation. Also, a destination sub-behavior can be correlated against a source behavior more than once. In some cases

this makes sense and is useful if a destination behavior encapsulates more of the source behavior. However, in some cases this is not true. The uncertainty is captured by the decrease in closeness factor for the correlation but no decrease in correlation is currently implemented for multiple destination matches.

The parameter correlation mechanism is a simpler form of the behavior correlation algorithm. As mentioned previously, this is primarily because it is focused on a conversion path not just similarity. The WHERE-IS metric is not used since sub-parameters on the destination side are never recursed into. Source parameters are broken up into their constituents if necessary and these are matched against the top-level destination parameters only. Missing parameters contribute a portion of the closeness if they are default. Unmatched required parameters on the destination side will set the entire behavior closeness to zero, because even if the behaviors are similar, if the parameters cannot be correlated then the behavior cannot be executed. Unmatched required source parameters only decrease the closeness determination by setting their closeness contribution to zero. Both source and destination parameters that are default and cannot be correlated are not set to zero only the closeness is reduced by a specified amount. Default parameters are defined as those which have preset values within their appropriate simulations and are not required to be set for the behavior to be executed.

Related Work

The interoperability of behavior is just beginning to emerge as an issue in the CGF arena. [Altman et al.; 1994] have suggested the problem of unifying semantics for interoperability and [Smith, 1995] has demonstrated the problem of behavior interoperability. The concept of a common modeling framework has been proposed by Smith but no concrete solutions have emerged. Similar solutions to the one proposed have been used in other completely different areas, however.

The use of similarity metrics has been used for several years in the retrieval of cases for CBR. Castillo [1991] uses three metrics to determine the similarity between stored plans for an intelligent agent. A taxonomic metric is used in the same fashion as the IS-A metric to relate specialization classes with one another by their inferential distance. A feature similarity metric is used to compare case features much like the HAS-A metric. Scalars are multiplied by every feature matched and the results are summed. Heuristics are used for qualitative features such as resources, locations, color, etc. An importance metric is also defined to allow cases to be matched based upon their closeness in importance. These metrics all use domain-dependent ontologies and heuristics in their calculation. These metrics are employed on plan cases to find the best plan matching the current goals, often retrieving related cases much in the same fashion that related behaviors in this research are correlated.

In the domain of model-based reasoning knowledge acquisition, a problem can occur when trying to identify unknown items and determine their function from CAD

databases for placement into a knowledge base to support model-based diagnosis.

Drawings in CAD databases do not always use the same terminology or spellings and the information contained within is usually not detailed enough to support model-based diagnosis. A solution proposed by [Gonzalez et al., 1992] known as the Heuristic String Identifier (HSI) uses heuristics to identify unknown items by comparing an unconstrained description of the item with known items in a database. In addition, functional constraints of the item are examined and matched with those in the database. Each potential match is assigned a confidence factor associated with the similarity between the corresponding item descriptions and functional characteristics (output units of a component, type of item, etc.). These heuristics are comprised of various string-matching metrics that are combined to determine the similarity between item descriptions. The combination of heuristics provides an evaluation that is more powerful than any of the individual heuristics alone [Gonzalez et al., 1992].

The HSI uses several levels of metrics. The first metric searches for the longest common subsequence of the two description strings and is used as a bias for applying the remaining metrics. If the confidence factor for this metric does not exceed a specified threshold then the remaining metrics are not applied. This serves to prune the search space. There are seven remaining string comparison metrics [Gonzalez et al., 1992]:

- WILDCARD-MATCH-RATIO
- CHAR-CLUSTER-BACKTRACK-RATIO
- CONSEC-CHARS-BY-ORDERED-WORD-RATIO

- CONSEC-CHARS-BY-UNORDERED-WORD-RATIO
- ORDERED-COMMON-WORD-RATIO
- UNORDERED-COMMON-WORD-RATIO
- COMMON-CONSECUTIVE-WORD-RATIO

The first four metrics address similarities at the character level and the remainder at the word level. Attributes that are addressed are clustering of words, abbreviations of terms, misspellings, non-standard terminology, word order, and extraneous words [Gonzalez et al., 1992]. Extraneous words in a unidentified description will decrease the confidence of an otherwise close match. Each heuristic is weighted to reflect the significance they each provide to the overall similarity.

The database used to match items against is a hierarchical organized representation of item descriptions. The combined metrics are applied against destination items in order from general to specific. There are three levels: general, intermediate and leaf. Metric results must pass specified thresholds in order to go to the next level of specialization, otherwise the result is stored as a weak match (unless at the most general level).

Beginning at the intermediate level, functional metrics are applied in conjunction with the string metrics. Item attributes such as the output units of the component are checked for consistency. Units are organized into conceptual clusters of like units. For example, all distances would be grouped together as would units of power. The goal of this kind of classification is to determine the underlying conceptualizations of the item [Gonzalez et al., 1992].

The problem HSI addresses is similar to the problem of matching (correlating) unknown behaviors using various metrics with a database of destination behaviors. Both databases are hierarchically organized but in the HSI approach no penalty is assigned for matching an item with a more general one as is the case with behaviors. The metrics used in both cases have assigned weights that are used to combine them in the total heuristic metric calculation. Like HSI, extra behaviors of the source behaviors decrease the closeness value, which is analogous to the decrease in similarity calculated for an item description containing extraneous words. However, in the case of behaviors, extra behaviors are probably not extraneous but important.

Unfortunately for the problem of CGF behaviors, no assumption can be made about the similarity of behaviors based upon the similarity of their names, so alternative attributes have been used such as the similarity of sub-behaviors (HAS-A) and the inheritance related metrics (IS-A/SIBLING-OF). The major difference, other than the metrics themselves, is that the behavior and parameter metrics that have been defined compete with one another and are not combined together for the same behavior (they are combined only through behavior decomposition). Behaviors that correlate with another behavior using multiple metrics do not necessarily correlate better than another behavior that correlates with just one. For example, if a behavior correlates very well with a destination behavior using the IS-A metric and the behavior correlates with a different behavior with the same IS-A metric value but also has an additional SIBLING-OF metric

value, does this mean that one correlates better because it has a related behavior present?

There is no evidence to support this.

The functional consistency checks HSI uses to identify underlying conceptualizations is the same approach taken when decomposing behaviors into more general and abstract behaviors. Both serve to conceptualize the object in question, whether it be an item or a behavior. The conceptual clustering of units in the HSI database is a direct analogue of the behavior and parameter ontologies. The conceptual clusters relate similar units, i.e. measuring the same item attribute, and can even suggest a conversion of units just like the parameter correlation.

The HSI approach uses an initial metric as a bias for applying the remaining metrics. In behavior correlation there is no bias metric per se. Pruning is difficult because there are no guarantees about the closeness of a particular behavior until its sub-behaviors have been examined. Only at the topmost level could a determination be made as to whether there was a possibility of beating the best correlation so far or not. Fortunately, the search space of unit assignable behaviors are usually small and the interval between orders being received by unit is usually long enough to permit correlation. The parameter correlation can be considered a form of bias however, since if any of the required destination parameters cannot be correlated then the behavior will be removed from further consideration.

CHAPTER 5

EVALUATION PROTOTYPE

This research focused on the correlation of CCTT tank platoon behaviors with that of ModSAF tank platoon behaviors so they could interoperate under one task organization. Only those behaviors that could be assigned to tank platoons via their respective GUIs were considered for source and destination correlation. Each CCTT behavior assigned to a ModSAF platoon would be correlated with the best matching ModSAF behavior and its parameters converted and the behavior assigned.

CATT-SAF

The Semi-Automated Forces (SAF) component of the Close Combat Tactical Trainer (CCTT), called Combined Arms Tactical Training SAF (CATT SAF), simulates US Army and Soviet Army tactical behaviors for vehicle, platoon, company and battalion echelons. These behaviors are developed from documented and validated Combat Instruction Sets (CISs) [Ourston et al., 1995] and, as such, are useful for training, mission rehearsal, acquisition, and test and evaluation (T&E) environments. CISs combine to form an execution matrix (mission). In some cases, more than one CIS is covered by a single behavior.

The behaviors are executed using an FSM approach in procedural code (no strict FSM format) with embedded behaviors executed through additional FSMs or executed across echelons via orders. The FSM representation is not used past the CIS level, i.e. some of the lower level primitive behaviors are represented as procedural code. There is currently no provision for a data driven approach to behavior execution. The following are the tasks (CISs) that can be assigned to CCTT tank platoons:

- ACTION DRILL
- ACTIONS AT OBSTACLE
- ACTIONS ON CONTACT
- ASSAULT
- ASSAULT POSITION ACTIVITIES
- ATTACK BY FIRE
- BOUNDING OVERWATCH
- CONSOLIDATE AND REORGANIZE
- DISPLACE TO SUBSEQUENT BP
- HALT
- HASTY OCCUPATION OF BP
- OCCUPY ASSEMBLY AREA
- OCCUPY BP
- PASSAGE OF LINES
- PLATOON DEFENSIVE MISSION
- PLATOON FIRE AND MOVEMENT
- REACT IF
- REACT TO DI ATTACK
- REACT TO AIR ATTACK
- RESUPPLY
- TACTICAL ROAD MARCH
- TRAVEL
- TRAVELING OVERWATCH

Reactive behaviors such as actions on contact and react to indirect fire are implicitly defined for each behavior via a situational interrupt table. This table lists the triggers for each applicable behavior that will invoke the reactive behaviors.

ModSAF

The Modular SAF (ModSAF) is a widely used research system (and thus more has been published in the literature about it) that also provides tactical behaviors, albeit not validated ones since they were not developed from validated CISs. It is primarily designed as research system for battle labs involving experimentation with new behaviors and equipment. Its hierarchical, cohesive, modular behavior structure supports ease of modification, extension, and flexible behavior implementation methods. ModSAF provides a framework for command and control but does not limit C2 implementation. ModSAF also provides a general representation for unit and individual behavior within its architecture but does not require any specific implementation. However, ModSAF does supply many tank platoon behaviors a priori and, as in CCTT, these behaviors are implemented using FSMs, specifically AAFSMs that provide extreme flexibility in behavior generation [Calder et al., 1993]. All behaviors, regardless of echelon are represented as AAFSMs down to the primitive level.. Orders to subordinates are handled in the same modular fashion, showing a good decomposition of behavior.

The foundation of the ModSAF C2 architecture is built on the concept of a task. Tasks specify the behavior control of a platform or unit and models the information processing done by the simulated entities. As previously mentioned, these tasks are

organized hierarchically to support varying levels of abstraction and aggregation. Tasks are composed of the task model, the task state, and task parameters. The task model is composed of the task specific states, an ended state, suspended state, and task data structures. The ended and suspended states perform special processing when a task is ended or suspended. The task state is shared to allow tasks be monitored by the tasks that launched them. This provides a level of command and control and allows another individual or unit to take over the task execution without interruption [Calder et al., 1993]. ModSAF defines five distinct types of tasks: unit, individual vehicle, reactive, enabling, and arbitration. Unit tasks are the most common, consisting of behaviors that units such as sections, platoons, companies, etc. typically execute. These tasks model the military command structure and thus are primarily concerned with controlling the behavior and monitoring the progress of subordinate units. Individual tasks are concerned with modeling the physical behavior of specific vehicles or infantry and interact with the weapon, sensor and communication subsystems of the vehicle. Examples include movement, collision reaction, obstacle avoidance, sensor scanning, enemy detection, attack detection, target selection, and weapons firing. Reactive tasks are variants of unit tasks. They combine a unit task behavior with reactive triggers that execute these behaviors based upon certain environmental conditions. The unit tasks may be embedded in the reactive task or called by the behavior. For tank platoon behaviors, ModSAF uses three different reactive behaviors: react to air attack, react to enemy contact, and react to indirect fire. Enabling tasks are similar to reactive tasks but

used for mission command and control. They trigger different predicted contingencies of a mission depending upon assessment of the current situation. Examples include crossing a phase line, detecting an enemy unit, reaching an H-Hour time, etc. The arbitration tasks are internal tasks that arbitrate between competing recommendations from other tasks to form a single recommendation. Usually these tasks are performed at the lowest level to control vehicle subsystems but can be used at higher levels to perform mission planning. ModSAF provides vehicle arbitration tasks such as movement, sensor, and targeting arbitration. Refer to [Calder et al., 1993] for a more in-depth discussion on task arbitration.

ModSAF uses the concept of task frames to organize sets of behaviors together to form distinct tasks that can be assigned to a unit (similar to a CIS) using the ModSAF GUI [Ceranowicz et al., 1994]. Multiple task frames form the execution matrix (mission) and manage the execution of tasks. Each task frame represents a phase of a mission and is composed of a preparatory frame and actual frame. The preparatory frame consists of tasks that prepare the overall task to be executed. This usually consists of a halt task. The actual frame contains the primary (foreground) task to be executed. Reactive tasks (background) can also be assigned in both frames. Task frames are placed on a task frame stack which can be transparent so that multiple behaviors can be executed simultaneously. This enables a single unit or vehicle to simulate multiple roles such as a commanding unit and normal unit. If the new task frame being placed on the task frame stack is not transparent, then the currently executing task frame is suspended. This is

common when reactive tasks push new tasks onto the task frame stack. When the reactive behavior is no longer needed, the original task frame is reinstated. Task frames support C2 behavior in that tasks can be added or deleted to a frame by a superior unit to control its behavior.

ModSAF task frames are data driven which provides some flexibility in setting up new behavior combinations. Task frames may be also created by tasks to combine other tasks together to form specific functions. Platoon tasks often create individual vehicle task frames by adding movement, weapon, and scanning behaviors to a new frame that is assigned to the vehicle. The task frame also supports the decomposition of complex behaviors into more simpler ones. The actual tasks in the task frame however, must be created as an AAFSM and then compiled in with ModSAF. The following is a list of the task frames that can be assigned to ModSAF tank platoon units from the GUI

[Ceranowicz, 1994; Courtemanche and Ceranowicz, 1995]:

- ASSAULT
- ASSEMBLE
- ATTACH
- ATTACK BY FIRE
- BREACH
- CHANGE FORMATION
- CONCEALMENT
- DELAY
- DETACH
- FOLLOW SIMULATOR
- FOLLOW VEHICLE
- HASTY OCCUPY POSITION
- MOVE
- OVERWATCH MOVEMENT
- PLOW BREACH
- PURSUE

SUPPLY
TACTICAL ROAD MARCH
TRAVELING OVERWATCH
WITHDRAW

At first glance, it can be seen that CCTT is more robust in its coverage of platoon behaviors [McEnany; Marshall, 1994]. CCTT has 23 assignable behaviors versus ModSAF's 20. More importantly, the CCTT behaviors are doctrinal and validated. Upon closer comparison of behaviors, many of the ModSAF behaviors are not as robust as their CCTT counterparts so 100% correlation is not possible. Many of the parameters are very different from one another for the same behavior, resulting in incomplete or unpredictable results. If ModSAF was truly data driven down to the FSM level with a complete set of code primitives that could be used, then CCTT behaviors could be imposed on ModSAF units. Since that is not the case however, interoperability among these two systems is only possible to the degree that the best matching ModSAF behavior will be selected for a corresponding CCTT behavior. The selected ModSAF behavior may do more or less than what is required by the CCTT behavior. This difference arises from a difference in philosophy behind the two systems. ModSAF, being a research and battle lab system uses different tactics. The systems differ in implementation interfaces, primitives, and even simulation fidelity.

Implementation of Approach

As a proof of concept of the approach, the ModSAF to CATT-SAF linkage proposes a framework to facilitate SAF-to-SAF interoperability by supporting the

correlation of behavior. The framework contains a general ontology of behaviors that is sufficient enough to allow correlation among behaviors. Each simulation has their own organization of behavior. The more primitives and behaviors that are in common, the better the correlation. In order to provide correlation, a common ontology of behaviors (IS-A hierarchy) is used as is a common ontology of behavior parameters.

An object database management system (ODBMS) known as Object Store was used as the foundation of the interoperability framework. Behaviors and parameters are stored as objects in the database each with references to other behavior and parameter objects as well as ontological information. A collection of destination that can be assigned to a unit is also maintained. Using an object database supports the current needs of interoperability. Using a standard, generic object domain model, different types of simulations can interoperate with linkages accomplished at a higher level than just the vehicle level. Several current research efforts such as Advanced Distributed Simulation (ADS), WarBreaker, and JSIMS are also looking into a object-oriented middleware layer for the DIS architecture [Peck, 1995].

Peck recommends ODBMSs for systems that require a large number of persistent fine-grained objects [Peck, 1995]. For interoperability at higher levels, a common domain model is needed to represent the entities, events, decisions, behaviors, etc. that can be used by the simulations that are interoperating together. Object-oriented modeling provides such a model that incorporates all the advantages of object-oriented programming (encapsulation, polymorphism, inheritance) [Rumbaugh, 1992]. This domain model (objects and their attributes) must be a superset of the domain models used

by the interoperating simulations. When this is true, a centralized ODBMS can provide the same world view to all the simulations, converting their individual representations as necessary. This is an important fact to consider in the case of CGF behavior. Additionally, ODBMSs provide for complex relationships, complex heterogeneous data types, collections of objects, and complex queries in addition to the basic advantages of object-oriented technology (encapsulation, polymorphism, inheritance). Queries offer the potential for the efficient extraction of relevant information from the world state and triggers can be used to alert simulation components when relevant objects are added or changed.

In the middleware component, an ODBMS acts as a mechanism for the sharing of persistent data and integration of applications that use this data across different platforms with multiple applications and multiple users. The object behavior representation used in this work does not take advantage of all the features mentioned above that ODBMSs can provide, but does allow the sharing of behavior information between two different SAF systems.

The behavior representation used for this work is representative of CCTT and ModSAF behaviors in that it supports the decomposition of complex behaviors into simpler behaviors. It also supports the command and control of higher echelon units by allowing lower echelon behaviors to also be specified. Figure 1 shows an example of the representation using the CCTT Assault behavior. This representation only contains the first level of behavior aggregation. Each sub-behavior has its own representation and

arguments. Together these form a complete behavior hierarchy. A partial hierarchy of behaviors is shown in Figure 3.

```
(ASSAULT
  (ARGS (unit UNIT_ID) (unit_kind PLATFORM) (route_to_ap ROUTE_TO_AP)
    (assault_route ASSAULT_ROUTE)
    (enemy_position ENEMY_POSITION)
    (trigger_line TRIGGER_LINE $Default)
    (assault_position ASSAULT_POSITION)
    (platoon_departure_time DEPARTURE_TIME $Default)
    (obstacle OBSTACLE $Default)
    (breach_route BREACH_ROUTE $Default)
    (pre_breach_route PRE_BREACH_ROUTE $Default)
    (post_breach_route POST_BREACH_ROUTE $Default)
    (alpha_section ALPHA_SECTION $Default)
    (bravo_section BRAVO_SECTION $Default)
  )
  (ISA TRAVEL SHOOT)
  (REACTIVE OFF)
  (BOUNDING_OVERWATCH "bounding_overwatch.bvr")
  (TRAVEL "travel.bvr")
  (VEHICLE_OCCUPY_POSITION "vehicle_occupy_pos.bvr")
  (SEEK_COVER_AND_CONCEALMENT "seekCC.bvr")
  (CONSOLIDATE_AND_REORGANIZE "consolidate_reorganize.bvr")
)
```

Figure 1. CCTT Assault Behavior in Terms of General Representation

The representation is laid out in data files similar to a frame used in knowledge representation systems. It provides slots for arguments, its parents (ISA), its children (PARENTOF), its sub-behaviors, and whether it is a reactive behavior or not. The arguments are supplied with a name, domain type, and optional default marker. The name field is not used in this work. Sub-behaviors are specified by their name and their corresponding file name. The file name is also not used in this work. Data files specifying

behaviors are parsed by the ODBMS behavior objects and stored. A similar representation is also used for the behavior parameters. Figure 2 shows the representation of the route parameter. A partial hierarchy for the parameters is shown in Figure 4.

```
(ROUTE
  (SUBPARMS (start_point START_POINT) (end_point END_POINT))
  (ISA LINE)
  (PARENTOF ASSAULT_ROUTE ROUTE_TO_AP PRE-BREACH_ROUTE
    POST-BREACH_ROUTE ROUTE_TO_REAR_OF_BP)
)
```

Figure 2. General Parameter Representation

Similar to behaviors, parameter frames contain slots for any sub-parameters it contains, a parent slot, and children slot. Data files containing parameter information are also converted into objects and stored in the database. Note that each semantically different parameter must have a unique type specification. If it did not then the correlation algorithm would not be able to disambiguate which source parameters correlate with which destination parameters. This parameter representation must be completely common (the behaviors need only have some in common) because a conversion path must be specified between a source and destination parameter.

