A BLACKBOARD APPROACH TO COMPUTER GENERATED FORCES

Wesley Braudaway, Ph. D.
IBM Federal Systems Company
Orlando, Florida

ABSTRACT

This paper presents the research for applying the AI blackboard paradigm to the problem of realistically emulating multiple battlefield entities within a training simulator. One challenge of these computer generated forces (CGF) is to emulate human behavior so human controlled and computer controlled entities are virtually indistinguishable. The blackboard paradigm provides a useful framework for attacking the CGF problem. The results of this research demonstrate the usefulness of a blackboard architecture for the CGF problem. The blackboard paradigm provides a means for integrating subtasks of the system that are implemented using different programming paradigms. Also, the "context" and "event" driven control strategy of the blackboard paradigm provides adaptive behavior for the computer control forces. These characteristics discriminate the blackboard architecture from other programming paradigms for use with the CGF problem. The research reported in this paper was funded by STRICOM under contract N61339–92–C–0032.

ABOUT THE AUTHOR

Dr. Wesley Braudaway received his Ph.D. from the Department of Computer Science at Rutgers University in New Jersey while on leave of absence from IBM Federal Systems Company. His thesis research involved Artificial Intelligence Knowledge Compilation and AI Design Automation. Since joining IBM in 1982, Dr. Braudaway has been involved in several research projects applying expert systems to submarine command and control problems. Over the last year he has been the principle investigator of the automated forces research task for applying AI to the computer generated forces problem. Dr. Braudaway is presently part of the Integrated Development Team to develop semi–automated forces for the CCTT program.

A BLACKBOARD APPROACH TO COMPUTER GENERATED FORCES

Wesley Braudaway, Ph. D.
IBM Federal Systems Company
Orlando, Florida

INTRODUCTION

The CGF Problem

A computer generated forces (CGF) system implements semi–automated opposing and ancillary forces that operate on a simulated battlefield and emulate realistic behavior of human lead military units. Training military for combat within the complex and chaotic environment of a full scale battlefield is a difficult and expensive task. Computer simulation coupled with *Artificial Intelligence (AI) technologies* provide a promising approach for an effective training environment of battlefield combat. This paper describes the research results of combining the AI *blackboard* paradigm[2] and *computer simulation* to automate realistic battlefield units. The research reported in this paper was funded by STRICOM under contract N61339–92–C–0032.

To model realistic battlefield situations, hundreds of entities from friendly and opposing forces must be present. *This magnitude necessitates the use of both human* controlled and computer controlled entities on the simulated battlefield. One challenge of these computer generated forces is to emulate human behavior so that the human controlled and computer controlled entities are indistinguishable. This implies that the CGF system must not only exhibit realistic behavior in a static environment, but must realistically adapt its behavior to dynamic events and emerging situations occurring on the battlefield. The combination of different events and contexts in which the events occur in a battlefield scenario, *can make a conventional control strategy for the* CGF solution very complex.

One challenge of extended research is to integrate successful approaches from previous research within a system architecture that also addresses the weaknesses of previous research results. In aggregate, a realistic CGF must be driven by battlefield events and the battlefield contexts in which these events occur. An effective and efficient CGF system must incorporate both knowledge–based (expert system like) decision making for *some tasks and algorithmic approaches for other tasks* as appropriate for those tasks' requirements. The system must also be easily modified for changes in the operational tactics that it simulates and extendible for incorporating new battlefield entities, tactics, and weapons.

As described in the next section, our preliminary research demonstrated that the AI blackboard paradigm exhibits these characteristics and the research reported in this paper validates this concept by applying the blackboard paradigm to the challenges of the CGF requirements.

This first section summarizes the prototype implementation and the research results. Section two presents an overview of the AI blackboard technology and *Section three summarizes the behavior model on which* the prototype design is based. The prototype implementation is described in Section four and the research results are discussed in Section five.

The Solution Overview

The objective of our research was to demonstrate the suitability of using the AI blackboard paradigm to integrate both procedural and AI techniques with a simulation system in order to achieve required CGF capabilities. To achieve this objective, we designed and implemented a prototype system that integrates a vehicle motion simulation testbed (IBM's Combined Arms Combat Simulator (CACS)) with a command decision system implemented using the AI blackboard paradigm. *This design demonstrates the features of the* blackboard paradigm for emulating vehicle behavior and unit cooperation within the bounds of a representative battlefield exercise.

The prototype implements the tactical behavior and coordination of a platoon of vehicles responding to a dynamic battlefield while conducting a single operation. Through this constrained focus, we rapidly prototyped a demonstration that validates the concept of a blackboard CGF solution while having a small impact on research costs and time. Although the scope of this research is constrained, further enhancements can be incorporated to include more complex behaviors to implement a more complete CGF solution.

The prototype system, as shown in Figure 1, is composed of the CACS simulation testbed implemented using the C language, and the "Command Decision System" implemented using the AI blackboard paradigm. The CACS simulates entities (e.g., tanks, helicopters, infantry, etc.) navigating on a plan view terrain map

while avoiding obstacles, following routes, and firing weapons as specified manually.

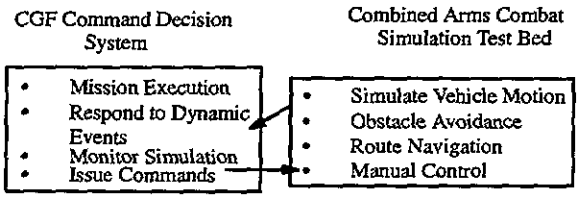

Figure 1 The demonstration prototype high level architecture

Functionally, the command decision system implements a task–oriented, hierarchical behavioral model as described in Le.[3] This behavioral model includes the ability to simulate the collective behavior of combatant agents cooperating toward an objective and the ability to simulate individual agents exhibiting individual behavior in reaction to local terrain and battle conditions. This behavioral model motivates the design of the command decision system as further described in Section 3.

By utilizing this behavioral model, the command decision system can emulate decision processes of some command level (for example; a platoon leader) for some designated mission. The command decision system monitors the events simulated by the CACS, makes decisions based on these observations, refines these decisions into actions, and issues commands to the CACS to implement these action. Because the command system provides automatic control over the CACS, the need for manual control is greatly reduced. To offer flexibility in operator control, however, the capability of overriding the commands from the command system is provided.

The prototype command system is implemented using the "Generic Blackboard Framework" (GBB – Trademark of Blackboard Technology Group, Inc.) which extends the Common Lisp and CLOS (Common Lisp Object System) environment on an IBM RISC/6000. This implementation integrates knowledge sources that utilizes both functional and procedural processes implemented using Common Lisp, rule–based processes implemented using the GBB–OPS language, object–oriented processes implemented using CLOS, and algorithmic processes implemented using the C language.

Summary of Results

The prototype system demonstrates the use and flexibility of the AI blackboard paradigm for the CGF problem. This system utilizes the blackboard features of integrating several different programming methodologies, providing adaptive behavior using a context driven control strategy, and providing an extensible solution for enhancing the CGF system.

BLACKBOARD TECHNOLOGY OVERVIEW

The blackboard architecture is an extension to the classical expert system structure as shown in Figure 2. The blackboard model has three components: a blackboard data structure, a set of knowledge sources, and a controller. Each knowledge source of this architecture is an intelligent agent that can solve some subproblem and uses the blackboard as its communication medium to other knowledge sources.