Correlation Algorithm Implementation

The general idea of the interoperability mechanism is that simulation “plugs” will be connected to each simulation. The source plug will monitor and intercept orders to destination units. It will convert these orders to a general parameter and behavior representation using the general behavior ontology and sufficient set of generic behaviors.

The behavior will then be correlated against all the assignable destination behaviors for that unit. The one with the best closeness determination will be sent to the unit.

The correlation algorithm uses incremental decomposition and abstraction of behaviors to determine the closeness. Each source behavior is recursed into and this is compared (via recursion again) to the levels of the destination behavior. Pseudo-code for both the behavior and parameter correlation algorithms and their corresponding metrics can be found in the appendix.

The top level procedure (*correlate*) of the algorithm loops through all the assignable destination behaviors and determines the closeness (*DetermineCloseness*) value for each one of them. The behavior with the maximum closeness is chosen and its parameter correlation displayed.

The next level of the algorithm (*DetermineCloseness*) calculates the behavior and parameter closeness. These are combined together as shown below. This procedure calls *CreateClosenessPly* for the behavior in question.

CreateClosenessPly creates what can be thought of as the ply of a tree that stores the metric values for all the levels of behavior decomposition. The procedure initiates the different metrics and keeps track of the maximum closeness value. Regardless of which metric returns the maximum closeness, the destination behavior is recursed into to check the closeness of its sub-behaviors. This is done by calling *CreateClosenessTree*, an indirect recursive call. *CreateClosenessPly* is recursively called for the next source behavior on this same level of decomposition.

CreateClosenessTree loops through each sub-behavior and sums their closeness values together to determine the closeness value for the entire behavior at this level of decomposition. *CreateClosenessPly* is called on each sub-behavior, an indirect recursive call.

The algorithm performs the IS-A, HAS-A, SIBLING-OF, and WHERE-IS metric on each behavior being correlated. The WHERE-IS metric is similar to the HAS-A metric in that the source behavior is correlated wherever it is found in the destination behavior. The HAS-A metric is used regardless whether the behavior is found or not, in order to verify the closeness of the sub-behaviors. The metric that returns the highest closeness value will be used as the semantic closeness determination. Each metric uses a closeness adjustment that decreases the semantic closeness. These adjustments are initially assigned intuitively and adjusted based upon experimentation with the simulations in question. Closeness percentages are also used to combine closeness values together. The following closeness adjustments and percentages are currently assigned for behavior correlation:

•	WHERE_IS_ADJUSTMENT	.20
•	HAS_A_ADJUSTMENT	.20
•	IS_A_ADJUSTMENT	.10
•	PARENT_OF_ADJUSTMENT	.10
•	SIBLING_OF_ADJUSTMENT	.20
•	REACTIVE_PERCENTAGE	.10
•	PARAMETER_PERCENTAGE	.30

The WHERE-IS metric decreases the closeness for the given behavior depending upon the difference in the level of decomposition of the source behavior and the level of decomposition the same behavior is found at in the destination behavior:

```
FOR I = 1 TO LEVEL_DIFFERENCE
    closeness = closeness - WHERE_IS_ADJUSTMENT*closeness
```

The *Where_Is* procedure recurses through each level of destination behavior and each sub-behavior to find the source behavior. For each level of decomposition difference, the current closeness is reduced by the WHERE_IS_ADJUSTMENT percentage amount. For example, if a source behavior is at the first level of decomposition (a sub-behavior) and is found at the third level of decomposition in a destination behavior, the closeness adjustment causes the closeness to be initially 80% then finally 64%. This assumes the initial closeness is 100% which may not always be the case (see HAS-A metric). Since the behavior may be found in more than location in the destination behavior as shown in the correlation cases, the found behavior with the smallest decomposition difference is used as the final metric result.

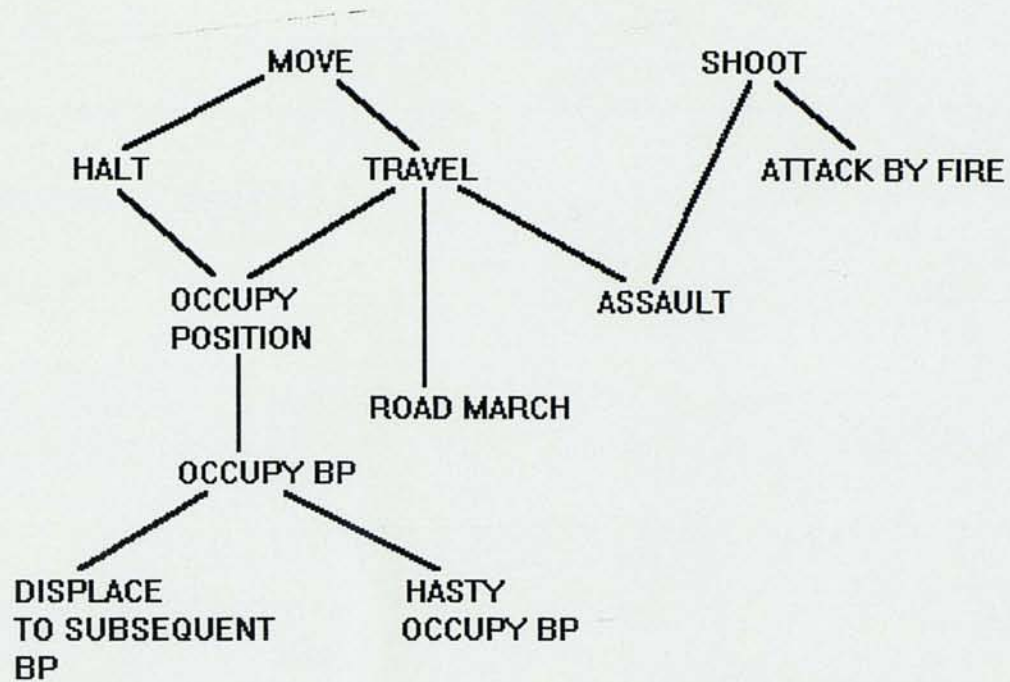


Figure 3. Partial Hierarchy for Tank Platoon Behaviors

The IS-A and PARENT-OF metrics are determined similarly based upon the inferential distance between the source behavior and the more specific or general behavior found in the destination behavior. The metrics use the relationships shown in Figure 3 to determine the inferential distance and apply the following calculation:

```

FOR I = 1 TO INFERENCE_DISTANCE
  closeness = closeness - (PARENT_OF_ADJUSTMENT or
    IS_A_ADJUSTMENT)*closeness
  
```

The *Parent_Of* procedure recurses through the source behavior's PARENT-OF links to examine every child behavior. Similarly, the *Isa* procedure recurses through the source behavior's IS-A links to examine every parent behavior. The IS-A metric has an additional adjustment because multiple parents may be involved. With multiple parents,

versions of all parents must be found in the destination behavior in order for the behavior to truly be represented. If not all parents are found, then the closeness must be decreased. Each parent contributes equally in the following fashion and is summed up to give the total metric closeness:

```
isa_contribution_percentage = 1.0 / Number_Of_Parents
total_closeness = 0
FOR each parent
    total_closeness = total_closeness + isa_contribution_percentage *
        isa_closeness
```

As previously discussed, the PARENT-OF metric does perform this additional step since additional parents of a more specific behavior do not contribute to the closeness. Both the IS-A and PARENT-OF metrics are further modified based upon where the more specific or general behavior is found in the destination behavior using the WHERE-IS metric.

The SIBLING-OF metric tries to correlate a similar (sibling) behavior in the destination behavior with a source behavior. The closeness is adjusted based upon where the sibling behavior is found in the destination behavior and the number of parents shared by the source behavior and the sibling behavior. The best metric of all the siblings is used as the final metric result:

```
FOR each sibling
    closeness = (WHERE-IS metric of sibling) *
        Number_Of_Parents_In_Common / Number_Of_Parents
    if (closeness > max)
        max = closeness
    closeness = max - SIBLING_OF_ADJUSTMENT*max
```


The *SiblingOf* procedure loops through each parent of the source behavior and then through each child of the parent and calculates the above metric.

The HAS-A metric tries to correlate the source behavior's sub-behaviors in the destination behavior. The sub-behaviors are subjected to the previous metrics. The maximum for each of these metrics is used for each sub-behavior which are then combined together. The sum is then reduced in closeness by the HAS_A_ADJUSTMENT amount. In this implementation, the HAS_A_ADJUSTMENT is actually applied first which the other metrics use as their initial value:

```

closeness = 1.0 - HAS_A_ADJUSTMENT
contribution_percentage = 1.0 / Number_Of_Subbehaviors
closeness_sum = 0
FOR each sub-behavior
    closeness_sum = closeness_sum + max_of_metrics_for_sub_behavior
closeness = closeness * closeness_sum

```

The contribution_percentage may adjusted if any sub-behaviors are reactive behaviors. Reactive behaviors, as previously discussed, contribute differently than other sub-behaviors. The reactive contribution percentage is defined for all reactive behaviors as a whole. When the reactive behavior metric values are summed, they will contribute no more than the REACTIVE_PERCENTAGE amount:

```

reactive_contribution_percentage =
    REACTIVE_PERCENTAGE / Number_Of_Reactive_Subbehaviors

```

For the non-reactive behaviors on the same level of decomposition, the contribution percentage is adjusted to reflect the presence of reactive behaviors:

$$\text{contribution_percentage} = (1.0 - \text{REACTIVE_PERCENTAGE}) / (\text{Number_Of_Subbehaviors} - \text{Number_Of_Reactive_Subbehaviors})$$

If the REACTIVE_PERCENTAGE is greater than the contributions of each individual non-reactive behaviors then this smaller contribution will be used instead. This is to prevent the reactive behaviors from dominating the other behaviors when many sub-behaviors are involved (10 if the reactive percentage is 10%). If this is the case, the calculations are adjusted in the following manner:

$$\begin{aligned} \text{contribution_percentage} &= 1.0 / (\text{Number_Of_Subbehaviors} - \text{Number_Of_Reactive_Subbehaviors} + 1) \\ \text{reactive_contribution_percentage} &= \text{contribution_percentage} / \text{Number_Of_Reactive_Subbehaviors} \end{aligned}$$

The total closeness is calculated by combining the behavior correlation and parameter correlation in the following fashion:

$$\text{total_closeness} = \text{PARAMETER_PERCENTAGE} * \text{parameter_correlation} + (1.0 - \text{PARAMETER_PERCENTAGE}) * \text{behavior_correlation}$$

There are several issues that arise from the correlation technique:

- When performing the IS-A metric, the HAS-A metric could be performed on the more general behavior. This was not done due to the small effects these results would have on the overall closeness versus the complexity and time introduced.

- A similar argument also applies to the SIBLING-OF metric.
- The IS-A metric could also be performed on SIBLING-OF tests to try to find a correlation if the sibling is not present or the SIBLING-OF metric could be performed on the IS-A tests. Again, the effect would be small.

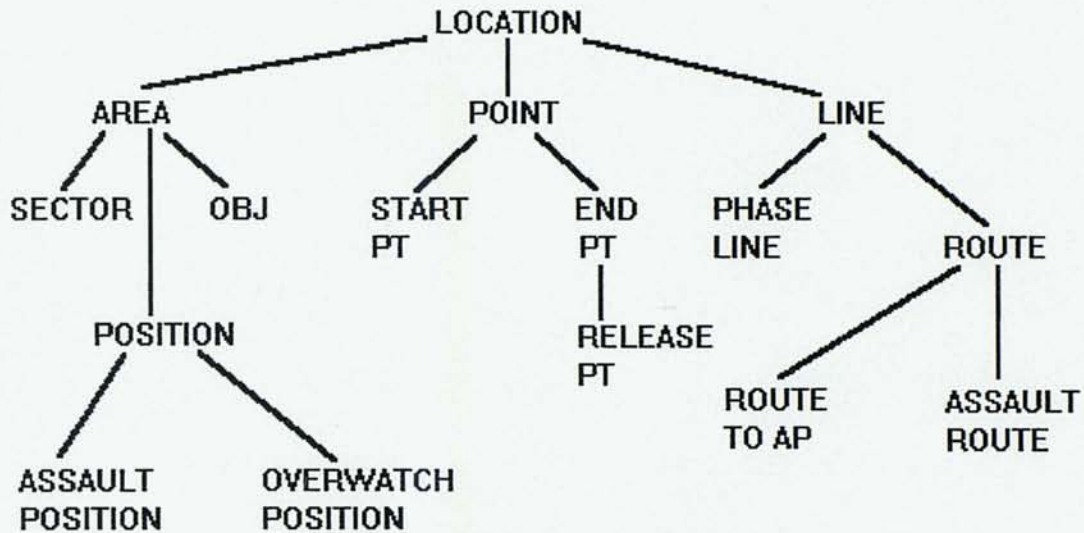


Figure 4. Partial Hierarchy for Behavior Parameters

The parameter correlation uses same routines as the behavior correlation each with a *parm_* prefix and modified for parameters and sub-parameters. The procedure *parm_correlate* attempts to correlate the source parameters with the destination parameters and then any uncorrelated destination parameters with the source parameters. The procedure *parm_Create_Closeness_Tree*, like its behavior counterpart, recursively calls *parm_Create_Closeness_Ply* for each sub-parameter and sums them together to form a single closeness value. The procedure *parm_Create_Closeness_Ply* performs the various metrics and chooses the maximum. It recursively calls *parm_Create_Closeness_Ply* to handle the next parameter on the same level of

decomposition. For the HAS-A metric, it recursively calls *parm_Create_Closeness_Tree* to see if the sub-parameters can be correlated.

The IS-A metric for parameters uses the relationships shown in Figure 4 and is composed of four separate procedures. The *Isa* procedure loops through each destination parameter to see if an inheritance path can be found between the source and a destination parameter, thus defining the conversion. It calls *CheckParent* to perform this function. The *CheckParent* routine loops through each destination parameter's parent until it is matched with a parent or child of the source behavior. To match with the parent, the *IsaParent* procedure is called. It determines if a source parameter is a parent of the destination parameter (or some parent of it). Similarly, to match with the child, the *IsaChild* routine determines if a source parameter is a child of the destination parameter. Both routines recursively call themselves until the match is found or the root (*IsaChild*) or a leaf (*IsaParent*) of the inheritance tree is reached.

The parameter correlation uses similar closeness adjustments and the same calculation methods for the IS-A and HAS-A (based upon sub-parameters instead of sub-behaviors) parameter metrics:

- PARM_HAS_A_ADJUSTMENT .05
- PARM_IS_A_ADJUSTMENT .10
- DEFAULT_ADJUSTMENT .25

The metric adjustment values for parameters are different than those of behaviors in order to reflect a different mind set. With parameter conversion, decomposing

parameters into its sub-parameters is easier than trying to convert a parameter to a more general or specific form so the closeness adjustment is smaller. The `DEFAULT_ADJUSTMENT` is used to reduce the parameter closeness when a parameter cannot be correlated and is a default parameter.

Each matched and unmatched parameter contributes equally to the parameter correlation. The number of parameter matches multiplied by each contribution percentage are summed to give the final parameter correlation value.

Parameter correlation is only performed for the top level source and destination behavior. It is not important that sub-parameters correlate since as long as the initial parameters correlate the behavior can be executed. Many times the sub-behavior parameters will be derived internally and thus have no correlatable counterparts. Unlike behavior correlation, the IS-A parameter correlation tries to find a specific-to-general then general-specific inheritance path to connect a source and destination parameter. Under behavior correlation, only the general and specific behaviors on the destination behaviors branch are checked as are the immediate siblings. More general or specific behaviors branching off a sibling behavior are not checked (until the behavior includes a common parent). It is assumed that this extra level of correlation would rarely be necessary and add very little to the closeness value. The HAS-A metric for parameter correlation is not too important in this domain since many structures are either decomposed in the ontology or are implementation dependent and thus not represented in the general ontology. The IS-A paths provide the primary conversion path.

The implementation of the parameter correlation has presented several issues:

- Complex data structures for parameters in some cases need to be broken up to avoid IS-A/HAS-A conflicts. For example, the CCTT SPEED is composed of CATCHUP-SPEED, DASH-SPEED, and MARCH-SPEED but needs to be an IS-A not HAS-A. If that were not the case then one speed could not be substituted for another. These conversions can be done when specific behaviors are converted to the general representation.
- Inheritance of parameter sub-parameters are not directly supported. Since few parameters have sub-parameters in the general representation for this proof of principle, the inherited sub-parameters are just duplicated in all descendant behaviors in the database.

Behavior Translation

The previous discussions have been dealing with CCTT and ModSAF behaviors represented in terms of a general representation. Before any of the correlation can be performed, however, specific behaviors must be translated into a form that provides a common language for interoperability between the two simulations. Thus when a TRAVEL behavior is being correlated and a TRAVEL destination behavior is found, the system can assign a higher closeness than if the behavior was unknown. As previously mentioned however, the system makes no assumptions about the behaviors being the same because they have the same name. The sub-behaviors are always checked to verify the closeness. The general representation serves as this common language. Examples of translations include converting specific-named behaviors to general names, removing redundant behaviors, breaking up aggregate parameter structures, etc. CCTT was used as

the model for this general representation since it has validated behaviors. Thus, only minor translations were needed for conversion to the general form. ModSAF behaviors, however, require more translation. Figure 5 shows the ModSAF assault behavior as defined by a ModSAF task frame. Figure 6 shows the corresponding behavior in terms of the general representation. The preparatory frame was removed since it is not specific to an assault, and several behaviors were combined and renamed. The developers of ModSAF decided to separate their mixed platoon behaviors (platoons with mechanized infantry, for example) from their homogeneous unit behaviors. The mixed behaviors are always assigned to units regardless. This distinction is not needed for correlation so the redundancy is removed. None of these translations are required, they only serve to enhance the correlation with some apriori knowledge about the systems being correlated. This can be done during run-time by a simple set of conversion rules.

```

HALT (Preparatory Frame)

UNIT_ASSAULT (Actual Frame)
  unit_mixed_travel
    unit_travel
      vehicle_move
    unit_follow_unit
      unit_travel
        vehicle_move
  unit_mixed_targeter
    unit_targeter
      vehicle_targeter
      vehicle_assess
      vehicle_search
  unit_mixed_prep_occupy_position
    unit_prep_occupy_position
      vehicle_occupy_position
  unit_occupy_position

```

Figure 5. ModSAF Assault

```

ASSAULT
  TRAVEL
    vehicle_MOVE
    vehicle_SEARCH
  FOLLOW_UNIT
    TRAVEL
      vehicle_MOVE
      vehicle_SEARCH
  TARGETER
    vehicle_SHOOT
    vehicle_ASSESS
    vehicle_SEARCH
  OCCUPY_POSITION
    vehicle_OCCUPY_POSITION

```

Figure 6. ModSAF Assault in General Form

Parameter Translation

A similar translation is done for behavior parameters as is done for behaviors.