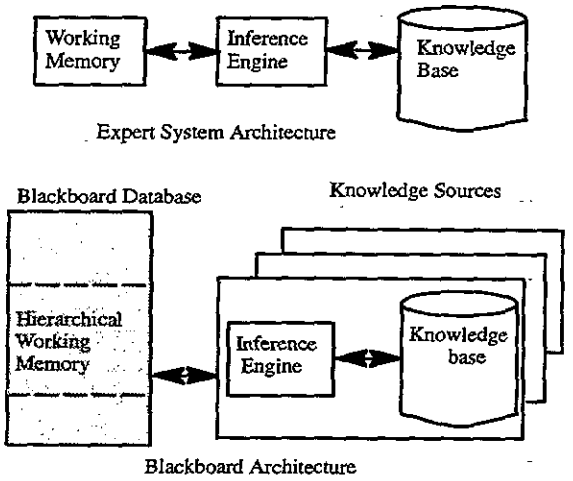

Figure 2 Expert system versus blackboard architecture

The Knowledge Sources

Each knowledge source in the blackboard paradigm is a self–contained specialist of a problem subtask. Each will solve the subtask independently of the other knowledge sources. New knowledge sources can be added to the system without any change to the other knowledge sources.

Each knowledge source is separately implemented as a rule–based program, a logic program, a functional program, or a procedural program using any of the supported languages. A knowledge source contains the knowledge needed to solve some subproblem and the mechanisms for applying that knowledge to the subproblem. During execution, a knowledge source can access information contained on the blackboard and modify the blackboard to record its results. The problem solving control of the blackboard paradigm is dis-

tributed in that each knowledge source is itself responsible for determining if conditions are satisfied that allow it to contribute to the problem solving.

The Controller

The control strategy of the *rule matching – conflict resolution – rule execution* cycle from classical expert systems is relatively complex compared to the control strategy used for the blackboard paradigm. Specifically, the blackboard controller collects all knowledge sources that indicate their readiness to contribute to the solution, adds them to an ordered queue regulated by each knowledge source's evaluation of its own importance, and sequentially gives control to each knowledge source in the queue. In contrast to classical expert systems, a knowledge source — not the controller — decides when it is ready to participate in problem solving and determines the importance of its contribution. These decisions are made by the knowledge source because only it has the knowledge and the means for using that knowledge to determine its own contribution to the solution.

The Blackboard Database

During the problem solving process, the blackboard database is a repository of all information, hypotheses, partial solutions, and problem solving state. Knowledge sources may access the data on the blackboard, remove data from the blackboard, or add new data to this structure. Similar to the working memory of a classical expert system, all communications and interactions between the knowledge sources are achieved through the blackboard database. However, unlike the classical expert system, the data on the blackboard may be represented in many different ways and at different levels of abstraction.

The Blackboard Operation

For this paper, this section describes one operational view of the blackboard paradigm. Many variations of the blackboard paradigm operation exist but will only subtly effect the performance of the CGF blackboard prototype system.

The operation of the blackboard paradigm is illustrated in Figure 3. Activities on the blackboard database such as adding, removing, and modifying information or objects are events that may cause particular knowledge sources to respond. Other special events may be initiated by the blackboard controller or by some knowledge source's execution. For each knowledge source defined for an application is a list of events that cause the knowledge source to respond.

Each knowledge source that responds to a given event during the system's operation, is asked by the controller if it should be activated. A knowledge source's decision to be activated and its evaluation of the importance of its activation are based on the knowledge source's evaluation of this information contained on the blackboard. Thus, the activation of a knowledge source not only depends on the events occurring on the blackboard but also the context in which the events occur.

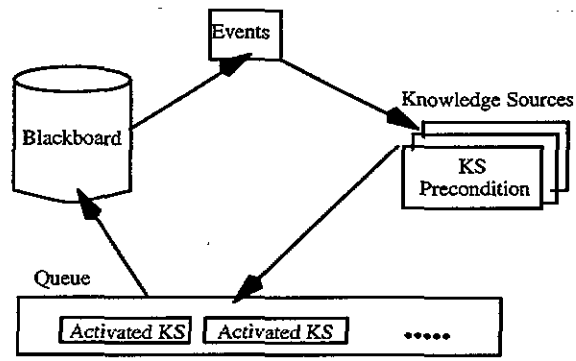

Figure 3 The blackboard system operation

The controller places each activated knowledge source on a queue according to the knowledge source's importance. The first knowledge source on the queue then executes and records its contribution to the solution on the blackboard database. This modification to the blackboard creates new events and the controller cycles until it is told to stop or the queue becomes permanently empty.