Simulation specific translation code is used to rename parameters and decompose complex parameter data structures into individual parameters. Also, the translation must remove parameters that are known to implementation specific and thus are not a true attribute that defines the semantics of the behavior. Translation to a common parameter model is even more important than the translation of behaviors. If a completely common parameter model does not exist between simulations the parameter correlation may not be possible and thus the behavior correlation will not be possible.

CHAPTER 6

PROTOTYPE TESTING AND EVALUATION

This research focused on the correlation of tank platoon behaviors from CCTT to ModSAF. When interoperating CCTT and ModSAF platoons under a single task organization, the ModSAF units will receive orders from the CCTT company commander. The ModSAF units must execute these orders to the best of its ability. To simulate this situation, a CCTT assault behavior was assumed to be sent to a ModSAF unit which is correlated against the ModSAF behaviors and a behavior assigned.

To illustrate the various situations that can occur when correlating behaviors, the algorithm will first be tested on the correlation of two specially created test behaviors shown in Figure 7. These are adhoc behaviors, specially created to illustrate how the heuristic metrics can be applied.

ATTACK (SAF A)**Parameters:**

UNIT_ID	PLATFORM
MARCH_SPEED (Default)	ROUTE
FORMATION (Default)	

Behaviors:

TACTICAL_ROAD_MARCH
 TRAVEL
 vehicle_MOVE
 vehicle_SEARCH
 vehicle_ASSESS

OCCUPY_POSITION
 vehicle_MOVE
 vehicle_SEARCH
 vehicle_ASSESS

ATTACK (SAF B)**Parameters:**

UNIT_ID	PLATFORM
SPEED (Default)	START_POINT
RELEASE_POINT	SPACING (Default)

Behaviors:

TRAVEL
 CAUTIOUS_MOVE
 vehicle_MOVE
 vehicle_SEARCH
 vehicle_SEARCH
 vehicle_ASSESS

HASTY_OCCUPY_POSITION
 vehicle_MOVE
 vehicle_SEARCH
 vehicle_ASSESS

Figure 7. Example SAF Behaviors

This test case exhibits the following correlation situations:

- correlation of a more specific behavior (TACTICAL_ROAD_MARCH) with a more general behavior (TRAVEL).
- correlation of the same behaviors at different levels of decomposition (vehicle_SEARCH at levels 2 and 3 of TRAVEL).
- correlation of a general behavior (OCCUPY_POSITION) with a more specific behavior (HASTY_OCCUPY_POSITION).
- correlation of default source and destination parameters (FORMATION AND SPACING).
- correlation of a more specific parameter (MARCH_SPEED) with a more general parameter (SPEED).
- combinations of correlations of parameters involving HAS-A and IS-A relationships. For example, the source ROUTE is decomposed into a START_POINT and END_POINT which is correlated with a START_POINT and more specific RELEASE_POINT of the destination behavior.

Using a database of test behaviors and parameters, the attack behavior of SAF A (source) correlated with SAF B (destination) behaviors TRAVEL, TRAVEL-2, ATTACK, and PLATOON_DEFENSIVE_MISSION, with semantic closeness values of .586109, .612869, .755325, .521805, respectively. The SAF B Attack has the highest closeness (75.5%) and thus is chosen for correlation. The SAF A attack parameters were correlated with the SAF B attack parameters in the following fashion with their corresponding closeness values:

SAF A UNIT_ID with SAF B UNIT_ID (SC = 1.0)
 SAF A PLATFORM with SAF B PLATFORM (SC = 1.0)

SAF A MARCH_SPEED with SAF B SPEED (SC = 0.9)
SAF A ROUTE into:
 START_POINT with SAF B START_POINT
 END_POINT with SAF B RELEASE_POINT (SC = 0.9025)
SAF A FORMATION ignored (SC = 0.75)
SAF B SPACING defaulted (SC = 0.75)

This test case has shown that the algorithm can apply correctly the heuristic metrics on situations that are likely to be encountered in SAF behavior correlation. However, further tests are necessary using actual CCTT and ModSAF behaviors to show the effectiveness of the approach.

Comparison of CCTT and ModSAF Behaviors

Table 1 shows the initial comparison between CCTT and ModSAF tank behaviors. CCTT behaviors that do not have a connection to a ModSAF behavior do not have a ModSAF counterpart. ModSAF, while having a more robust behavior architecture, does not have more robust coverage. Since ModSAF was designed as a research system for battle labs for testing new equipment, the full suite of validated behaviors is not required.

The test results will show that behavior correlations already known a priori will be correlated by the algorithm. For CCTT behaviors with no ModSAF counterpart, the ModSAF behavior that best matches the CCTT behavior will be chosen for correlation.

Table 1.

CCTT AND MODSAF TANK PLATOON BEHAVIORS

CCTT TANK PLATOON BEHAVIORS	MODSAF TANK PLATOON BEHAVIORS
Action Drill	Assault
Actions At Obstacle	Assemble
Assault An Enemy Position	Attach
Assault Position Activities	Attack By Fire
Attack By Fire	Breach
Bounding Overwatch	Change Formation
Consolidate And Reorganize	Concealment
Displace To Subsequent Bp	Delay
Halt	Detach
Hasty Occupation Of Bp	Follow Vehicle
Occupy Assembly Area	Halt
Occupy Bp	Hasty Occupy Position
Passage Of Lines	Overwatch Movement
Platoon Defensive Mission	Plow Breach
Platoon Fire And Movement	Pursue
Resupply	Supply
Tactical Road March	Tactical Road March
Travel	Travel
Traveling Overwatch	Traveling Overwatch
	Withdraw

Proof of Principle

As a proof of principle, twelve CCTT behaviors will be correlated with one of twenty ModSAF behaviors. Seven of these behaviors will have expected pairings provided by subject matter experts. The remaining five will have no corresponding ModSAF behavior. The unknown correlation results are subject to interpretation since no agreed correlation already exists. Table 2 presents the correlations that will be tested via the experiments.

Table 2.

CCTT-MODSAF CORRELATIONS

CCTT BEHAVIOR	MODSAF BEHAVIOR
Assault an Enemy Position	Assault
Attack by Fire	Attack by Fire
Bounding Overwatch	Overwatch Movement
Tactical Road March	Tactical Road March
Travel	Travel
Hasty Occupy Position	Hasty Occupy Position
Traveling Overwatch	Traveling Overwatch
Occupy Bp	<i>unknown</i>
Passage of Lines	<i>unknown</i>
Platoon Defensive Mission	<i>unknown</i>
Platoon Fire and Movement	<i>unknown</i>
Consolidate and Reorganize	<i>unknown</i>

Testing ASSAULT and ATTACK BY FIRE will test the algorithms ability to discriminate between similar offensive actions. Testing BOUNDING OVERWATCH will test the algorithms ability to discriminate between several ModSAF forms of movement, namely TRAVEL, TACTICAL ROAD MARCH, OVERWATCH MOVEMENT, and TRAVELING OVERWATCH. A similar reason applies to TACTICAL ROAD MARCH. The ModSAF TACTICAL ROAD MARCH is not as robust and thus may not be determined to be the best correlation. Testing TRAVEL will set the lower bound for the test since this behavior exhibits a strong correlation to the ModSAF TRAVEL behavior.

CCTT and ModSAF Reactive Behaviors

Many of the behaviors used in the experiments have reactive behavior components which are compared with the reactive behaviors of the ModSAF behaviors. Rather than list these reactive behaviors for comparison with every experiment, they will be listed here and referenced as part of the behavior decomposition for the experiment behaviors in question. Figures 8-10 describes the ModSAF reactive behaviors and Figures 11-12 describes the CCTT reactive behaviors.

The Actions On Contact behavior involves the movement to contact with the enemy during offensive operations. Actions On Contact expresses the actions to be performed when a platoon makes unexpected contact (visually or by fire) with a moving or stationary enemy. The platoon may return fire, initiate a battle drill or seek cover and concealment, report the contact (spot report), perform fire and movement, or assault the enemy. In any case, a follow up spot report is usually sent to the company commander.


```

ACTIONS_ON_CONTACT
  ASSAULT
    OCCUPY_POSITION
      vehicle_ALTERNATE
      vehicle_MOVE
      vehicle_TERRAIN
      vehicle_SEARCH
    TARGETER
      vehicle_SHOOT
      vehicle_ASSESS
      vehicle_SEARCH
    TRAVEL
      FOLLOW_UNIT
        vehicle_MOVE
        vehicle_SEARCH
        vehicle_ENEMY
      vehicle_MOVE
      vehicle_ENEMY
      vehicle_SEARCH
  CONTACT_DRILL
    TARGETER
  WITHDRAW_DRILL
    WITHDRAW
      MOUNT
      vehicle_MOVE
      vehicle_SEARCH
      vehicle_ASSESS
      OCCUPY_POSITION
      ...
      vehicle_SMOKE
    TARGETER
  OCCUPY_POSITION_DRILL
    TARGETER
    ...
    OCCUPY_POSITION
    ...

```

Figure 8. ModSAF React to Enemy Contact Behavior

```

REACT_AIR
  SCATTER
    TARGETER
      vehicle_SHOOT
      vehicle_ASSESS
      vehicle_SEARCH
    TRAVEL
      FOLLOW_UNIT
        vehicle_MOVE
        vehicle_SEARCH
        vehicle_ENEMY
      vehicle_MOVE
      vehicle_ENEMY
      vehicle_SEARCH
    ACTIONS_ON_CONTACT
  TARGETER
  ...

```

Figure 9. ModSAF React to Air Attack Behavior

The React to Air Attack behavior describes the actions to take when a tank platoon comes under air attack. Upon contact with an enemy aircraft the platoon is to halt and, if attacked, seek covered and concealed positions. From these positions the tanks will shoot at the aircraft with various firing patterns (leading the aircraft, for example). An important distinction here is that the covered and concealed positions pertaining to aircraft are very different than those pertaining to ground forces. Cover may consist simply of a tree canopy while cover may be a hill or better yet a cave.

```

REACT_IF
  MINE_BREACH
    BREACH
      TRAVEL
        FOLLOW_UNIT
          vehicle_MOVE
          vehicle_SEARCH
          vehicle_ENEMY
        vehicle_MOVE
        vehicle_ENEMY
        vehicle_SEARCH

    OCCUPY_POSITION
      vehicle_ALTERNATE
      vehicle_MOVE
      vehicle_TERRAIN
      vehicle_SEARCH

    TARGETER
      vehicle_SEARCH
      vehicle_ASSESS
      vehicle_SHOOT

    ACTIONS_ON_CONTACT
  MINE_WITHDRAW_DRILL
    MINE_WITHDRAW
      vehicle_BACKTRACK
      vehicle_MOVE

    TARGETER
    ACTIONS_ON_CONTACT

```

Figure 10. ModSAF React to Indirect Fire Behavior

The React to Indirect Fire behavior describes actions to be taken when a platoon comes under artillery, mortar, or chemical attack. If the platoon is moving when attacked all vehicles maintain speed and direction while moving out of the attack area. If the platoon is stationary the tanks move to cover and concealed turret down positions (the

turrets or hulls cannot be hit) and wait until the attack ceases to continue their mission or move out of the impact area.

```

ACTIONS_ON_CONTACT
  ACTION_DRILL
    SEEK_COVER_AND_CONCEALMENT
      vehicle_SEARCH
    GENERATE_SITREP
      vehicle_OCCUPY_POSITION
      vehicle_MOVE
      vehicle_SEARCH
      vehicle_HIDE
      vehicle_HALT
      vehicle_MOVE
      vehicle_SEARCH
  CONTACT_DRILL
    vehicle_MOVE
    vehicle_SHOOT
  REACT_AIR
    SEEK_COVER_AND_CONCEALMENT
    ...
    vehicle_OCCUPY_POSITION
    ...
  REACT_TO_DI_ATTACK
    PLATOON_DEFENSIVE_MISSION (See Figure 29)
    ...
    SEEK_COVER_AND_CONCEALMENT
    ...
    vehicle_OCCUPY_POSITION
    ...
    CONSOLIDATE_AND_REORGANIZE (See Figure 25)
    ...

```

Figure 11. CCTT Actions on Contact Reactive Behavior

```
REACT_IF
  TRAVEL
    vehicle_MOVE
  SEEK_COVER_AND_CONCEALMENT
    vehicle_SEARCH
  vehicle_OCCUPY_POSITION
    vehicle_MOVE
    vehicle_SEARCH
    vehicle_HIDE
      vehicle_HALT
      vehicle_MOVE
      vehicle_SEARCH
```

Figure 12. CCTT React to Indirect Fire Reactive Behavior

Experiment 1

The first experiment involves correlating the CCTT Assault An Enemy Position behavior. A typical tank platoon assault behavior is concerned with issuing movement and firing commands to its vehicles. These commands instruct the vehicles to perform an on-line attack and occupy the position attacked. More specifically, the tank platoon closes with and destroys the enemy by overrunning and seizing the occupied enemy position. The tanks move rapidly in line formation under the cover from direct and indirect fire to the far side of the objective. Figure 13 shows the CCTT assault an enemy position behavior. CCTT is more robust than ModSAF in that it provides for an initial travel to the assault position (route_to_ap parameter), allows for the breach of obstacles along the way, and a consolidation and reorganization of forces after the assault has been completed.

CCTT ASSAULT AN ENEMY POSITION:

```

TRAVEL
    vehicle_MOVE
BOUNTING_OVERWATCH
    TRAVEL
        vehicle_MOVE
    vehicle_OCCUPY_POSITION
        vehicle_MOVE
        vehicle_SEARCH
        vehicle_HIDE
            vehicle_HALT
            vehicle_MOVE
            vehicle_SEARCH
SEEK_COVER_AND_CONCEALMENT
    vehicle_SEARCH
vehicle_OCCUPY_POSITION
...
GENERATE_REQUEST_FOR_INDIRECT_FIRE
CONSOLIDATE_AND_REORGANIZE
    SEEK_COVER_AND_CONCEALMENT
        vehicle_SEARCH
    vehicle_OCCUPY_POSITION
...
GENERATE_SITREP

```

Figure 13. CCTT Assault An Enemy Position

For the CCTT Assault An Enemy Position behavior, the following semantic closeness values were calculated for the ModSAF behaviors:

EXECUTE ASSAULT	0.522923
EXECUTE ASSEMBLE	0.264287
EXECUTE ATTACH	0.425562
EXECUTE ATTACK BY FIRE	0.39817
EXECUTE BREACH	0.431557
EXECUTE CHANGE FORMATION	0.265716
EXECUTE CONCEALMENT	0.401982
EXECUTE DELAY	0.419041
EXECUTE DETACH	0.425562

EXECUTE FOLLOW VEHICLE	0.0
EXECUTE HALT	0.264287
EXECUTE HASTY OCCUPY POSITION	0.377462
EXECUTE OVERWATCH MOVEMENT	0.43608
EXECUTE PLOW BREACH	0.431557
EXECUTE PURSUE	0.0
EXECUTE ROAD MARCH	0.422094
EXECUTE SUPPLY	0.400714
EXECUTE TRAVEL	0.422094
EXECUTE TRAVELING OVERWATCH	0.488249
EXECUTE WITHDRAW	0.409559

The highest correlation is with the ModSAF assault behavior with a semantic closeness of 52%. The actual closeness value is not so important as is the relative values between the different ModSAF behaviors. This pairing is the expected correlation. The semantic closeness values of zero represent cases where required ModSAF parameters could not be correlated. Figure 14 shows the ModSAF Assault behavior. The common primitives of `vehicle_MOVE` and `vehicle_SEARCH` (common to `OCCUPY_POSITION`) and the `TRAVEL` behavior are the primary reasons for the correct correlation. For similar reasons, the second and third choices (`TRAVELING_OVERWATCH` and `OVERWATCH_MOVEMENT`, respectively) exhibited high semantic closeness values. The presence of these primitives in several `OCCUPY_POSITION` behaviors offset some of the missing behaviors even though the positions being occupied are very different. The different positions are captured by the parameter correlation but their effect on the overall closeness is much smaller.

MODSAF ASSAULT:

```

EXECUTE_ASSAULT
  ASSAULT
    TRAVEL
      vehicle_MOVE
      vehicle_SEARCH
      vehicle_ENEMY
      FOLLOW_UNIT
        vehicle_MOVE
        vehicle_SEARCH
        vehicle_ENEMY
    TARGETER
      vehicle_SHOOT
      vehicle_ASSESS
      vehicle_SEARCH
    OCCUPY_POSITION
      vehicle_ALTERNATE
        vehicle_MOVE
      vehicle_TERRAIN
      vehicle_SEARCH

```

Figure 14. ModSAF Assault Behavior

The CCTT parameters were correlated with the ModSAF parameters in the following fashion with their corresponding closeness values:

```

CCTT UNIT_ID to ModSAF UNIT_ID (SC = 1.0)
CCTT PLATFORM to ModSAF PLATFORM (SC = 1.0)
CCTT ROUTE_TO_AP to ModSAF ROUTE (SC = 0.9)
CCTT ASSAULT_ROUTE to ModSAF ROUTE (SC = 0.9)
CCTT ENEMY_POSITION to POSITION to AREA to ModSAF OBJECTIVE
(SC = 0.729)
CCTT TRIGGER LINE to LINE to ModSAF ROUTE (SC = 0.81)
CCTT ASSAULT_POSITION to POSITION to AREA to ModSAF OBJECTIVE
(SC = 0.729)
CCTT DEPARTURE_TIME to NO MATCH (SC = 0.0)
CCTT OBSTACLE defaulted (SC = 0.9)
CCTT BREACH_ROUTE to ModSAF ROUTE (SC = 0.9)
CCTT PRE-BREACH_ROUTE to ModSAF ROUTE (SC = 0.9)

```


CCTT POST-BREACH ROUTE to ModSAF ROUTE (SC = 0.9)
CCTT ALPHA_SECTION ignored (SC = 0.75)
CCTT BRAVO_SECTION ignored (SC = 0.75)
ModSAF LEFT_TACTICAL_BOUNDARY defaulted (SC = 0.75)
ModSAF RIGHT TACTICAL BOUNDARY defaulted (SC = 0.75)
ModSAF SPEED defaulted (SC = 0.75)
ModSAF DISMOUNTED_SPEED defaulted (SC = 0.75)
ModSAF STOPPING_ASSAULT_CRITERIA defaulted (SC = 0.75)
ModSAF SECURE_OBJECTIVE_FLAG defaulted (SC = 0.75)
ModSAF FORMATION defaulted (SC = 0.75)
ModSAF SPACING defaulted (SC = 0.75)
ModSAF X_DI_OFFSET defaulted (SC = 0.75)
ModSAF Y_DI_OFFSET defaulted (SC = 0.75)
ModSAF ASSAULT_REASON defaulted (SC = 0.75)
ModSAF DI_FORMATION defaulted (SC = 0.75)

The results agree with the predictions with one exception that illustrates one inherent problem with the parameter correlation. Destination parameters that are equally related to more than one source parameter cause an ambiguity as to which parameter correlation is the correct one. In this experiment there are five equally related source routes and only one destination route. We know that the ASSAULT_ROUTE is the best correlation but it is unclear as to how the algorithm can determine this automatically. Correlating in the other direction, a single source behavior can be matched against more than one destination behavior. In some cases this may be satisfactory but in other cases it may cause unexpected results and thus the destination parameters should have been allowed to default. Some a priori knowledge code may need to be used to modify the parameter correlation for known problems before assigning the behavior. As an example, code can be used that will check to see if all the routes are the same and if they are, default all the routes except the assault route. Also, the best correlations should take precedence over

lesser correlations such as the TRIGGER_LINE in this case. The CCTT TRIGGER_LINE should be ignored since there are better ROUTE correlations. This is a trivial task that can be done when the actual parameter conversions are done. The ordering of the parameters may also be used to specify a priority as a conflict resolution scheme. However this may not always be correct when the simulations being correlated is determined at run time.

Experiment 2

The Attack By Fire behavior is commonly used when a tank platoon is outnumbered by the enemy, the enemy has anti-tank capability, or the platoon is to provide cover/supporting fire in support of another force. The behavior is characterized by the platoon occupying covered and concealed positions and shooting at the enemy without attempting to engage and assault the enemy. Figure 15 shows the decomposition for the CCTT Attack By Fire behavior.

CCTT ATTACK BY FIRE

```

EXECUTE_ATTACK_BY_FIRE
  ACTIONS_ON_CONTACT
  ATTACK_BY_FIRE
    SEEK_COVER_AND_CONCEALMENT
      vehicle_SEARCH
    vehicle_OCCUPY_POSITION
      vehicle_MOVE
      vehicle_SEARCH
      vehicle_HIDE
        vehicle_HALT
        vehicle_MOVE
        vehicle_SEARCH
    vehicle_ENGAGE_ENEMY
      vehicle_MOVE
      vehicle_TARGETER
        vehicle_SEARCH
        vehicle_SHOOT
  GENERATE_SITREP

```

Figure 15. CCTT Attack By Fire Behavior

For the CCTT Attack By Fire behavior, the following semantic closeness values were calculated for the ModSAF behaviors:

EXECUTE ASSAULT	0.511527
EXECUTE ASSEMBLE	0.230692
EXECUTE ATTACH	0.438184
EXECUTE ATTACK BY FIRE	0.607225
EXECUTE BREACH	0.435787
EXECUTE CHANGE FORMATION	0.273721
EXECUTE CONCEALMENT	0.420325
EXECUTE DELAY	0.500338
EXECUTE DETACH	0.438184
EXECUTE FOLLOW VEHICLE	0.0
EXECUTE HALT	0.230692
EXECUTE HASTY OCCUPY POSITION	0.438654
EXECUTE OVERWATCH MOVEMENT	0.433975
EXECUTE PLOW BREACH	0.435787

EXECUTE PURSUE	0.0
EXECUTE ROAD MARCH	0.435919
EXECUTE SUPPLY	0.435263
EXECUTE TRAVEL	0.435919
EXECUTE TRAVELING OVERWATCH	0.47883
EXECUTE WITHDRAW	0.424878

The results show that the algorithm did indeed determine the ModSAF Attack By Fire to be the best correlation, with a semantic closeness of 60%. When comparing the CCTT and ModSAF Attack By Fire behaviors, the common TARGETER and OCCUPY_POSITION sub-behaviors contribute to the high correlation. Alternative choices such as Assault and Delay are similar to Attack By Fire because they also involve shooting at the enemy and occupying positions.

MODSAF ATTACK BY FIRE

```

EXECUTE_ATTACK_BY_FIRE
  ATTACK_BY_FIRE
    TARGETER
      vehicle_SHOOT
      vehicle_ASSESS
      vehicle_SEARCH
    OCCUPY_POSITION
      vehicle_ALTERNATE
      vehicle_MOVE
      vehicle_TERRAIN
      vehicle_SEARCH
  REACT_AIR

```

Figure 16. ModSAF Attack By Fire Behavior

The CCTT parameters were correlated with the ModSAF parameters in the following fashion with their corresponding closeness values:

CCTT UNIT_ID to ModSAF UNIT_ID (SC = 1.0)
 CCTT PLATFORM to ModSAF PLATFORM (SC = 1.0)
 CCTT OVERWATCH_POSITION to ModSAF OVERWATCH_POSITION
 (SC = 1.0)
 CCTT ENEMY_LOCATION to LOCATION to AREA to ModSAF
 ENGAGEMENT_AREA (SC = 0.729)
 ModSAF BATTLE_POSITION to POSITION to CCTT
 OVERWATCH_POSITION (SC = 0.81)
 ModSAF LEFT_TRP defaulted (SC = 0.75)
 ModSAF RIGHT_TRP defaulted (SC = 0.75)
 ModSAF SPEED defaulted (SC = 0.75)

The parameter correlation for these behaviors is straightforward with extra ModSAF parameters being allowed to take on their default values. Note that the CCTT OVERWATCH_POSITION provides not only the ModSAF OVERWATCH_POSITION but the ModSAF battle position as well. Since there is no other CCTT POSITION to use, the ModSAF platoon must occupy the same positions and attack the enemy by fire in the ENGAGEMENT_AREA derived from the CCTT ENEMY_LOCATION.

Experiment 3

The Bounding Overwatch behavior is primarily characterized by a platoon splitting up into alpha and bravo sections, each taking turns providing cover. It is the slowest form of movement but is very useful when traveling in an environment where enemy attack is likely. When one section is moving, the other section maintains a line-of-sight with the moving section from an overwatch position. When the moving section reaches its

overwatch position, the other section moves in a similar fashion. The bounding technique may be alternating or successive. In alternating bounding overwatch, one section moves to its overwatch position and the other section then moves to the same position and this process repeats. For successive bounding overwatch, moving sections will “leapfrog” the other section’s position to another overwatch position further forward with this process being repeated. Figure 17 shows the CCTT Bounding Overwatch and its sub-behavior components.

CCTT BOUNDING OVERWATCH

```

EXECUTE_BOUNDING_OVERWATCH
  ACTIONS_ON_CONTACT
  REACT_IF
  BOUNDING_OVERWATCH
    SECTION_TRAVEL
      vehicle_MOVE
    HALT
    vehicle_MOVE
    SEEK_COVER_AND_CONCEALMENT
      vehicle_SEARCH
    vehicle_OCCUPY_POSITION
      vehicle_MOVE
      vehicle_SEARCH
      vehicle_HIDE
        vehicle_HALT
        vehicle_MOVE
        vehicle_SEARCH
  
```

Figure 17. CCTT Bounding Overwatch Behavior

For the CCTT Bounding Overwatch behavior, the following semantic closeness values were calculated for the ModSAF behaviors:

EXECUTE ASSAULT	0.552553
EXECUTE ASSEMBLE	0.358629
EXECUTE ATTACH	0.506746
EXECUTE ATTACK BY FIRE	0.0
EXECUTE BREACH	0.51186
EXECUTE CHANGE FORMATION	0.297466
EXECUTE CONCEALMENT	0.43693
EXECUTE DELAY	0.498551
EXECUTE DETACH	0.506746
EXECUTE FOLLOW VEHICLE	0.0
EXECUTE HALT	0.358629
EXECUTE HASTY OCCUPY POSITION	0.0
EXECUTE OVERWATCH MOVEMENT	0.554897
EXECUTE PLOW BREACH	0.51186
EXECUTE PURSUE	0.0
EXECUTE ROAD MARCH	0.542453
EXECUTE SUPPLY	0.507796
EXECUTE TRAVEL	0.542453
EXECUTE TRAVELING OVERWATCH	0.51186
EXECUTE WITHDRAW	0.484894

The ModSAF Overwatch Movement behavior received the highest correlation (55%) as expected. When comparing the CCTT behavior with that of ModSAF OVERWATCH_MOVEMENT in Figure 18, one can see that the SECTION_TRAVEL and OCCUPY_POSITION behaviors along with common reactive behaviors contributed to its high semantic closeness. The ModSAF Assault behavior was a close second, mainly due to differences in the travel behavior and different parameter correlations.

MODSAF OVERWATCH MOVEMENT

```

EXECUTE_OVERWATCH_MOVEMENT
  OVERWATCH_MOVEMENT
    FOLLOW_UNIT
      vehicle_MOVE
      vehicle_SEARCH
      vehicle_ENEMY
    SECTION_TRAVEL
      vehicle_MOVE
    HALT
      DI_HALT
      vehicle_MOVE
    OCCUPY_POSITION
      vehicle_ALTERNATE
        vehicle_MOVE
      vehicle_TERRAIN
      vehicle_SEARCH
  REACT_AIR
  REACT_IF
  ACTIONS_ON_CONTACT

```

Figure 18. ModSAF Overwatch Movement Behavior

The CCTT parameters were correlated with the ModSAF parameters in the following fashion with their corresponding closeness values:

```

CCTT UNIT_ID to ModSAF UNIT_ID (SC = 1.0)
CCTT PLATFORM to ModSAF PLATFORM (SC = 1.0)
CCTT ALPHA SECTION ignored (SC = 0.75)
CCTT BRAVO SECTION ignored (SC = 0.75)
CCTT ROUTE to ModSAF ROUTE (SC = 1.0)
CCTT START_POINT to NO MATCH (SC = 0.0)
CCTT OVERWATCH_POSITION to NO MATCH (SC = 0.0)
CCTT BOUNDING_TECHNIQUE to ModSAF BOUNDING_TECHNIQUE
(SC = 1.0)
CCTT ENEMY_DIRECTION to NO MATCH (SC = 0.0)
ModSAF SPEED defaulted (SC = 0.75)
ModSAF DISMOUNTED_SPEED defaulted (SC = 0.75)
ModSAF MARCH_SPEED defaulted (SC = 0.75)

```


ModSAF SPACING defaulted (SC = 0.75)
ModSAF FORMATION defaulted (SC = 0.75)
ModSAF DI_FORMATION (SC = 0.75)
ModSAF CONFORM_TO_TERRAIN_FLAG (SC = 0.75)
ModSAF X_DI_OFFSET (SC = 0.75)
ModSAF Y_DI_OFFSET (SC = 0.75)

While the ModSAF and CCTT Bounding Overwatch behaviors are similar from a behavior standpoint, the parameters do not share many commonalities. Most CCTT and ModSAF parameters are ignored and allowed to take on default values. The ModSAF Bounding Overwatch is not as robust as CCTT but does provide for mixed units as seen by the dismounted infantry (DI) parameters that are defaulted.

Experiment 4

Traveling overwatch is similar to bounding overwatch but is a faster form of movement. It is useful for environments where enemy contact is probable but not too likely. The section in front maintains a constant speed while the rear section moves from rear overwatch position to overwatch position to cover the leading section. Figure 19 shows the CCTT Traveling Overwatch behavior.

CCTT TRAVELING OVERWATCH

```

EXECUTE_TRAVELING_OVERWATCH
  TRAVELING_OVERWATCH
    SECTION_TRAVEL
      vehicle_MOVE
    SEEK_COVER_AND_CONCEALMENT
      vehicle_SEARCH
    vehicle_OCCUPY_POSITION
      vehicle_MOVE
      vehicle_SEARCH
      vehicle_HIDE
        vehicle_HALT
        vehicle_MOVE
        vehicle_SEARCH
  
```

Figure 19. CCTT Traveling Overwatch Behavior

For the CCTT Traveling Overwatch behavior, the following semantic closeness values were calculated for the ModSAF behaviors:

EXECUTE ASSAULT	0.643822
EXECUTE ASSEMBLE	0.310528
EXECUTE ATTACH	0.52054
EXECUTE ATTACK BY FIRE	0.552476
EXECUTE BREACH	0.518223
EXECUTE CHANGE FORMATION	0.275766
EXECUTE CONCEALMENT	0.509361
EXECUTE DELAY	0.550527
EXECUTE DETACH	0.52054
EXECUTE FOLLOW VEHICLE	0.0
EXECUTE HALT	0.310528
EXECUTE HASTY OCCUPY POSITION	0.515839
EXECUTE OVERWATCH MOVEMENT	0.5807
EXECUTE PLOW BREACH	0.518223
EXECUTE PURSUE	0.0
EXECUTE ROAD MARCH	0.554991
EXECUTE SUPPLY	0.520992

EXECUTE TRAVEL	0.554991
EXECUTE TRAVELING OVERWATCH	0.744768
EXECUTE WITHDRAW	0.506122

In this case, the algorithm was able to distinguish similar forms of movement such as OVERWATCH_MOVEMENT and TRAVELING_OVERWATCH by choosing to correlate the ModSAF TRAVELING_OVERWATCH with the CCTT TRAVELING_OVERWATCH behavior with a semantic closeness of 75%. It was also able to distinguish between the second place finisher, Assault, mostly due to parameter mismatches and the lack of shooting behavior. Figure 20 shows the ModSAF Traveling Overwatch behavior.

MODSAF TRAVELING OVERWATCH

```

EXECUTE_TRAVELING_OVERWATCH
  TRAVELING_OVERWATCH
    OCCUPY_POSITION
      vehicle_ALTERNATE
        vehicle_MOVE
      vehicle_TERRAIN
      vehicle_SEARCH
    SECTION_TRAVEL
      vehicle_MOVE
  REACT_AIR
  REACT_IF
  ACTIONS_ON_CONTACT

```

Figure 20. ModSAF Traveling Overwatch Behavior

The CCTT parameters were correlated with the ModSAF parameters in the following fashion with their corresponding closeness values:

CCTT UNIT_ID to ModSAF UNIT_ID (SC = 1.0)
CCTT PLATFORM to ModSAF PLATFORM (SC = 1.0)
CCTT ROUTE to ModSAF ROUTE (SC = 1.0)
CCTT LEAD_POSITION to NO MATCH (SC = 0.0)
CCTT OVERWATCH_POSITION to NO MATCH (SC = 0.0)
CCTT ENEMY_DIRECTION to NO_MATCH (SC = 0.0)
CCTT LEAD_SECTION ignored (SC = 0.75)
CCTT OVERWATCH_SECTION ignored (SC = 0.75)
ModSAF SPEED defaulted (SC = 0.75)
ModSAF MARCH_SPEED defaulted (SC = 0.75)
ModSAF FORMATION defaulted (SC = 0.75)
ModSAF CONFORM_TO_TERRAIN_FLAG defaulted (SC = 0.75)
ModSAF SPACING defaulted (SC = 0.75)
ModSAF FOLLOW_DISTANCE defaulted (SC = 0.75)

The CCTT Traveling Overwatch behavior allows the ordering superior unit (or trainee) to be more flexible in assigning and positioning the sections of the platoon for movement.

ModSAF does not provide this flexibility.

Experiment 5

A Tactical Road March is normally used to move platoons from rear areas to front line assembly areas in preparation for a mission. This movement is usually rapid and usually along roads. It is conducted at a fixed march speed, often with fixed time intervals and halt or check points. The chance of enemy contact is minimal and thus is reflected in the movement. Figure 21 shows the CCTT Tactical Road March behavior.

CCTT TACTICAL ROAD MARCH

```

EXECUTE_TACTICAL_ROAD_MARCH
  ACTIONS_ON_CONTACT
  REACT_IF
  TACTICAL_ROAD_MARCH
    TRAVEL
      vehicle_MOVE
    HALT
  GENERATE_SITREP

```

Figure 21. CCTT Tactical Road March Behavior

For the CCTT Tactical Road March behavior, the following semantic closeness

values were calculated for the ModSAF behaviors:

EXECUTE ASSAULT	0.0
EXECUTE ASSEMBLE	0.39629
EXECUTE ATTACH	0.50278
EXECUTE ATTACK BY FIRE	0.0
EXECUTE BREACH	0.51583
EXECUTE CHANGE FORMATION	0.317307
EXECUTE CONCEALMENT	0.362746
EXECUTE DELAY	0.0
EXECUTE DETACH	0.502738
EXECUTE FOLLOW VEHICLE	0.0
EXECUTE HALT	0.39629
EXECUTE HASTY OCCUPY POSITION	0.0
EXECUTE OVERWATCH MOVEMENT	0.49608
EXECUTE PLOW BREACH	0.51583
EXECUTE PURSUE	0.0
EXECUTE ROAD MARCH	0.477789
EXECUTE SUPPLY	0.453126
EXECUTE TRAVEL	0.477789
EXECUTE TRAVELING OVERWATCH	0.49128
EXECUTE WITHDRAW	0.410459

This case demonstrates the only failure of the correlation algorithm with the expected correlations. The algorithm chooses the ModSAF Breach behavior as the best correlation with the CCTT Tactical Road March. However, the reason for this miscorrelation lies with the ModSAF Road March behavior itself, not the correlation algorithm. The ModSAF Road March behavior is not very robust and is virtually identical to the generic ModSAF Travel behavior. Because of this and the fact that the ModSAF Breach behavior does not specify a required obstacle parameter, the Breach behavior in Figure 22 demonstrates the best semantic closeness of 52%. When examining the two behaviors on a purely behavior standpoint, they both have similar Travel and Occupy Position behaviors (the Tactical Road March occupies positions when it reaches halt or check points and the Breach behavior occupies the obstacle position), even though these positions are different.

MODSAF BREACH

EXECUTE_BREACH

BREACH

TRAVEL

vehicle_MOVE

vehicle_SEARCH

vehicle_ENEMY

FOLLOW_UNIT

vehicle_MOVE

vehicle_SEARCH

vehicle_ENEMY

OCCUPY_POSITION

vehicle_ALTERNATE

vehicle_MOVE

vehicle_TERRAIN

vehicle_SEARCH

Figure 22. ModSAF Breach Behavior

The CCTT parameters were correlated with the ModSAF parameters in the following fashion with their corresponding closeness values:

CCTT UNIT_ID to ModSAF UNIT_ID (SC = 1.0)
CCTT PLATFORM to ModSAF PLATFORM (SC = 1.0)
CCTT SPEED ignored (SC = 0.75)
CCTT ROUTE to ModSAF BREACH_ROUTE (SC = 0.9)
CCTT FORMATION ignored (SC = 0.75)
CCTT SPACING ignored (SC = 0.75)
CCTT START_POINT to NO MATCH (SC = 0.0)
CCTT RELEASE_POINT to NO MATCH (SC = 0.0)
CCTT CHECK_POINT ignored (SC = 0.75)
CCTT HALT_POINT ignored (SC = 0.75)
CCTT HALT_INTERVAL ignored (SC = 0.75)
CCTT HALT_TIME (SC = 0.75)

A clue to the behavior miscorrelation can be seen by the amount of ignored and defaulted parameters in the parameter correlation. If the ModSAF Breach behavior had a required obstacle parameter then the Breach behavior would have been eliminated from consideration.

Experiment 6

Travel is the most simple tank platoon behavior. Similar to Tactical Road March it is used when enemy contact is not likely and when speed is needed. It is also used as part of other behaviors such as Traveling Overwatch and Bounding Overwatch. Traveling is characterized by continuous movement in a specified formation as fast as the METT-T factors will allow. Figure 23 shows the CCTT Execute Traveling behavior.