Distinguishing Features of the Blackboard Paradigm

Although the blackboard paradigm has some features in common with expert systems and object–oriented programming, there are important features that distinguish the blackboard paradigm for the CGF problem. The blackboard paradigm provides both event and context driven programming while expert systems are more data and goal driven. Although object–oriented programming is event driven where events are actions on objects, its applications are only decomposed in terms of objects and its components. In contrast, the blackboard paradigm allows the integration of both an object decomposition and a functional decomposition of a problem. In addition, for a complex object–oriented application, it can be difficult for the developer to schedule a long sequence of method activations in re-

sponse to an event. In contrast, the blackboard paradigm's controller explicitly schedules the multiple knowledge sources that respond to a single event.

The blackboard paradigm has other advantages over the classical expert system by rectifying weaknesses found in classical expert systems. Each knowledge source can be implemented using the most appropriate paradigm for solving its subproblem. That is, the knowledge source can itself be a rule–based expert system, a logic program, a C language procedure, or a set of Lisp functions. Therefore, the expert system shell's requirement of solving all parts of the problem using the same problem solving method and the same knowledge representation is removed.

This characteristic is particularly useful for solving the CGF problem. A solution must use a variety of both algorithmic and non–algorithmic tasks including mission planning, situation assessment, data fusion, decision making, decision refinement, route planning, monitoring, line of sight computations, and many more. Some of these tasks are best solved procedurally while others suggest a knowledge–based approach.

In addition, the blackboard paradigm allows the cooperative integration of these different problem solving approaches which use and produce data, hypothesis and solution states having different representations and defined on different levels of abstraction. The blackboard paradigm allows these alternative formats to coexist on the blackboard for use during problem solving.

The blackboard paradigm's event and context driven control strategy is also a requirement of the CGF system. By representing the simulated battlefield on the blackboard, any events occurring on the battlefield correspond to events occurring on the blackboard. Each knowledge source reacting to these events can react to the context on the battlefield in which these events occur since this context is represented on the blackboard. The correspondence between the operation of the blackboard and the cognitive unit behavior to battlefield situations will provide the ability to simulate units adaptively reacting to the dynamic battlefield.

THE CGF BEHAVIORAL EMULATION

Generalized Combat Model

The goal of modeling automated forces is to develop both the ability to simulate collective behavior of combat units and the individual combat agents' behavior of reacting to local terrain and battle conditions. The agents defined by this model must be able to mimic the cognitive capabilities of their human counterparts by perceiving their environment, updating and maintaining a model of the developing tactical situation, planning actions, reacting to dynamic situations, monitoring their execution, and communicating to other agents. Also, to be effective, this generalized model must not only characterize the behavior of a particular type of combat agent but also specify a functional model whose instantiation can be used to simulate a variety of combatants including tanks, infantry, air support, and so on.

As shown in Figure 4, the major functional elements of this model include basic vehicle movement and maintenance, obstacle avoidance, unit motion coordination, military mission/task execution, and intelligence/planning. These elements are arranged hierarchically in a manner similar to a subsumptive approach.[1] However, the difficulty of using a purely subsumptive approach is that it only solves the reactive role of the combat agents (i.e., event driven) whereas in a complex battlefield environment, acceptable behavior depends on the battlefield and the operational context in which the agents must react.

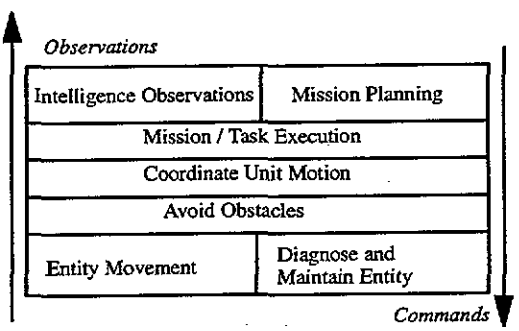

Figure 4 The generalized combat model

To achieve this capability, the hierarchical model must be adapted to include a task–oriented approach similar to the one used by Schudy and Duarte[4]. Therefore, behavior at each hierarchical level must be defined in terms of not only the type of agents but also the type of battlefield operations conducted by these agents.

This behavioral model is also based on the conventional communication paths between cooperative entities and command levels. Within the military command echelon, commands are disseminated "down" the chain of command. Plans from the highest level are refined into tasks and then into actions as they are passed to lower levels in the echelon. The combined behavior achieved through these actions will implement the original plan. At the higher levels in the echelon, information is abstracted from observations occurring at lower levels

and is utilized to construct the mission plans. By accumulating observations "up" and passing plans "down" the command echelon, the entities will cooperate to achieve a mission objective.

The Battlefield Exercise

Before designing and implementing the prototype system for this research effort, we defined a set of operations and behaviors that the prototype's units would emulate. These requirements are specified within a battlefield exercise that bounds the set of expected events and the set of expected behaviors to a manageable set for the research activity.

The command system designed and implemented for this research task emulates the behavior of multiple platoons conducting independent zone reconnaissance operations. Opposing forces are controlled manually by an operator to create various scenarios that test and demonstrate the automated platoons' responses to dynamic events. The *operation order* requires each platoon to conduct a zone reconnaissance in a bounded zone containing three phase lines, coordinate progress at these phase lines, and return from the third phase line to the line of departure via the most expeditious route. The purpose of the reconnaissance mission is to detect, classify, and react to any opposing forces sighted from within the reconnaissance zone but not to engage these opposing forces.

PROTOTYPE SYSTEM ARCHITECTURE

The behavioral model for the command echelon is used to design the command system that emulates the performance of a cooperative group of entities (e.g., a platoon). As described in Figure 1, the system architecture is composed of two processes: the CACS and the command system.

The CACS was not developed prior to this research effort. From an AI perspective, the route following and obstacle avoidance techniques implemented in the CACS are interesting and are reported by Le.[3]

Blackboard Organization

To emulate the platoon units on the battlefield, the command decision system must be able to observe events occurring on the battlefield, determine effective responses to those events, refine those responses into vehicle actions, and communicate those actions to the CACS.

The command system is organized using abstraction levels on the blackboard. These levels and the knowledge sources that interact with each level are abstractly represented in Figure 5. Information represented at a particular level of the blackboard corresponds to the information needed by an associated level of the military command echelon. For example, a vehicle's commander must be aware of his route to an identified objective while a platoon leader must be aware of the area where the platoon is actively operating.

The blackboard database is divided into two primary levels: the command level and the simulation supervisor level. The command level contains all information affecting the command decisions for the emulated units. The simulation supervisor level contains all battlefield information obtained from the CACS and commands sent to the CACS. The command level is further divided into levels corresponding to levels of a military echelon (for example, battalions, companies, platoons, and platforms). For the implemented prototype, the command level contains the platoon and platform levels.

The command level of the blackboard contains the organization of each unit controlled by the command system and any information affecting that control. For example, objects at this level describe a platoon's organization as composed of specific platforms (i.e., tanks, infantry, armed fighting vehicles, etc.).

The simulation supervisor level of the blackboard contains information about the simulated battlefield and every vehicle on the battlefield as simulated by the CACS. Therefore, the simulation supervisor level contains perfect information about the battlefield and activities occurring on the battlefield. To accurately emulate the performance of manned vehicles, the decisions made at the command level should only be based on information realistically available to the manned vehicles. For example, vehicles that are more than 3500 meters from the controlled units cannot realistically be seen by those units. Therefore, command decisions for these units should not be based on the existence of these vehicles.

This motivates the design decision of dividing the blackboard into the command level and the simulation supervisor level. The simulation supervisor level serves to *hide* information that the controlled units cannot realistically utilize. There are no features of the blackboard COTS product used for this study that provides secured access between these blackboard levels. Therefore, information hiding is achieved through design alternatives and programming discipline. For this prototype system, the knowledge sources at the simula-

tion supervisor level choose the information that should be available to the command level and *filters* this information to the command level of the blackboard.