CCTT EXECUTE TRAVELING

```
EXECUTE_TRAVEL
  ACTIONS_ON_CONTACT
  REACT_IF
  TRAVEL
    vehicle_MOVE
```

Figure 23. CCTT Travel Behavior

For the CCTT Execute Traveling behavior, the following semantic closeness values were calculated for the ModSAF behaviors:

EXECUTE ASSAULT	0.0
EXECUTE ASSEMBLE	0.6913
EXECUTE ATTACH	0.569824
EXECUTE ATTACK BY FIRE	0.0
EXECUTE BREACH	0.58006
EXECUTE CHANGE FORMATION	0.578048
EXECUTE CONCEALMENT	0.540548
EXECUTE DELAY	0.0
EXECUTE DETACH	0.569824
EXECUTE FOLLOW VEHICLE	0.0
EXECUTE HALT	0.6913
EXECUTE HASTY OCCUPY POSITION	0.0
EXECUTE OVERWATCH MOVEMENT	0.595329
EXECUTE PLOW BREACH	0.58006
EXECUTE PURSUE	0.0
EXECUTE ROAD MARCH	0.773357
EXECUTE SUPPLY	0.757975
EXECUTE TRAVEL	0.899357
EXECUTE TRAVELING OVERWATCH	0.60456
EXECUTE WITHDRAW	0.677486

Upon inspection of the corresponding ModSAF Travel behavior in Figure 24, it is obvious that the behaviors are quite similar. This is expected as the behaviors become more and

more simple and primitive. As mentioned previously, many common primitives are used to define behaviors in the same domain. Thus the CCTT Travel behavior correlates with the ModSAF Travel behavior with a semantic closeness of 90%.

MODSAF TRAVEL

EXECUTE_TRAVEL

TRAVEL

vehicle_MOVE

vehicle_SEARCH

vehicle_ENEMY

FOLLOW_UNIT

vehicle_MOVE

vehicle_SEARCH

vehicle_ENEMY

REACT_AIR

REACT_IF

ACTIONS_ON_CONTACT

Figure 24. ModSAF Travel Behavior

The CCTT parameters were correlated with the ModSAF parameters in the following fashion with their corresponding closeness values:

CCTT UNIT_ID to ModSAF UNIT_ID (SC = 1.0)
 CCTT PLATFORM to ModSAF PLATFORM (SC = 1.0)
 CCTT SPEED to ModSAF SPEED (SC = 1.0)
 CCTT ROUTE to ModSAF ROUTE (SC = 1.0)
 CCTT FORMATION to ModSAF FORMATION (SC = 1.0)
 CCTT SPACING to ModSAF SPACING (SC = 1.0)
 ModSAF LEFT_TACTICAL_BOUNDARY defaulted (SC = 0.75)
 ModSAF RIGHT_TACTICAL_BOUNDARY defaulted (SC = 0.75)
 ModSAF MARCH_SPEED to CCTT SPEED (SC = 0.9)
 ModSAF DISMOUNTED_SPEED to CCTT SPEED (SC = 0.9)
 ModSAF DI_FORMATION to CCTT FORMATION (SC = 0.9)
 ModSAF CONFORM_TO_TERRAIN_FLAG defaulted (SC = 0.75)

ModSAF X_DI_OFFSET to DISTANCE to CCTT SPACING (SC = 0.81)
ModSAF Y_DI_OFFSET to DISTANCE to CCTT SPACING (SC = 0.81)

The parameter correlation is satisfactory with the only questionable correlation being the CCTT SPACING with the X and Y DI_OFFSET parameters. The parameter ontology may need to be modified to reflect a larger semantic difference between tank spacing distances and dismounted infantry distances from their vehicles. In this case it does not make any difference since the experiments are dealing with tank platoons not mixed units.

Experiment 7

Consolidate and Reorganize is a broad behavior that specifies actions to be performed after an assault or enemy assault is defeated. In the consolidation phase an objective is secured and defended against counterattacks through various means usually involving specific defensive formation patterns. Each platoon supports one another in this effort. The reorganization phase prepares units for continued actions by evacuating casualties, conducting equipment maintenance, and redistributing personnel, supplies, and equipment. Figure 25 shows the CCTT Consolidate and Reorganize behavior.

CCTT CONSOLIDATE AND REORGANIZE

```

EXECUTE_CONSOLIDATE_AND_REORGANIZE
  ACTIONS_ON_CONTACT
  REACT_IF
  CONSOLIDATE_AND_REORGANIZE
    SEEK_COVER_AND_CONCEALMENT
      vehicle_SEARCH
    vehicle_OCCUPY_POSITION
      vehicle_MOVE
      vehicle_SEARCH
      vehicle_HIDE
        vehicle_HALT
        vehicle_MOVE
        vehicle_SEARCH
  GENERATE_SITREP

```

Figure 25. CCTT Consolidate and Reorganize Behavior

For the CCTT Consolidate and Reorganize behavior, the following semantic closeness values were calculated for the ModSAF behaviors:

EXECUTE ASSAULT	0.447875
EXECUTE ASSEMBLE	0.25383
EXECUTE ATTACH	0.479121
EXECUTE ATTACK BY FIRE	0.452866
EXECUTE BREACH	0.487322
EXECUTE CHANGE FORMATION	0.31191
EXECUTE CONCEALMENT	0.450362
EXECUTE DELAY	0.489362
EXECUTE DETACH	0.479121
EXECUTE FOLLOW VEHICLE	0.0
EXECUTE HALT	0.25383
EXECUTE HASTY OCCUPY POSITION	0.419482
EXECUTE OVERWATCH MOVEMENT	0.448852
EXECUTE PLOW BREACH	0.487322
EXECUTE PURSUE	0.0
EXECUTE ROAD MARCH	0.450368
EXECUTE SUPPLY	0.479528
EXECUTE TRAVEL	0.450368

EXECUTE TRAVELING OVERWATCH	0.457251
EXECUTE WITHDRAW	0.468943

There is no corresponding ModSAF Consolidate and Reorganize behavior so the best match must be chosen. Based upon examination of the ModSAF behaviors in Figure 26 with the CCTT behavior, the common occupy position behaviors between the CCTT Consolidate and Reorganize and ModSAF Delay provided the best correlation of 49%. The CCTT Consolidate and Reorganize is typically used after an attack to occupy a position and consolidate resources. The ModSAF Delay is also used at the end of an attack to allow friendly forces to escape. The remaining platoon occupies a position and fires at the enemy, hoping to delay the opposing force long enough to allow the remainder of the friendly force to escape. This correlation may not make much sense but does illustrate the problem that can occur when interoperating heterogeneous simulations with different behaviors.

MODSAF DELAY

```

EXECUTE_DELAY
  MOUNT
    HALT
      DI_HALT
      vehicle_MOVE
    PICKUP
      vehicle_MOUNT
      vehicle_MOVE
  WITHDRAW
    MOUNT
    vehicle_MOVE
    vehicle_SEARCH
    vehicle_ASSESS
    OCCUPY_POSITION
      vehicle_ALTERNATE
      vehicle_MOVE
      vehicle_TERRAIN
      vehicle_SEARCH
    vehicle_SMOKE

```

Figure 26. ModSAF Delay Behavior

The CCTT parameters were correlated with the ModSAF parameters in the following fashion with their corresponding closeness values:

```

CCTT UNIT_ID to CCTT UNIT_ID (SC = 1.0)
CCTT PLATFORM to CCTT PLATFORM (SC = 1.0)
CCTT CONSOLIDATE_LOCATION to LOCATION to AREA to POSITION to
  BATTLE_POSITION (SC = 0.6561)

```

There are not many parameters to correlate with these two behaviors. The

CONSOLIDATE_LOCATION is correlated with the ModSAF BATTLE_POSITION

even though these do have very different semantic meanings. This is reflected in the low closeness value of 65%.

Experiment 8

Occupy Battle Position is characterized by finding attack/defense positions around some battle position. Implicit is adequate time to dig or find hull down or turret down positions. Hull down positions prevent the hull of the tank to be hit while still allowing the turret to shoot or be hit. Turret down positions do not allow the turret to be hit or the turret to shoot. Ideally, the tanks would like to move into a hull down position, shoot, and then return to a turret down position. Figure 27 shows the CCTT Occupy Battle Position behavior.

CCTT OCCUPY BATTLE POSITION

```

EXECUTE_OCCUPY_BATTLE_POSITION
  ACTIONS_ON_CONTACT
  REACT_IF
  OCCUPY_BATTLE_POSITION
    SEEK_COVER_AND_CONCEALMENT
      vehicle_SEARCH
    TRAVEL
      vehicle_MOVE
    HALT
  vehicle_OCCUPY_POSITION
    vehicle_MOVE
    vehicle_SEARCH
    vehicle_HIDE
      vehicle_HALT
      vehicle_MOVE
      vehicle_SEARCH

```

Figure 27. CCTT Occupy Battle Position Behavior

For the CCTT Occupy Battle Position behavior, the following semantic closeness values were calculated for the ModSAF behaviors:

EXECUTE ASSAULT	0.589559
EXECUTE ASSEMBLE	0.374152
EXECUTE ATTACH	0.552605
EXECUTE ATTACK BY FIRE	0.538714
EXECUTE BREACH	0.551146
EXECUTE CHANGE FORMATION	0.276081
EXECUTE CONCEALMENT	0.468669
EXECUTE DELAY	0.564026
EXECUTE DETACH	0.552605
EXECUTE FOLLOW VEHICLE	0.0
EXECUTE HALT	0.374152
EXECUTE HASTY OCCUPY POSITION	0.510893
EXECUTE OVERWATCH MOVEMENT	0.564983
EXECUTE PLOW BREACH	0.551146
EXECUTE PURSUE	0.0
EXECUTE ROAD MARCH	0.528196
EXECUTE SUPPLY	0.513598
EXECUTE TRAVEL	0.528196
EXECUTE TRAVELING OVERWATCH	0.563408
EXECUTE WITHDRAW	0.504522

The ModSAF Assault behavior (Figure 14) is a common behavior that is often adapted and expanded to provide more complex or different attack behaviors. It is this fact that explains the resulting correlations with many of the unknown CCTT behaviors that are assault variants. This is the situation in this case with the CCTT Occupy Battle Position being correlated with the ModSAF Assault with a semantic closeness of 59%. The Occupy Position sub-behavior found in Assault is present in any attack behavior.

The CCTT parameters were correlated with the ModSAF parameters in the following fashion with their corresponding closeness values:

CCTT UNIT_ID to ModSAF UNIT_ID (SC = 1.0)
 CCTT PLATFORM to ModSAF PLATFORM (SC = 1.0)
 CCTT BATTLE_POSITION to POSITION to AREA to ModSAF OBJECTIVE
 (SC = 0.729)
 CCTT ROUTE_TO_REAR_OF_BP to ModSAF ROUTE (SC = 0.9)
 CCTT HIDE_POSITION ignored (SC = 0.75)
 CCTT AVENUES_OF_APPROACH ignored (SC = 0.75)
 CCTT PLATOON_POSITION ignored (SC = 0.75)
 CCTT SECTOR to AREA to ModSAF OBJECTIVE (SC = 0.81)
 CCTT ENGAGEMENT_AREA to AREA to OBJECTIVE (SC = 0.81)
 CCTT END_DEFENSE_TIME ignored (SC = 0.75)
 ModSAF LEFT_TACTICAL_BOUNDARY defaulted (SC = 0.75)
 ModSAF RIGHT_TACTICAL_BOUNDARY defaulted (SC = 0.75)
 ModSAF SPEED defaulted (SC = 0.75)
 ModSAF DISMOUNTED_SPEED defaulted (SC = 0.75)
 ModSAF STOPPING_ASSAULT_CRITERIA defaulted (SC = 0.75)
 ModSAF SECURE_OBJECTIVE_FLAG defaulted (SC = 0.75)
 ModSAF FORMATION defaulted (SC = 0.75)
 ModSAF SPACING defaulted (SC = 0.75)
 ModSAF X_DI_OFFSET defaulted (SC = 0.75)
 ModSAF Y_DI_OFFSET defaulted (SC = 0.75)
 ModSAF ASSAULT_REASON defaulted (SC = 0.75)
 ModSAF DI_FORMATION defaulted (SC = 0.75)

Experiment 9

The Passage of Lines behavior is used to move units through friendly positions either in a rearward or forward fashion. Forward specifies moving into enemy territory and rearward specifies moving across enemy lines back into friendly territory. The behavior is distinguished by the fact that the moving platoon forms a column formation and passes through a single lane without firing to eliminate possible fratricide. Supporting

units can fire while the unit is performing the passage of lines. The passing unit can fire at the enemy before the passing lane is reached while the supporting units cannot. Figure 28 shows the CCTT Perform Passage Of Lines behavior.

CCTT PERFORM PASSAGE OF LINES

```

EXECUTE_PASSAGE_OF_LINES
  PASSAGE_OF_LINES
    TRAVEL
      vehicle_MOVE
    ACTION_DRILL
      SEEK_COVER_AND_CONCEALMENT
        vehicle_SEARCH
      GENERATE_SITREP
      vehicle_OCCUPY_POSITION
        vehicle_MOVE
        vehicle_SEARCH
        vehicle_HIDE
          vehicle_HALT
          vehicle_MOVE
          vehicle_SEARCH
      GENERATE_REQUEST_FOR_INDIRECT_FIRE
      GENERATE_SITREP
  
```

Figure 28. CCTT Passage of Lines Behavior

For the CCTT Perform Passage of Lines behavior, the following semantic closeness values were calculated for the ModSAF behaviors:

EXECUTE ASSAULT	0.0
EXECUTE ASSEMBLE	0.21917
EXECUTE ATTACH	0.334817
EXECUTE ATTACK BY FIRE	0.0
EXECUTE BREACH	0.332842
EXECUTE CHANGE FORMATION	0.23466
EXECUTE CONCEALMENT	0.312218
EXECUTE DELAY	0.0

EXECUTE DETACH	0.334817
EXECUTE FOLLOW VEHICLE	0.0
EXECUTE HALT	0.21917
EXECUTE HASTY OCCUPY POSITION	0.0
EXECUTE OVERWATCH MOVEMENT	0.355739
EXECUTE PLOW BREACH	0.332842
EXECUTE PURSUE	0.0
EXECUTE ROAD MARCH	0.337484
EXECUTE SUPPLY	0.302522
EXECUTE TRAVEL	0.337484
EXECUTE TRAVELING OVERWATCH	0.39317
EXECUTE WITHDRAW	0.303382

The Perform Passage of Lines behavior is one of the more complicated tank platoon behaviors. Thus, a close correlation with any ModSAF behavior is not expected. This was the case with the closest correlation being the ModSAF Traveling Overwatch behavior (Figure 20) with a semantic closeness of 39%. Since these are both forms of movement in the presence of enemy forces, subject matter experts have deemed this a reasonable correlation under the given circumstances. For the same reason, Overwatch Movement is the second choice at 36%.

The CCTT parameters were correlated with the ModSAF parameters in the following fashion with their corresponding closeness values:

CCTT UNIT_ID to ModSAF UNIT_ID (SC = 1.0)
 CCTT PLATFORM to ModSAF PLATFORM (SC = 1.0)
 CCTT ROUTE to ModSAF ROUTE (SC = 1.0)
 CCTT ENEMY_DIRECTION ignored (SC = 0.0)
 CCTT REARWARD_FLAG ignored (SC = 0.0)
 ModSAF SPEED defaulted (SC = 0.75)
 ModSAF MARCH_SPEED defaulted (SC = 0.75)
 ModSAF FORMATION defaulted (SC = 0.75)
 ModSAF CONFORM_TO_TERRAIN_FLAG defaulted (SC = 0.75)

ModSAF SPACING defaulted (SC = 0.75)
ModSAF FOLLOW_DISTANCE (SC = 0.75)

Experiment 10

The Platoon Defensive Mission is a more complicated form of the Occupy Battle Position behavior. It provides for more defense time to prepare defenses including camouflage, hasty mine fields, tank traps, barriers, etc. The tank positions include both hull down and turret down positions including primary and alternate positions. Figure 29 shows the CCTT Platoon Defensive Mission behavior.

CCTT PLATOON DEFENSIVE MISSION

```

EXECUTE_PLATOON_DEFENSIVE_MISSION
  PLATOON_DEFENSIVE_MISSION
    OCCUPY_BATTLE_POSITION
      SEEK_COVER_AND_CONCEALMENT
        vehicle_SEARCH
      TRAVEL
        vehicle_MOVE
      HALT
        vehicle_OCCUPY_POSITION
          vehicle_MOVE
          vehicle_SEARCH
          vehicle_HIDE
            vehicle_HALT
            vehicle_MOVE
            vehicle_SEARCH
        HASTY_OCCUPY_POSITION
          SEEK_COVER_AND_CONCEALMENT
            vehicle_SEARCH
          TRAVEL
            vehicle_MOVE
          vehicle_OCCUPY_POSITION
        ...
      SEEK_COVER_AND_CONCEALMENT
        vehicle_SEARCH
      vehicle_OCCUPY_POSITION
    ...
  vehicle_ENGAGE_ENEMY
    vehicle_MOVE
    vehicle_TARGETER
      vehicle_SEARCH
      vehicle_SHOOT
  CONSOLIDATE_AND_REORGANIZE
    SEEK_COVER_AND_CONCEALMENT
      vehicle_SEARCH
    vehicle_OCCUPY_POSITION
  ...
  GENERATE_SITREP

```

Figure 29. CCTT Platoon Defensive Mission Behavior

For the CCTT Platoon Defensive Mission behavior, the follow semantic closeness values were calculated for the ModSAF behaviors:

EXECUTE ASSAULT	0.540253
EXECUTE ASSEMBLE	0.281675
EXECUTE ATTACH	0.482584
EXECUTE ATTACK BY FIRE	0.497283
EXECUTE BREACH	0.490112
EXECUTE CHANGE FORMATION	0.265389
EXECUTE CONCEALMENT	0.464716
EXECUTE DELAY	0.493852
EXECUTE DETACH	0.482584
EXECUTE FOLLOW VEHICLE	0.0
EXECUTE HALT	0.281675
EXECUTE HASTY OCCUPY POSITION	0.454541
EXECUTE OVERWATCH MOVEMENT	0.488002
EXECUTE PLOW BREACH	0.490112
EXECUTE PURSUE	0.0
EXECUTE ROAD MARCH	0.466327
EXECUTE SUPPLY	0.449975
EXECUTE TRAVEL	0.466327
EXECUTE TRAVELING OVERWATCH	0.502675
EXECUTE WITHDRAW	0.482678

As mentioned previously, any behaviors that involve attacking the enemy will be similar to the ModSAF Assault behavior. Since there is no ModSAF Platoon Defensive Mission, the Assault (Figure 14) is the best correlation with a semantic closeness of 54%. The major difference between these two is that one has a defensive focus (wait for the enemy to attack you and then attack from occupied positions) and the other an offensive focus (attack the enemy's occupied position).