Knowledge Source Responsibilities

As shown in Figure 5, knowledge sources are associated with a particular level of the blackboard. However, knowledge sources can abstract or refine information from one level of the blackboard to another level. For example, the "Filter Information" knowledge source moves entity detection information to the Command level only if a controlled entity can make that detection.

Simulation Supervisor Level: The set of knowledge sources at the supervisor level accomplish three tasks. One group of knowledge sources monitors the status of the battlefield and the entities simulated on the battlefield. Another group sends commands to the CACS that affect the entities simulated on the battlefield. The last group of knowledge sources filter information from this blackboard level to the command level based on the ability of the emulated units to realistically perceive this information. The knowledge sources at the supervisor level are either algorithmic or functional processes as represented in Figure 5 as processes defined by rectangular boxes.

Platform Command Level. The platform sublevel is the lowest level of the command system and contains information about each of the controlled platforms. The knowledge sources at this level specifically control a single platform as an independent unit. Since most of the platform dynamics are implemented as part of the CACS, the knowledge sources emulating a platform in the current prototype provide the automatic routing and communications capabilities of each platform.

Platoon Command Level. The platoon sublevel of the blackboard contains information about the organization of the platoon and the platoon's current operation. For this prototype system, the knowledge sources at this level emulate a zone reconnaissance operation for a platoon of M3 calvary fighting vehicles. The activities implemented by the knowledge source include coordinating the platoon at the zone's phase lines, reacting to opposing force detections and classifications, and reacting to opposing force attacks.

The "M3 Platoon Recon" knowledge source is a rule–based program, as indicated in Figure 5 by the ellipse, that emulates the decisions of a platoon leader responding to battlefield events. For the reconnaissance mission, these events occur when a platform reaches a control measure route way point, when a platform detects or classifies an opposing force, and when a platform detects a munitions firing. The M3 platoon reconnaissance knowledge source is responsible for creating the platoon's response to the battlefield events within the parameters and tactics defined for the platoon's mission. In general, a knowledge source at this level is responsible for emulating a specific type of platoon conducting a specific operation.

Design Summary

Every object on the command level of the blackboard refers to a particular vehicle, platform, or platoon leader. This includes the opposing force detections, firing detections, and control measure way points. Each knowledge source activation is created in response to some event affecting these objects and is therefore

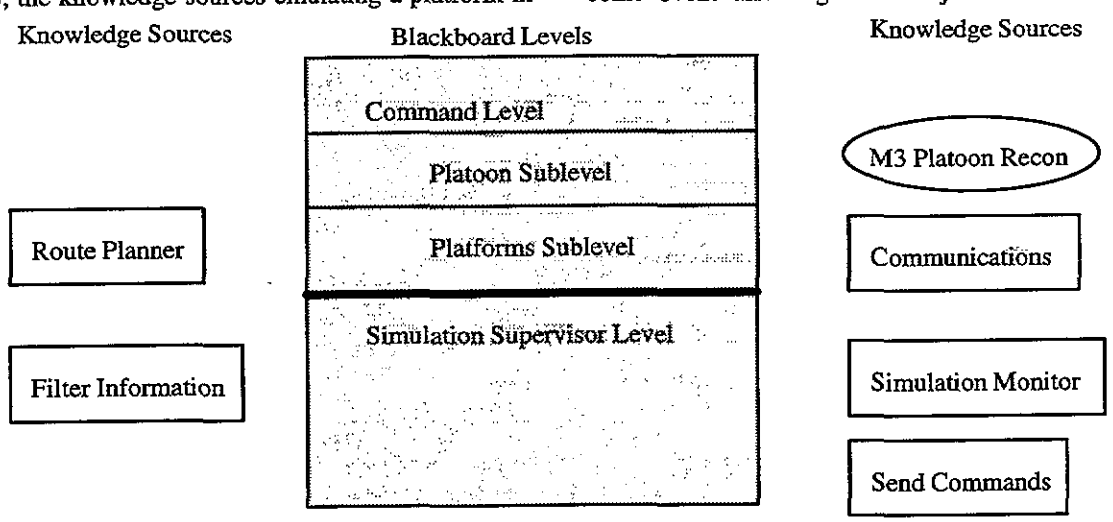

Figure 5 The command system architecture

associated with a single platoon. Thus, with no additional code and utilizing the same blackboard, these knowledge sources can control multiple independent platoons without incurring any interactions between these multiple platoons. For example, multiple M3 platoons, each conducting reconnaissance operations, will behave independently. While one platoon is responding to a firing detection, a second platoon will continue its reconnaissance operation without any influence from the activity of the first platoon.

RESULTS AND CONCLUSIONS

The automated control of forces for the CGF problem is similar to the classic Command, Control and Communications (C3) problem. In its general form the command and control components for CGF should have a hierarchical structure that matches a military echelon. The highest level of command and control must process and fuse data from different sources, assess the battlefield situation based on that data, plan mission objectives to react to the battlefield situation, and allocate the resources required to achieve those objectives. The next levels in the command and control model for CGF must progressively refine these objectives into tasks and then into actions at the unit level on the battlefield. The lower levels must also pass abstractions of their battlefield observations up the hierarchy to aid the decision making processes at these higher levels.

Therefore, to effectively emulate a C3 model, the system should include tasks such as data fusion, situation assessment, planning, decision making, task refinement, action execution, battlefield monitoring, detection, classification, routing, and others. However, the decision process for each unit must be able to "opportunistically" respond to events occurring on the battlefield within the parameters defined by doctrine and the current situation (i.e., the context).

The results of research reported in this paper show that a blackboard paradigm can provide the control of the computer generated forces using a C3 model. To evaluate the concept of using the blackboard paradigm for the CGF problem, a blackboard CGF prototype was designed and implemented to demonstrate the relevant blackboard features including its integration of different programming methodologies, its event and context driven control strategy, and its extendibility for the CGF problem.

Programming Paradigm Integration

The implementations of all tasks defined by the Command, Control and Communication model are not well suited to a single programming method. That is, some C3 tasks are best solved algorithmically such as computing the line of sight between two battlefield entities. Other tasks such as situation assessment are best solved using a knowledge–based approach. The variety of processes defined for C3 requires the integration of several problem solving methodologies into a cooperative system.

As provided by the blackboard paradigm, the prototype system utilizes the ability to integrate multiple programming paradigms. This allows each subtask to be implemented using an appropriate programming method. For example, decision making involved in reacting to firing events, opposing force detection events, and platoon coordinate events were implemented using rule–based programming. Likewise, the line of sight and routing algorithms that require fast numeric computations were implemented as conventional, compiled procedures.

By utilizing the blackboard paradigm's common communication medium (the blackboard) and its control strategy that is distributed among the independent knowledge sources (each implemented using the most appropriate methodology), these alternative programming methods were easily integrated to contribute to the CGF solution. This flexibility allows the design of each CGF component to match a cognitive model for the operation of actual battlefield forces.

Context and Event Driven Behavior

At any instant during the battlefield simulation, events may arise that require a response from the emulated entities on the battlefield. These events can include an entity's arrival at some key terrain location (e.g., an assembly area), a detection of some opposing force, a detection of a munition firing, the arrival of some key time event, and many others. With no other complexities, an event driven simulation system could adequately cope with this requirement. However, in the CGF simulation many more variables are introduced.

The variety of events that must be handled by the control system and the variety of responses available with respect to the battlefield context, greatly increases the complexity of an event driven control strategy for the simulation. That is, the different combinations of events, contexts, and sequences in which the events occur represent a combinatorial control problem. Prede-

fined sequential control strategies (e.g., procedural or finite state automata approaches) for this problem will not only be very complex, but will also provide a fixed (or static) behavior to an event because they cannot account for all contexts and sequences in which the event occurs. To provide adaptive behavior, the system must be able to easily and correctly respond to battlefield events within the context of the battlefield.