The CCTT parameters were correlated with the ModSAF parameters in the following fashion with their corresponding closeness values:

CCTT UNIT_ID to ModSAF UNIT_ID (SC = 1.0)
CCTT PLATFORM to ModSAF PLATFORM (SC = 1.0)
CCTT ROUTE to ModSAF ROUTE (SC = 1.0)
CCTT ENEMY_POSITION ignored (SC = 0.75)
CCTT ENEMY_DIRECTION ignored (SC = 0.75)
CCTT SECTOR to AREA to OBJECTIVE (SC = 0.81)
CCTT ENGAGEMENT_AREA to AREA to OBJECTIVE (SC = 0.81)
CCTT BATTLE_POSITION to POSITION to AREA to OBJECTIVE
(SC = 0.729)
CCTT TRIGGER_LINE to LINE to ModSAF ROUTE (SC = 0.81)
CCTT HIDE_POSITION ignored (SC = 0.75)
ModSAF LEFT_TACTICAL_BOUNDARY defaulted (SC = 0.75)
ModSAF RIGHT_TACTICAL_BOUNDARY defaulted (SC = 0.75)
ModSAF SPEED defaulted (SC = 0.75)
ModSAF DISMOUNTED_SPEED defaulted (SC = 0.75)
ModSAF STOPPING_ASSAULT_CRITERIA defaulted (SC = 0.75)
ModSAF SECURE_OBJECTIVE_FLAG defaulted (SC = 0.75)
ModSAF FORMATION defaulted (SC = 0.75)
ModSAF SPACING defaulted (SC = 0.75)
ModSAF X_DI_OFFSET defaulted (SC = 0.75)
ModSAF Y_DI_OFFSET defaulted (SC = 0.75)
ModSAF ASSAULT_REASON defaulted (SC = 0.75)
ModSAF DI_FORMATION defaulted (SC = 0.75)

Of interest in this parameter correlation is the correlation of defensive positions to the ModSAF OBJECTIVE. ENEMY_POSITION is not correlated since its closeness is not above the default value of 75%. This will be acceptable if it is assumed that the enemy is in close quarters with the defending platoon and thus the defensive position can be considered the same as the enemy position/objective to be taken.

Experiment 11

The Platoon Fire and Movement is usually an emergency behavior in which an enemy is attacking the platoon and there is no cover and concealment available. The platoon tries to leave the battle area as quickly as possible by shooting at the enemy and performing evasive maneuvers. Figure 30 shows the CCTT Platoon Fire and Movement behavior.

CCTT PLATOON FIRE AND MOVEMENT

```

EXECUTE_PLATOON_FIRE_AND_MOVEMENT
  ACTIONS_ON_CONTACT
    PLATOON_FIRE_AND_MOVEMENT
      SEEK_COVER_AND_CONCEALMENT
        vehicle_SEARCH
      GENERATE_REQUEST_FOR_INDIRECT_FIRE
      TRAVEL
        vehicle_MOVE
      vehicle_OCCUPY_POSITION
        vehicle_MOVE
        vehicle_SEARCH
        vehicle_HIDE
          vehicle_HALT
          vehicle_MOVE
          vehicle_SEARCH
      ACTION_DRILL
        SEEK_COVER_AND_CONCEALMENT
          vehicle_SEARCH
        GENERATE_SITREP
        vehicle_OCCUPY_POSITION
      ...
      BOUNDING_OVERWATCH
        SECTION_TRAVEL
          vehicle_MOVE
        HALT
        vehicle_MOVE
        SEEK_COVER_AND_CONCEALMENT
          vehicle_SEARCH
        vehicle_OCCUPY_POSITION
      ...
      CONTACT_DRILL
        vehicle_MOVE
        vehicle_SEARCH

```

Figure 30. CCTT Platoon Fire and Movement Behavior

For the CCTT Platoon Fire and Movement behavior, the follow semantic closeness values were calculated for the ModSAF behaviors:

EXECUTE ASSAULT	0.528677
EXECUTE ASSEMBLE	0.287744
EXECUTE ATTACH	0.428525
EXECUTE ATTACK BY FIRE	0.435571
EXECUTE BREACH	0.429607
EXECUTE CHANGE FORMATION	0.267765
EXECUTE CONCEALMENT	0.399758
EXECUTE DELAY	0.424521
EXECUTE DETACH	0.428525
EXECUTE FOLLOW VEHICLE	0.0
EXECUTE HALT	0.287744
EXECUTE HASTY OCCUPY POSITION	0.393894
EXECUTE OVERWATCH MOVEMENT	0.436492
EXECUTE PLOW BREACH	0.429607
EXECUTE PURSUE	0.0
EXECUTE ROAD MARCH	0.424327
EXECUTE SUPPLY	0.406162
EXECUTE TRAVEL	0.424327
EXECUTE TRAVELING OVERWATCH	0.486616
EXECUTE WITHDRAW	0.404693

Like the previous cases with no corresponding ModSAF behavior, the best correlation for CCTT Platoon Fire and Movement is the ModSAF Assault (Figure 14) with a semantic closeness of 53%.

The CCTT parameters were correlated with the ModSAF parameters in the following fashion with their corresponding closeness values:

CCTT UNIT_ID to ModSAF UNIT_ID (SC = 1.0)
 CCTT PLATFORM to ModSAF PLATFORM (SC = 1.0)
 CCTT ENEMY_POSITION to POSITION to AREA to ModSAF OBJECTIVE
 (SC = 0.729)
 CCTT ENEMY_CAPABILITY to NO MATCH (SC = 0.0)

CCTT BATTLE_DRILL ignored (SC = 0.75)
CCTT ROUTE to ModSAF ROUTE (SC = 1.0)
CCTT OVERWATCH_POSITION ignored (SC = 0.75)
CCTT ALPHA_SECTION ignored (SC = 0.75)
CCTT BRAVO_SECTION ignored (SC = 0.75)
ModSAF LEFT_TACTICAL_BOUNDARY defaulted (SC = 0.75)
ModSAF RIGHT_TACTICAL_BOUNDARY defaulted (SC = 0.75)
ModSAF SPEED defaulted (SC = 0.75)
ModSAF DISMOUNTED_SPEED defaulted (SC = 0.75)
ModSAF STOPPING_ASSAULT_CRITERIA defaulted (SC = 0.75)
ModSAF SECURE_OBJECTIVE_FLAG defaulted (SC = 0.75)
ModSAF FORMATION defaulted (SC = 0.75)
ModSAF SPACING defaulted (SC = 0.75)
ModSAF X_DI_OFFSET defaulted (SC = 0.75)
ModSAF Y_DI_OFFSET defaulted (SC = 0.75)
ModSAF ASSAULT_REASON defaulted (SC = 0.75)
ModSAF DI_FORMATION defaulted (SC = 0.75)

As with most of the correlations with no ModSAF equivalent, many of the parameters of both behaviors are ignored and defaulted.

Experiment 12

Hasty Occupy Position is a simpler form of occupy battle position in which there is no time to prepare any defenses such as barriers, hull down positions, etc. usually found in occupy battle position or platoon defensive mission. The platoon must simply find and occupy the best cover and concealed positions possible as soon as possible. Figure 31 shows the CCTT Hasty Occupy Position behavior.

CCTT HASTY OCCUPY POSITION

```

EXECUTE_HASTY_OCCUPY_POSITION
  ACTIONS_ON_CONTACT
  HASTY_OCCUPY_POSITION
    SEEK_COVER_AND_CONCEALMENT
      vehicle_SEARCH
    TRAVEL
      vehicle_MOVE
    vehicle_OCCUPY_POSITION
      vehicle_MOVE
      vehicle_SEARCH
      vehicle_HIDE
        vehicle_HALT
        vehicle_MOVE
        vehicle_SEARCH

```

Figure 31. CCTT Hasty Occupy Position Behavior

For the CCTT Hasty Occupy Position behavior, the follow semantic closeness values were calculated for the ModSAF behaviors:

EXECUTE ASSAULT	0.585431
EXECUTE ASSEMBLE	0.26135
EXECUTE ATTACH	0.538474
EXECUTE ATTACK BY FIRE	0.55736
EXECUTE BREACH	0.535281
EXECUTE CHANGE FORMATION	0.294035
EXECUTE CONCEALMENT	0.495715
EXECUTE DELAY	0.492834
EXECUTE DETACH	0.538474
EXECUTE FOLLOW VEHICLE	0.0
EXECUTE HALT	0.26135
EXECUTE HASTY OCCUPY POSITION	0.594519
EXECUTE OVERWATCH MOVEMENT	0.54966
EXECUTE PLOW BREACH	0.535281
EXECUTE PURSUE	0.0
EXECUTE ROAD MARCH	0.532719
EXECUTE SUPPLY	0.512
EXECUTE TRAVEL	0.532719

EXECUTE TRAVELING OVERWATCH	0.554781
EXECUTE WITHDRAW	0.496009

For this final experiment there is a corresponding ModSAF behavior and it does have the highest correlation. The ModSAF Hasty Occupy Position (Figure 32) correlates with the CCTT behavior of the same name with a semantic closeness of 59%. The common OCCUPY_POSITION behavior and the small difference in the number of sub-behaviors accounts for the higher correlation differentiating the ModSAF Hasty Occupy Position from the other behaviors. The parameters also correlate very well, with several parameters in common between the two behaviors.

MODSAF HASTY OCCUPY POSITION:

```

EXECUTE_HASTY_OCCUPY_POSITION
  OCCUPY_POSITION
    vehicle_ALTERNATE
    vehicle_MOVE
    vehicle_TERRAIN
    vehicle_SEARCH
  REACT_AIR
  REACT_IF
  ACTIONS_ON_CONTACT

```

Figure 32. ModSAF Hasty Occupy Position Behavior

The CCTT parameters were correlated with the ModSAF parameters in the following fashion with their corresponding closeness values:

```

CCTT UNIT_ID to ModSAF UNIT_ID (SC = 1.0)
CCTT PLATFORM to ModSAF PLATFORM (SC = 1.0)
CCTT BATTLE_POSITION to ModSAF BATTLE_POSITION (SC = 1.0)
CCTT ROUTE_TO_REAR_OF_BP to NO MATCH (SC = 0.0)

```

CCTT ENGAGEMENT_AREA to ModSAF ENGAGEMENT_AREA (SC = 1.0)
CCTT ENEMY_LOCATION to NO MATCH (SC = 0.0)
ModSAF LEFT_TRP defaulted (SC = 0.75)
ModSAF RIGHT_TRP defaulted (SC = 0.75)
ModSAF OVERWATCH_POSITION to POSITION to BATTLE_POSITION
(SC = 0.81)
ModSAF SPEED (SC = 0.75)

Experimental Conclusions

Based upon the results of the experiments, it has been shown that the use of heuristic metrics in conjunction with a corresponding behavior and parameter ontology is sufficient for correlating CCTT and ModSAF behaviors. Table 3 summarizes the results of the experiments. Out of seven expected correlations, six were correlated correctly with the one exception due to a deficiency in ModSAF, as mentioned previously. The remaining five unknown correlations were deemed acceptable by subject matter experts under the given constraints. Most of the correlations resulted in closeness values around 50% thus demonstrating the dramatic differences that can be present in externally similar systems.

Table 3.

SUMMARY OF EXPERIMENTAL RESULTS

#	CCTT SOURCE	MODSAF RESULT	MODSAF EXPECTED	SEMANTIC CLOSENESS	ACCEPT- ABLE
1	ASSAULT ENEMY POSITION	ASSAULT	ASSAULT	0.522923	YES
2	ATTACK BY FIRE	ATTACK BY FIRE	ATTACK BY FIRE	0.607225	YES
3	BOUNDING OVERWATCH	OVERWATCH MOVEMENT	OVERWATCH MOVEMENT	0.554897	YES
4	TRAVELING OVERWATCH	TRAVELING OVERWATCH	TRAVELING OVERWATCH	0.744768	YES
5	TACTICAL ROAD MRCH	BREACH	TACTICAL ROAD MRCH	0.51583	NO
6	TRAVEL	TRAVEL	TRAVEL	0.899357	YES
7	CONSOLIDAT REORGANIZE	DELAY	<NONE>	0.489362	YES
8	OCCUPY BP	ASSAULT	<NONE>	0.589559	YES
9	PASSAGE OF LINES	TRAVELING OVERWATCH	<NONE>	0.39317	YES
10	PLATOON DEFENSIVE MISSION	ASSAULT	<NONE>	0.540253	YES
11	PLATOON FIRE AND MOVEMENT	ASSAULT	<NONE>	0.528677	YES
12	HASTY OCCUPY POSITION	HASTY OCCUPY POSITION	<NONE>	0.594519	YES

CHAPTER 7

SUMMARY AND CONCLUSION

This research has shown that CGF behaviors can be correlated with behaviors from different simulations so they can interoperate with one another to support simulation training. Specific source behaviors are translated to a form in terms of general behaviors which are then correlated to any desired specific destination simulation behavior without prior knowledge of the pairing. As the experiments show, the correlation may not be 100% since the simulations may have different semantics. The experiments do show that the use of heuristic metrics in conjunction with a corresponding behavior and parameter ontology is sufficient for correlating heterogeneous simulation behavior.

This research has shown that using a database of CCTT behaviors and ModSAF behaviors written in a general form, a common ontology of behavior parameters, and a set of heuristic metrics, that CCTT and ModSAF tank platoons can interoperate (to a degree) under one task organization. These metrics successfully correlated known pairings provided by experts and provided reasonable correlations for behaviors that have no corresponding destination behavior. Of the seven known pairings experiments, six showed the expected results. Even though the correct ModSAF behaviors were selected, however, many of the closeness values were quite low. This is further proof of how simulations that appear similar externally can actually be very different in their internal semantics. As mentioned previously, the one failed experiment was not due to an error in

the correlation algorithm but due to the drastic difference in robustness of the supposedly the same behavior. The five unknown pairings produced acceptable results (as determined by experts) when considering that there was no corresponding ModSAF behaviors for these CCTT behaviors. The ModSAF and CCTT units are still interoperating but not to the degree desired. Often 100% interoperability of like simulations (same class such as virtual or constructive) requires complete reengineering of one of the simulations to the extent that it is no longer beneficial to use two different simulations at all.

Even though the experiments were a success, the correlation does have limitations that may have an effect for different behaviors or different simulations. If this is the case, the heuristic metrics serve as a foundation for more complex methods of behavior correlation. Behaviors may have the same sub-behaviors but these sub-behaviors may have a different semantic interpretation. For example, the OCCUPY POSITION behavior is treated as the same in different behaviors but the positions being occupied may be different. This difference is reflected in the parameter correlation but its effect on the overall correlation is small and may not be enough to cause the correlation to choose a different behavior. The parameter ontology can be expanded to include more semantic information, i.e. more classifications. Context information can be added by creating more specific forms of occupy position that specify the type of position in the behavior name (occupy_consolidation_position or occupy_resupply_position, for example). Also, parameter values can be analyzed as a means of obtaining more information about the behavior. Often, the default parameter values or the values of flags can change the

execution of a behavior. This is especially true in the case of ModSAF because it uses one set of behavior code to execute different behaviors. The ModSAF Travel, Tactical Road March, Pursue, and Follow Vehicle behaviors are an example of this. In some cases, their parameters can distinguish them but in others they cannot. For the CCTT and ModSAF behaviors in these experiments these limitations were not a problem.

Another situation that did not arise in these experiments but may in other simulations is that of destination behavior ambiguity. The definition of semantic closeness used by this research treats behaviors as the same if they share the same common core of behaviors as the source behavior to be correlated. In other words, superfluous destination behaviors (and the number of them) have no effect on the correlation. One destination behavior with one extra behavior is treated the same as another that may have three extra behaviors. Intuitively, it may be better to choose the behavior with the least amount of extra behaviors over another possibility with more.

As mentioned previously, the parameter correlation has difficulty when there is more than one equally related parameter and only one destination parameter to correlate to. In the case of correlating the CCTT Assault behavior we know that the ASSAULT_ROUTE was the best correlation but it is unclear as to how the algorithm can determine this automatically. Correlating in the other direction, a single source behavior can be matched against more than one destination behavior. In some cases this may be satisfactory but in other cases it may cause unexpected results and thus the destination parameters should have been allowed to default. The question of what is a good default

adjustment also becomes a factor. When should a parameter be allowed to default instead of using some form of inheritance or decomposition conversion. This only occurred occasionally in these experiments but some apriori knowledge code may need to be used to modify the parameter correlation for known problems before assigning the behavior. As an example, code can be used that will check to see if all the routes are the same and if they are, default all the routes except the assault route.

Of lesser importance is differentiating between plural and single parameters. In this implementation, plurals are treated as a singular parameter. For example, an array of halt points are treated as a single halt point. In many cases this is satisfactory but some behaviors may have nuances in semantics depending upon whether an array of data or a single datum is used. For the CCTT and ModSAF experiments this was never an issue.

The previously described limitations beg the question of using a more detailed representation of behavior. Behaviors can be broken down into their states, actions occurring in these states, and state transitions. Unfortunately, this not only assumes a FSM approach to the behavior generation of the simulations in question, but this level of detail causes more problems than it solves. There is an infinite number of combinations of states and transitions that could be used to represent the same behavior. Determining if two behaviors are similar with this amount of detail is not practical.

Even with the limitations mentioned above, this research has shown that a less sophisticated form of correlation with a simple behavior representation can indeed correlate behavior correctly and satisfactorily in most cases. It also demonstrates the

promise of using heuristic metrics and knowledge frameworks to solve semantic interoperability problems. As the state of the art in CGF increases, these semantic interoperability issues will become the dominant factor in the pursuit of large-scale and joint exercises. This research is but the first step towards the heterogeneous simulations of the future.

CHAPTER 8

FUTURE WORK

There are several areas related to this research where further work can be focused. There were several issues addressed in the correlation algorithm. Specifically, how to handle source parameters that correlate to more than one destination parameter equally, and destination parameters that correlate to more than one source behavior. Both can cause unexpected behavior when the behavior is executed with these parameter conversions. In addition, the algorithm can be modified to examine parameter values as a means of reducing the ambiguity among destination behaviors where behaviors are the same but use a parameter value to adjust the behavior slightly. Since the general representation of the behaviors makes no assumptions about run-time selection of sub-behaviors, behaviors may have the same sub-behaviors and from a correlation standpoint appear the same, but during run-time execute different combinations of the sub-behaviors.

Using the same correlation algorithm and heuristic metrics, a general approach to the validation and verification analysis of simulation behavior can be achieved. Assuming that the source behavior is already doctrinal and validated, other SAF systems can be graded on their compliance (i.e. closeness) with the source behavior. This can be further expanded to comparing the source and destination behaviors by their simulation results, i.e. verification and validation by observation. Along a similar vein, SAF behaviors could be learnt by observing another validated system. This would overcome the decrease in

interoperability observed with this approach since the simulation semantics do not correlate well.

A logical extension of this work is SAF behavior generation. By adapting simulations to be more data-driven, the behavior of a simulation can be imposed upon another simulation. The general representation of behavior used in this research can be easily enhanced to support a general representation language for FSMs. Coupled with an interpreter and a set of simulation specific primitives sharing common names, behaviors can be correlated exactly and executed at run time. This does not just meet an interoperability goal. There is a desire for easily modifiable SAF systems that can accommodate foreign doctrine or changing threats. A general behavior language will enable SAF developers to perform this task much easier than currently allowed. This language can be augmented with a set of domain-specific parameter types (such as the ones used in the parameter ontology for this work), domain-specific predicates and domain-specific actions. These domain-specific items will be converted at run-time to the appropriate simulation-specific representations and functions. This run-time interpretation can also lead to a pre-compilation of the FSMs to increase run-time performance.

In this research it was assumed that one simulation was always controlling the mission behavior of another. A more complicated approach is to remove this assumption and deal with collaborative arbitration. In this case, simulations are no longer interoperating under the same task organization, but agents in the simulation are working together to produce the appropriate behavior. Criteria must be developed whereby one

simulation behavior is chosen over another or they are combined into a new behavior and executed.

APPENDIX

Correlation Algorithm Pseudocode

```
// The Where_Is metric will compare the difference in level of
// decomposition between a source behavior and its location in a
// destination behavior

Where_Is(max,source_level,current_level,
         source_behavior,destination_behavior,isReactive)
begin

// if the difference between the current_level and the source_level is
// greater than what we found so then we need not go any further

if (source_behavior <> destination_behavior) then

    // found the behavior, calculate decrease in closeness based upon how many levels of
    // decomposition we went through compared to the decomposition level of the source

    adjustment = 1.0
    level = absolute value of (current_level - source_level)
    for (i = 0 to level -1)
        adjustment = adjustment - WHERE_IS_ADJUSTMENT*adjustment
    if (adjustment > max)
        max = adjustment
    end if

for each destination sub-behavior begin

    // don't allow reactive source behaviors to be searched for in
    // non-reactive destination behaviors and vice-versa

    if ((NOT behavior_isReactive AND NOT destination reactive)
        OR (behavior_isReactive AND destination reactive))
        Where_Is(max,source_level,current_level+1,
                source_behavior,destination_behavior,isReactive)
    end for
end Where_Is
```

```
// The ParentOf metric determines the inferential distance between a
// source behavior and a more specific destination behavior
```

```
ParentOf(destination_behavior,source_level,children,count,isReactive)
begin
  metric_value = 0.0
  max = 0.0
  if (children exist) begin

    // see if we can find it anywhere

    Where_Is(metric_value,source_level,0,children,destination_behavior,isReactive)

    // location of more specific behavior is already adjusted depending
    // upon its location but we must also add a penalty for be more
    // specific than we want, taking into account the level of inheritance

    for j = 0 to count
      metric_value = metric_value - metric_value*PARENTOF_ADJUSTMENT
      count = count + 1
      max = metric_value

      // Only thing we are not taking into account is that more specific
      // behavior may have more than one parent that the behavior we
      // are looking for do not have. We can assume that like in the
      // HAS-A case the behavior does more than we want and that is OK

      metric_value = 0.0
      if (children has children) {
        for each child of the child
          begin
            metric_value = ParentOf(destination_behavior,source_level,children,
                                   count,isReactive)
            if (metric_value > max)
              max = metric_value // keep the best
          end for
        end if
      end if
    return max
  end ParentOf
```



```

// The Isa metric determines the inferential distance between a source
// behavior and a more general destination behavior

Isa(destination_behavior,source_level,parent,count,isReactive)
begin
  metric_value = 0.0
  max = 0.0
  if (parent exists) begin

    // see if we can find it anywhere

    Where_Is(metric_value,source_level,0,parent,Destination,isReactive)

    // location of more general behavior is already adjusted depending
    // upon its location but we must also add a penalty for be more
    // general than we want, taking into account the level of inheritance

    for j = 0 to count
      metric_value = metric_value - metric_value*ISA_ADJUSTMENT
      count = count + 1
      max = metric_value
      metric_value = 0.0

    // try the next parent

    if (parent has parents) {
      parent_contribution_percentage = 1.0 / number of parents
      for each parent of the parent
        metric_value = metric_value + parent_contribution_percentage
          * Isa(Destination,source_level,parent,count,isReactive)
      if (metric_value > max)
        max = metric_value // keep the best
      end if
    end if
  return max
end Isa

```

```

// The SiblingOf metric determines the closeness of a source behavior with
// a destination behavior that shares at least one common parent. The
// more parents in common, the better the closeness

SiblingOf(source,destination,int source_level,isReactive)
begin
  max = 0.0
  for (each parent) begin

    // check the children of each parent of the source behavior

    for (each child of the parent) begin
      metric_value = 0.0
      if (child = source) begin

        // see if the destination behavior has it

        Where_Is(metric_value,source_level,0,child,destination,isReactive)

        // adjust the metric

        metric_value = metric_value * number of parents in common / number of parents
        metric_value = metric_value - SIBLING_ADJUSTMENT*metric_value
        if (metric_value > max)
          max = metric_value // keep the best
        end if
      end for
    end for
  end for
  return max
end SiblingOf

// CreateClosenessPly calculates all the metrics for the next source
// behavior. It can be thought of as the next ply in a tree with each
// ply representing each sub-behavior of the current behavior

CreateClosenessPly(behaviors,source,destination,source_level,contribution_percentage,
  reactive_contribution_percentage,isReactive)
begin
  max = 0.0
  metric_value = 0.0

  // WHERE-IS: add to the array the various closeness values calculated based
  // upon the various levels the behavior is found in on the destination

```

```

Where_Is(metric_value,source_level,0,source,Destination,
          isReactive OR source is reactive)
if (metric_value > max)
  max = metric_value

// HAS-A: We are always going to do a HAS-A so we only want to
// adjust the CreateClosenessTree call if the behavior was not
// found. If the behavior is a primitive, then it has no HAS-A metric

metric_value = 0.0
if ((max = 0) AND (number of sub-behaviors <> 0))
  metric_value = 1.0 - HAS_A_ADJUSTMENT
if (metric_value > max)
  max = metric_value

// IS-A: Do a specific to general comparison

if (max < (1.0 - ISA_ADJUSTMENT)) begin // best ISA can do
  metric_value = 0.0
  if (source's parents exist) begin

    // adjust for the number of parents found on the destination

    parent_contribution_percentage = 1.0 / number of parents
    for (each parent)
      metric_value = metric_value + isa_contribution_percentage
      * Isa(destination,source_level,parent,0,isReactive OR source is reactive)
    end if
  if (metric_value > max)
    max = metric_value
  end if

// IS-A/PARENT-OF: Do a general to specific comparison

if (max < (1.0 - PARENTOF_ADJUSTMENT)) begin
  metric_value = 0.0
  if (children exist) begin

    // adjust for the number of children found

    children_contribution_percentage = 1.0 / number of children
    for each child

```

```

        metric_value = metric_value + children_contribution_percentage
        * ParentOf(destination,source_level,child,0,isReactive OR source is reactive)
    end if
    if (metric_value > max)
        max = metric_value
    end if

// SIBLING-OF: Get closeness matches for any siblings of this
// behavior. Go through each parent of the behavior and try all
// more specific behaviors of each of these parents

if (max < (1.0 - SIBLING_ADJUSTMENT)) begin
    metric_value = SiblingOf(source,destination,source_level,isReactive OR
                            source is reactive)

    if (metric_value > max)
        max = metric_value
    end if

// go through each closeness combination and try the next behavior on the
// same level

if (source is reactive)
    metric_value = reactive_contribution_percentage *
        CreateClosenessTree(source,destination,source_level,max,
                            isReactive OR source is reactive)
else
    metric_value = contribution_percentage * CreateClosenessTree(
        source,destination,source_level,max,isReactive OR source is reactive)

// get next behavior at this level if we have one

if (more behaviors on this level) {

    // go onto the next level;

    metric_value = metric_value + CreateClosenessPly(behaviors,behavior,destination,
        source_level,contribution_percentage,
        reactive_contribution_percentage,isReactive)
}
end if
return metric_value
end CreateClosenessPly

```

```
// The CreateClosenessTree function will determine the closeness value of a
// behavior based upon the closeness values of its sub-behaviors. In a sense
// this is exactly the HAS-A metric only that this is done whether the
// behavior is found or not because no assumption is made about behaviors
// having the same name being the same.
```

```
CreateClosenessTree(source,destination,source_level,adjustment,isReactive)
begin
```

```
if (no sub-behaviors) return adjustment // if primitive return adjustment
```

```
// determine the influence of each sub-behavior on the total semantic closeness
// If behavior is reactive we don't want to count this as much as all the others
```

```
float contribution_percentage=0.0
float reactive_contribution_percentage=0.0
if (number of reactive sub-behaviors <> 0) {
```

```
    // allow all reactive behaviors to contributed together as the
    // REACTIVE_PERCENTAGE or the contribution of one behavior whichever
    // is lower
```

```
    contribution_percentage = 1.0 /
      number of sub-behaviors - number of reactive sub-behaviors + 1)
    if (contribution_percentage < REACTIVE_PERCENTAGE)
      reactive_contribution_percentage = contribution_percentage /
        number of reactive sub-behaviors
    else begin
      reactive_contribution_percentage = REACTIVE_PERCENTAGE /
        number of reactive sub-behaviors
      contribution_percentage = (1.0 - REACTIVE_PERCENTAGE) /
        number of sub-behaviors - number of reactive sub-behaviors)
    end else
  end if
else contribution_percentage = 1.0 / number of sub-behaviors
```

```
// go into first sub-behavior
```

```
return adjustment * CreateClosenessPly(behaviors,next_behaviors,destination,
  source_level+1,contribution_percentage,reactive_contribution_percentage,
  isReactive)
end CreateClosenessTree
```

```

// Determine_Closeness determines the semantic closeness of a behavior
// with a destination taking into account the behavior closeness and
// the parameter closeness summed together according to their respective
// percentage contributions

Determine_Closeness(source,Behavior *destination)
begin

    // if parameters do not match at all, then at least one required parameter
    // was not able to be correlated so the behavior cannot be used.

    if (parm_correlate(source_args,destination_args, parameter_correlation) < 0)
        return 0.0

    behavior_correlation = CreateClosenessPly(bc,source,destination,0,1.0,0.0)

    return ((1.0 - PARAMETER_PERCENTAGE) * behavior_correlation +
            PARAMETER_PERCENTAGE * parameter_correlation)
end DetermineCloseness

// Correlate determines the best semantic closeness of a behavior with
// destination behaviors and displays the corresponding parameter
// correlation to the screen

correlate(behavior_name,destination_behaviors)
begin
    max = 0.0

    // check each assignable destination behavior

    for (each destination behavior) begin
        closeness = Determine_Closeness(source_behavior,destination_behavior)
        print out result

        if (closeness > max)
            max = closeness
    end for
    print out best result
end correlate

```

Parameter Correlation Algorithm Pseudocode

```
// IsaParent checks to see if a parent of a source parameter is a
// destination parameter
```

```
IsaParent(source, destination, count)
begin
  if (source = destination)
    return count
  for (each parent) begin
    found = IsaParent(parent, destination, count + 1)
    if (found)
      return found
  end for
  return 0
end IsaParent
```

```
// IsaChild checks to see if a child of a source parameter is a
// destination parameter
```

```
IsaChild(source, destination, count)
begin
  if (source = destination)
    return count
  for (each child) begin
    found = IsaChild(child, destination, count+1)
    if (found)
      return found
  end for
  return 0
end IsaChild
```

```
// CheckParent will check to see if a destination parameter or one of its
// parents is a parent of the source behavior and will check to see if
// a destination parameter or one of its children is a child of the source
// parameter.
```

```
CheckParent(source, destination, count)
begin
  found = IsaParent(source, destination, count)
  if (not found)
    found = IsaChild(source, destination, count)
  else return found
```

```

// follow every parent path if multiple parents exist

for (each parent) begin
  found = IsaParent(source,parent,count+1)
  if (not found)
    found = IsaChild(source,parent,count+1)
  else return found
end for
return found
end CheckParent

// parm_Isa determines the intersection path of parents or children of a
// source parameter with any destination parameter. The best one is used
// as the result of the metric. A destination parameter or one of its
// parents may be the parent of a source parameter or vice-versa.

parm_Isa(source,destination_parameters)
begin
  max=0.0

  // check each destination parameter

  for (each destination parameter) begin
    count = CheckParent(source,destination parameter,0)
    if (count is valid) begin

      // we found a conversion path

      metric_value = 1.0
      for j = 0 to count
        metric_value = metric_value - metric_value*PARM_ISA_ADJUSTMENT
        if (metric_value > max)
          max = metric_value
        end if
      end for
    end for
  return max
end parm_Isa

```



```

// parm_CreateClosenessPly, like its behavior counterpart, will evaluate
// each of parameter metrics and choose the best match

parm_CreateClosenessPly(parameters,source,destination_parameters,source_level,
    contribution_percentage)
begin
    max = 0.0
    metric_value = 0.0

    // if we can find the parameter, great

    if (IsAParameter(source,destination_parameters) <> 0) begin

        // If not found perform the metrics
        // Note: Default parameters can be psuedo ignored if not found

        // HAS-A:
        // If the parameter is a primitive, then it has no HAS-A metric

        if (sub-parameters of source exist) begin
            metric_value = 1.0 - PARM_HAS_A_ADJUSTMENT
            max = parm_CreateClosenessTree
                (source,destination_parameters,source_level,metric_value)
        end if

        // ISA: See if the parameter has any common parent with any of the
        // destination parameters.

        if (max < (1.0 - PARM_ISA_ADJUSTMENT)) begin
            metric_value = parm_Isa(source,destination_parameters)
            if (metric_value > max)
                max = metric_value
        end if

        if (max < (1.0 - DEFAULT_ADJUSTMENT)) begin

            // Apply the default metric if the previous metrics
            // did no better

            if (parameter is default)
                max = 1.0 - DEFAULT_ADJUSTMENT
            end if
        end if
    end if
end if

```

```

else max = 1.0

// adjust the value accordingly

metric_value = contribution_percentage*max

// try the next parameter on the
// same level

if (more parameters) begin

    // go onto the next level

    val = parm_CreateClosenessPly(parameters,parameter,
        destination_parameters,source_level,contribution_percentage)
    if (val = 0.0)
        // Parameter was not able to be correlated
        return 0.0 // abort the matching
    metric_value = metric_value + val

end if
return metric_value
end parm_CreateClosenessPly

// parm_CreateClosenessTree is used to perform the HAS-A for parameter
// correlation/conversion. The closeness value for a parameter is based
// upon the closeness values of its sub-parameters and how they correlate
// to destination parameters

parm_CreateClosenessTree(Source,destination_parameters,source_level,
    adjustment)
begin

// check for primitive, if so return no adjustment to closeness

if (no sub-parameters)
    return adjustment

// determine the influence of each sub-parameter on the total
// semantic closeness

contribution_percentage = 1.0 / number of sub-parameters

```

```

// loop through each sub-Parameter

float adjustment1 = 0.0

for (each sub-parameter) begin
    val = parm_CreateClosenessPly(parameters,sub-parameter,source_level+1,
        contribution_percentage)
    if ((val == 0.0) AND not converting from source to destination
        return 0.0 // could not find subparameter, abort
    adjustment1 = adjustment1 + val
end for
return (adjustment1*adjustment)
end parm_CreateClosenessTree

// parm_correlate correlates all the source parameters with destination
// parameters. It returns the closeness value or returns 0, if a required
// destination parameter went unmatched

parm_correlate(source_parameters,destination_parameters,total)
begin

// first we need to match all the parameters from the given behavior
// to that of the destination behavior. Any required unmatched parameters
// reduce the closeness but do not eliminate the behavior since the
// the destination may not do as much and this may be the best we can do

if (source_parameters not empty) begin
    source_contribution_percentage = 1.0 / number of source parameters
    parm_CreateClosenessPly(source_parameters,
        first source parameter,destination_parameters,0,source_contribution_percentage)
end if

// Now, try to match any unmatched destination parameters with the
// source parameters. Required destination parameters must be matched
// or the behavior cannot be performed

if (destination_parameters not empty) {
    dest_contribution_percentage = 1.0 / number of destination parameters
    parm_CreateClosenessPly(destination_parameters,
        first destination parameter,source_parameters)
end if

```

```

// we need to calculate the total number of parameters counting
// matched parameters only once. If any destination parameter
// has a value of 1.0 then its closeness has already been taken
// care of by a match with some source parameter so we don't
// want to count it twice. Any other value means that the
// destination parameter had to be matched with some source
// parameter(s).

// loop through all the closeness values for the source parameters
// and sum them up

contribution_percentage = 1.0 / number of source parameter matches
metric_value = 0.0
for (each closeness values)
    metric_value = metric_value + ( match metric_value*contribution_percentage)
total = metric_value

// loop through all the destination parameters and any that are not 1.0
// (see above) or 0.0 add to the closeness sum

metric_value = 0.0
for (each destination parameter) {
    if (required destination parameter unmatched)
        // All required destination parameters must match
        return 0.0
    else if (a new match)
        metric_value = metric_value + (match metric_value*contribution_percentage)
end for
total = total + metric_value
return 1
end parm_correlate

```

REFERENCES

- Agre**, Phil, "What are Plans For?," *Robotics and Autonomous Systems*, Volume 6, 1990, pp. 17-34.
- Allen**, James, *Natural Language Understanding, Second Edition*, Redwood City, CA: Benjamin/Cummings, 1995.
- Altman**, Marty; **Kilby**, Mark; **Lisle**, Curtis, "On a Shared Environment Concept for Distributed Simulation," *Proceedings of the 11th Workshop on Standards for the Interoperability of Distributed Simulations*, Institute for Simulation and Training, Orlando, FL, Sept 26-30 1994, pp. 535-543.
- Arkin**, R. C., "Motor Schema Based Mobile Robot Navigation," *International Journal of Robotics Research*, Volume 8, Number 4, August 1989, pp. 92-112.
- Arkin**, R. C., "Cooperation without Communication: Multiagent Schema-Based Robot Navigation," *Journal of Robotic Systems*, Volume 9, Number 3, 1992, pp. 351-364.
- Aronson**, Jesse, "The SimCore Tactics Representation and Specification Language," *Proceedings of the Fourth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando, FL, May 4-6 1994, pp. 187-193.
- Bartel**, Lisa; **Evans**, Deona; **McCue**, Michael; **Whitney**, Stu, "An Object-Oriented Simulation Architecture," *Proceedings of the 11th Workshop on Standards for the Interoperability of Distributed Simulations*, Institute for Simulation and Training, Orlando, FL, Sept 26-30 1994, pp. 171-179.
- Becket**, Welton; **Badler**, Norman I., "Integrated Behavioral Agent Architecture," *Proceedings of the Third Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando, FL, March 17-19 1993, pp. 57-68.
- Bimson**, Kent; **Mall**, Howard; **McCormack**, Jenifer; **Ourston**, Dirk., "Command Entity Cognitive Behaviors for SAF and CGF," *Proceedings of the Fifth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando, FL, May 9-11 1995, pp. 203-208.

- Braudaway**, Wesley, "A Blackboard Approach to Computer Generated Forces," *Proceedings of the Third Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando, FL, March 17-19 1993, pp. 11-20.
- Brooks**, R. A., "A Robust Layered Control System for a Mobile Robot," *IEEE Journal of Robotics and Automation*, Volume RA-2, Number 1, 1986.
- Brooks**, R. A., "New Approaches to Robotics," *Science*, Volume 253, September 1991, pp. 1227-1254.
- Calder**, Robert B.; **Evans**, Alan B., "Construction of a Corps Level CGF," *Proceedings of the Fourth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando, FL, May 4-6 1994, pp. 487-495.
- Calder**, Robert B.; **Peacock**, Jeffrey C.; **Wise**, Ben P.; **Stanzione**, Thomas; **Chamberlain**, Forrest; **Panagos**, James, "Implementation of a Dynamic Aggregation/Deaggregation Process in the JPSD CLCGF," *Proceedings of the Fifth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando, FL, May 9-11 1995, pp. 83-91.
- Calder**, Robert B.; **Peacock**, Jeffrey C.; **Panagos**, James; **Johnson**, Thomas E., "Integration of Constructive, Virtual, Live, and Engineering Simulations in the JPSD CLCGF," *Proceedings of the Fifth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando, FL, May 9-11 1995, pp. 71-79.
- Calder**, Robert B.; **Smith**, Joshua E.; **Courtemanche**, Anthony J.; **Mar**, John M. F.; **Ceranowicz**, Andrew, "ModSAF Behavior Simulation and Control," *Proceedings of the Third Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando, FL, March 17-19 1993, pp. 347-356.
- Caroll**, K. P.; **McClaran**, S. R.; **Nelson**, E. L.; **Barnett**, D. M.; **Friesen**, D. K.; **Willaims**, G. N., "AUV path planning: An A* Approach," *Proceedings of the 1992 Symposium on AUV Technology*, June 1992, pp. 3-8.
- Castillo**, David, *Toward a Paradigm for Simulating Intelligent Agents*, Doctoral Dissertation, University of Central Florida, 1991.

Ceranowicz, Andrew., "ModSAF Capabilities," *Proceedings of the Fourth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando, FL, May 4-6 1994, pp. 3-8.

Ceranowicz, Andrew; **Coffin**, Dan; **Smith**, Joshua; **Gonzalez**, Roger; **Ladd**, Carol., "Operator Control of Behavior in ModSAF," *Proceedings of the Fourth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando, FL, May 4-6 1994, pp. 9-16.

Chaib-draa, B.; **Paquet**, E.; **Lamontagne**, L., "Integrating Reaction, Planning and Deliberation in Architecture for Multiagent Environment," *Proceedings of the Third Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando, FL, March 17-19 1993, pp. 45-55.

Chen, P. C., "Improving Path Planning with Learning," *Proceedings of the 9th International Conference on Machine Learning*, 1992, pp. 55-61.

Clark, Karen J.; **Brewer**, Dave., "Bridging the Gap Between Aggregate Level and Object Level Exercises," *Proceedings of the Fourth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando, FL, May 4-6 1994, pp. 437-442.

Cohen, Danny, "DIS -- Back to Basics," *Proceedings of the 11th Workshop on Standards for the Interoperability of Distributed Simulations*, Institute for Simulation and Training, Orlando, FL, Sept 26-30 1994, pp. 603-617.

Courtemanche, Anthony J.; **Ceranowicz**, Andy, "ModSAF Development Status," *Proceedings of the Fifth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando, FL, May 9-11 1995, pp. 3-12.

Cox, Allan; **Gibb**, Alastair; **Page**, Ian, "Army Training and CGFs - A UK Perspective," *Proceedings of the Fourth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando, FL, May 4-6 1994, pp. 353-363.

Cox, Allan; **Maybury**, Jon; **Weeden**, Nickie, "Aggregation Disaggregation Research - A UK Approach," *Proceedings of the 13th Workshop on Standards for the Interoperability of Distributed Simulations*, Institute for Simulation and Training, Orlando, FL, Sept 18-22 1995, pp. 449-463.

Dean, T.L.; **Wellman**, M.P., *Planning and Control*, Morgan and Kaufmann, 1991.

DIS Steering Committee, *The DIS Vision: A Map to the Future of Distributed Simulation Version 1*, Institute for Simulation and Training, May 1994.

Engelmore, R.; Morgan, A., editors, *Blackboard Systems*, California: Addison Wesley Publishing, 1988.

Fishwick, Paul A., *Simulation Model Design and Execution - Building Digital Worlds*, Englewood Cliffs, NJ: Prentice Hall, 1995.

Foss, Billy; Franceschini, Robert, "A Unified Aggregate Protocol," *Proceedings of the 13th Workshop on Standards for the Interoperability of Distributed Simulations*, Institute for Simulation and Training, Orlando, FL, Sept 18-22 1995, pp. 575-581.

Franceschini, Robert W.; Karr, Clark R., "Integrated Eagle/BDS-D: Results and Current Work," *Proceedings of the 11th Workshop on Standards for the Interoperability of Distributed Simulations*, Institute for Simulation and Training, Orlando, FL, Sept 26-30 1994, pp. 419-423.

Franceschini, Robert W.; Karr, Clark R., "Proposal for Revision of the DIS Aggregate Protocol," *Proceedings of the 11th Workshop on Standards for the Interoperability of Distributed Simulations*, Institute for Simulation and Training, Orlando, FL, Sept 26-30 1994, pp. 425-426.

Gagne, Denis, "Realistic Doctrinal Behaviors in Computer Generated Forces Through Plurality," *Proceedings of the Fifth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando, FL, May 9-11 1995, pp. 521-527.

Gat, Erann, "Path Planning and Execution Monitoring for a Planetary Rover," *Proceedings of the IEEE International Conference on Robotics and Automation*, IEEE Computer Society, Los Alamitos, CA, 1990, pp. 20-25.

Gat, Erann, "Integrating Planning and Reacting in a Heterogeneous Asynchronous Architecture for Controlling Real-world Autonomous Mobile Robots," *Proceedings of the Tenth National Conference on Artificial Intelligence*, AAAI, Menlo Park, CA, 1992, pp. 809-815.

Gat, Erann; Fearey, Joe; Provenzano, Joe, "Semi-Automated Forces for Corps Battle Simulation," *Proceedings of the Third Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando, FL, March 17-19 1993, pp. 69-74.

Ge, Xiaolin; **James**, John; **Nerode**, Anil, "A Multiple Agent Hybrid Control Architecture for Automated Forces: Design and Software Implementation," *Proceedings of the Fifth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando, FL, May 9-11 1995, pp. 45-50.

Generazio, Hoa; **Young**, Donovan, "The Aggregation/Disaggregation Problem in Vertical Integration of Dissimilar Combat Simulations," *Proceedings of the 12th Workshop on Standards for the Interoperability of Distributed Simulations*, Institute for Simulation and Training, Orlando, FL, March 13-17 1995, pp. 267-280.

Goel, A. K.; **Callentine**, T. J., "An Experience-based Approach to Navigational Route Planning," *Proceedings of the 1992 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 888-893.

Gomez, Fernando; **Chandrasekaran**, B., "Knowledge Organization and Distribution for Medical Diagnosis," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. SMC-11, No. 1, 1981, pp. 34-42.

Gonzalez, Avelino J.; **Ahlers**, Robert, "Context-based Representation of Intelligent Behavior in Simulated Opponents," *Proceedings of the Fifth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando, FL, May 9-11 1995, pp. 53-61.

Gonzalez, Avelino J.; **Dankel**, Douglas D, *The Engineering of Knowledge-Based Systems Theory and Practice*, Englewood Cliffs, NJ: Prentice Hall, 1993.

Gonzalez, Avelino J.; **Myler**, Harley R.; **Kladke**, Robin R., "Identification of Unconstrained Item Descriptions Using String-Match Heuristics," *International Journal of Expert Systems*, Volume 4, Number 3, pp. 337-364.

Hardy, Thomas; **Healy**, Mike., "Constructive & Virtual Interoperation: A Technical Challenge," *Proceedings of the Fourth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando, FL, May 4-6 1994, pp. 503-507.

Harmon, S. Y.; **Yang**, S. C.; **Tseng**, D. Y., "Command and Control Simulation for Computer Generated Forces," *Proceedings of the Fourth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando, FL, May 4-6 1994, pp. 263-271.

Humphrey, Darren D., "Defining Critical Parameters for Interoperability," *Proceedings of the 10th Workshop on Standards for the Interoperability of Distributed Simulations*, Institute for Simulation and Training, Orlando, FL, March 14-18 1994, pp. 245-247.

Institute for Simulation and Training, *Standard for Distributed Interactive Simulation Application Protocols Version 2.0 Fourth Draft*, March 16, 1994.

Jones, Randolph M.; **Laird**, John E.; **Tambe**, Milind; **Rosenbloom**, Paul, "Generating Behavior in Response To Interacting Goals," *Proceedings of the Fourth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando, FL, May 4-6 1994, pp. 317-323.

Jones, Randolph M.; **Tambe**, Milind; **Laird**, John E.; **Rosenbloom**, Paul S., "Intelligent Automated Agents for Flight Training Simulators," *Proceedings of the Third Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando, FL, March 17-19 1993, pp. 33-42.

Jones, Ross E., "Using CGF for Analysis and Combat Development," *Proceedings of the Third Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando, FL, March 17-19 1993, pp. 209-218.

Karr, Clark R.; **Rajput**, Sumeet; **Cisneros**, Jaime E.; **Nee**, Hai-Lin, "Automated Mission Planning in ModSAF," *Proceedings of the Fifth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando, FL, May 9-11 1995, pp. 159-167.

Karr, Clark R.; **Root**, Eric, "Integrating Aggregate and Vehicle Level Simulations," *Proceedings of the Fourth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando, FL, May 4-6 1994, pp. 425-435.

Kazarian, Jason P.; **Shultz**, Marjorie A., "The IRIS Architecture: Integrating Constructive, Live, and Virtual Simulations," *Proceedings of the 10th Workshop on Standards for the Interoperability of Distributed Simulations*, Institute for Simulation and Training, Orlando, FL, March 14-18 1994, pp. 135-145.

Keirseey, David; **Krozel**, Jimmy; **Payton**, David; **Tseng**, David, "Case-Based Computer Generated Forces," *Proceedings of the Fifth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando, FL, May 9-11 1995, pp. 307-316.

Kuokka, Daniel R., "A Framework for Integrating Autonomous Agents," *Proceedings of the Third Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando, FL, March 17-19 1993, pp. 181-190.

- Kwak**, Se-Hung, "A Comparison Study of Behavioral Representation Alternatives," *Proceedings of the Fifth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando, FL, May 9-11 1995, pp. 529-539.
- Landweer**, Phil, "Action/Cognition Behavior Model for Computer Generated Forces," *Proceedings of the Third Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando, FL, March 17-19 1993, pp. 335-344.
- Lee**, Jin Joo; **Norris**, W.D.; **Fishwick**, Paul A., "An object-oriented multimodeling design for integrating simulation and planning tasks," *Journal of Systems Engineering*, March, 1993, pp. 220-235
- Lee**, Jin Joo; **Fishwick**, Paul A., "Simulation-Based Planning for Computer Generated Forces," *Proceedings of the Fourth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando, FL, May 4-6 1994, pp. 451-460.
- Loral** Advanced Distributed Simulation, *ModSAF Software Architecture Design and Overview Document (SADOD)*, Loral, Cambridge, MA, 1995.
- Mastaglio**, Thomas; **Mengell**, Larry; **O'Connell**, James, "ACTORS: A Knowledge-Based Computer Architecture for Integrating Live, Virtual and Constructive Simulations," *Proceedings of the Third Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando, FL, March 17-19 1993, pp. 191-198.
- McEnany**, Brain R.; **Marshall**, Henry, "CCTT SAF Functional Analysis," *Proceedings of the Fourth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando, FL, May 4-6 1994, pp. 195-207.
- Moskal**, Patrick; **Johnson**, Teresa; **Schiavone**, Guy; **Cortes**, Art, "A Parametric Approach to Examining Interoperability (IOP) Issues: Initial Experiments," *Proceedings of the 11th Workshop on Standards for the Interoperability of Distributed Simulations*, Institute for Simulation and Training, Orlando, FL, Sept 26-30 1994, pp. 553-558.
- Neuberger**, Thomas; **Shea**, Dennis, "Applying Synthetic Environments to Operational Training: A Perspective from Kernel Blitz 95," *Proceedings of the 13th Workshop on Standards for the Interoperability of Distributed Simulations*, Institute for Simulation and Training, Orlando, FL, Sept 18-22 1995, pp. 305-312.

Nielsen, Paul E., "Intelligent Computer Generated Forces for Command and Control," *Proceedings of the Fifth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando, FL, May 9-11 1995, pp. 211-218.

O'Byrne, Jeff, "Computer Generated Forces Providing a Context for the Exercise Force," *Proceedings of the Third Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando, FL, March 17-19 1993, pp. 387-392.

Ourston, Dirk, "Manuever Vector Based Route Planning," *Proceedings of the 8th Florida Artificial Intelligence Research Symposium (FLAIRS)*, Melbourne, FL, 1995, pp. 355-359.

Ourston, Dirk; **Blanchard**, David; **Chandler**, Edward; **Loh**, Elsie, "From CIS to Software," *Proceedings of the Fifth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando, FL, May 9-11 1995, pp. 275-285.

Payton, D. W., "An Achitecture for Reflexive Autonomous Vehicle Control," *Proceedings of the IEEE International Conference on Robotics and Automation*, San Francisco, CA, April 1986, pp. 1838-1845.

Peck, Charles C.; **Lander**, W. Brent; **Hancock**, John P., "Applications of Distributed Object Technology to DIS," *Proceedings of the 12th Workshop on Standards for the Interoperability of Distributed Simulations*, Institute for Simulation and Training, Orlando, FL, March 13-17 1995, pp. 537-552.

Petty, Mikel, "The Turing Test as an Evaluation Criterion for Computer Generated Forces," *Proceedings of the Fourth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando, FL, May 4-6 1994, pp. 107-116.

Petty, Mikel D.; **Franceschini**, Robert W., "Disaggregation Overload and Spreading Disaggregation in Constructive+Virtual Linkages," *Proceedings of the Fifth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando, FL, May 9-11 1995, pp. 103-111.

Picket, H. Kent; **Petty**, Mikel D., "Report on The State of Computer Generated Forces 1994," *Proceedings of the Fifth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando, FL, May 9-11 1995, pp. 549-557.

Pratt, David R.; **Bhargava**, Hemant K.; **Culpepper**, Michael; **Locke**, John, "Collaborative Autonomous Agents in the NPSNET Virtual World," *Proceedings of the Fourth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando, FL, May 4-6 1994, pp. 177-185.

Pratt, David R.; **Johnson**, Matthew; **Locke**, John, "The Janus/BDS-D Linkage Project: Constructive and Virtual Model Interconnection," *Proceedings of the Fourth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando, FL, May 4-6 1994, pp. 443-447.

Ram, Ashwin; **Santamaria**, Juan Carlos, "Continuous Case-Based Reasoning," *Proceedings of the AAAI-93 Workshop on Case-Based Reasoning*, Washington, DC, 1993, pp. 86-93.

Riecken, Mark; **O'Brien**, Sheila, "DIS Synthetic Environment Interoperability Issues," *Proceedings of the 10th Workshop on Standards for the Interoperability of Distributed Simulations*, Institute for Simulation and Training, Orlando, FL, March 14-18 1994, pp. 535-544.

Rosenbloom, Paul S.; **Johnson**, W. Lewis; **Jones**, Randolph M.; **Koss**, Frank; **Laird**, John E.; **Lehman**, Jill Fain; **Rubinoff**, Robert; **Schwamb**, Karl B.; **Tambe**, Milind, "Intelligent Automated Agents for Tactical Air Simulation: A Progress Report," *Proceedings of the Fourth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando, FL, May 4-6 1994, pp. 69-75.

Rumbaugh, J., *Object-Oriented Modeling and Design*, Englewood Cliffs, NJ: Prentice Hall, 1991.

Rush, Brian; **Whitely**, David, "Architectural Framework for Distributed Interactive Simulation Systems," *Proceedings of the 10th Workshop on Standards for the Interoperability of Distributed Simulations*, Institute for Simulation and Training, Orlando, FL, March 14-18 1994, pp. 664-685.

Sandmeyer, Richard S.; **Dymond**, Marguerite M., "Assessment of Alternative Computer Generated Forces," *Proceedings of the 13th Workshop on Standards for the Interoperability of Distributed Simulations*, Institute for Simulation and Training, Orlando, FL, Sept 18-22 1995, pp. 231-236.

Schank, Roger (ed), *Conceptual Information Processing*, Amsterdam: North-Holland Publishers, 1975.

Schank, Roger; **Abelson**, R. P., *Scripts, Plans, Goals, and Understanding*, Hillsdale, NJ: Erlbaum Publishers, 1977.

Schank, Roger; **Colby**; K., *Computer Models of Thought and Language*, San Francisco, CA: Freeman Publishers, 1973.

Shlaer, S.; **Mellor**, S. J., *Object Life Cycles: Modeling the World in States*, Englewood Cliffs, NJ: Yourdon Press, 1990.

Siksik, D. N., "Intelligent Computer Generated Forces Through Expert Systems," *Proceedings of the Third Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando, FL, March 17-19 1993, pp. 3-10.

Smith, Roger D., "The Conflict Between Heterogenous Simulations and Interoperability," *Proceedings of the 17th Interservice/Industry Training Systems and Education Conference (CD-ROM)*, The American Defense and Preparedness Association, Albuquerque, NM, November 13-16, 1995.

Smith, S; **Karr**, C, "The IST Semi-Automated Forces Testbed," *Technical Report IST-Tr-92-7*, Institute for Simulation and Training, 1992.

Spuhl, Karl A.; **Findley**, David A., "Correlation Considerations in the Simulation Environment," *Proceedings of the 11th Workshop on Standards for the Interoperability of Distributed Simulations*, Institute for Simulation and Training, Orlando, FL, Sept 26-30 1994, pp. 55-58.

Stober, David R.; **Kraus**, Matthew K.; **Foss**, William F.; **Franceschini**, Robert W.; **Petty**, Mikel D., "Survey of Constructive + Virtual Linkages," *Proceedings of the Fifth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando, FL, May 9-11 1995, pp. 93-102.

STRICOM (Simulation, Training, and Instrumentation Command), "The Linkage of Constructive and Virtual Simulations," *Sample Task Package #2 for Advanced Distributed Simulation Technology II (ADST II)*, Nov 10, 1994.

Tallis, Hans, "Flexible Control of Military Subordinates in a Reactive Simulation," *Proceedings of the Third Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando, FL, March 17-19 1993, pp. 109-119.

Taylor, James G., *Lanchester Models of Warfare, Volumes I and II*, Operations Research Society of America, Baltimore, MD, 1983.

Vasudevan, C.; Ganesan, K., "Case-Based Path Planning for Autonomous Underwater Vehicles," *Proceedings of the 1994 IEEE International Symposium on Intelligent Control*, Columbus, Ohio, August 16-18, 1994, pp. 160-165.

Vasudevan, C.; Ganesan, K.; Smith, S. M., "A Fuzzy-CBR Scheme for Reactive Navigation of Autonomous Underwater Vehicles," *Proceedings of the 8th Florida Artificial Intelligence Research Symposium (FLAIRS)*, Melbourne, FL, 1995, pp. 57-61.

Warren, C. W., "A Technique for Autonomous Underwater Vehicle Route Planning," *IEEE Journal of Oceanic Engineering*, Volume 15, Number 3, 1990, pp. 199-204.

Weaver, Bruce, "Behavioral Representation Completeness in CGF Implementations," *Proceedings of the Third Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando, FL, March 17-19 1993, pp. 375-383.

Wilcox, Brian; Matthies, Larry; Nguyen, Tom; Mishkin, Andrew; Gennes, David; Cooper, Brian; Litwin, Todd; Stone, Henry, "Robot Vehicles for Planetary Exploration," *Proceedings of the IEEE Conference on Robotics and Automation*, Volume 1, IEEE, Piscataway, NJ, 1992, pp. 175-180.

Wilkins, D. E., "Domain Independent Planning: Representation and Plan Generation," *Artificial Intelligence*, Volume 22, 1984, pp. 269-301.

Wittig, Thies, "A Distributed Artificial Intelligence Approach for a Combat Training Simulator," *Proceedings of the Third Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando, FL, March 17-19 1993, pp. 121-129.