The blackboard data structure in the prototype system abstractly represents the simulated battlefield including terrain, entities, units, detections, and so on. Events on the battlefield correspond to events on the blackboard which activate particular knowledge sources. In response to these events, the knowledge sources individually evaluate the battlefield context represented on the blackboard to determine if they should respond to not only the event but also the context in which the event has occurred. Therefore, by using the context and event driven control strategy provided by the blackboard paradigm, the control strategy of the prototype is the simple control loop described in Section 2. Also by using this control strategy, the demonstration prototype adaptively responds to the dynamic battlefield events.

Extendible CGF Design

A CGF system can have many uses including training, evaluating battlefield tactics, and evaluating the effectiveness of new units or weapons. To provide capabilities for these uses, the system must be designed to allow the incorporation of new unit types, new weapon types, and new battlefield tactics. That is, the system must be modular and easily extendible.

By implementing separate subtasks as knowledge sources that are independent and interact with other subtasks only through a blackboard, the blackboard paradigm provides the means for making the system very modular and extendible for added units, tactics, and operations. By encoding specific subtasks into separate, independent knowledge sources, the system can be functionally extended by adding new knowledge sources. These knowledge sources can be designed by acquiring only the expertise that is relevant to the new task rather than expertise about the entire CGF problem. For example, to incorporate a new platform type we need only interview an expert on the tactics and capabilities of that new platform.

To construct a system that is easily extendible and modifiable, the CGF system architecture should conceptually model the architecture and behavioral model of the emulated military echelon. The hierarchical nature of the blackboard data structure provides a skeleton upon which this behavioral model can be constructed. The blackboard can be divided into levels which segregate and abstract the data and decisions among the appropriate command levels. The independent knowledge sources can provide the appropriate behavior for each entity on each level of the military echelon.

This demonstration prototype can easily be extended both horizontally, providing new platoon unit types and operations, and vertically, providing the higher command levels of company, battalion, and so on. The knowledge sources associated which each level can be responsible for making decisions relevant to that level, refining decisions from high levels into actions, or abstracting observations from lower levels. The command echelon, the decision and mission refinement down the echelon, and observation abstraction up the echelon are all explicitly represented within this blackboard design. This provides a common model of battlefield command operation that should enhance the extensibility and modifiability of the system.

The Command System Implementation

The prototype command system (not including the CACS) contains approximately 1,100 lines of C code. This code implements the computation intensive tasks (e.g., route planning) and the system's communication with the CACS. The system also contains approximately 4,200 lines of Lisp code. This code specifies much of the blackboard and knowledge source definitions, and implements the basic blackboard manipulation functions such as filtering information on the blackboard. The "M3 platoon reconnaissance" knowledge source contains 38 rules and 400 lines of supporting Lisp code. The command system also defines 30 different types of objects that can be created on the blackboard for this implementation.

SUMMARY

The research task described in this paper addressed the requirement for an architecture that allows the integration of various CGF subtasks, provides a realistic emulation of multiple battlefield entities, and allows the extensibility required of a CGF solution. The AI blackboard paradigm offers features that satisfy these requirements. The research confirms this hypothesis by demonstrating a blackboard CGF prototype system that emulates multiple, independent M3 platoons conducting zone reconnaissance operations.

REFERENCES

1. Brooks, R., A robot layered control system for a mobile robot. *IEEE Journal of Robots and Automation*, RA–1(1), pages 1–10, 1986.

2. Engelmore, R. and Morgan, A., editors. *Blackboard System*. Addison Wesley Publishing, California, 1988.

3. Le, H. T., Phinney, S. E., and Seward, V. C. Semiautomated forces: A behavioral modeling approach. In *The Proceedings of the Thirteenth Interservice/Industry Training Systems Conference*, December 1991.

4. Shudy, R. B., and Duarte, C. N. Advanced autonomous underwater vehicle software architecture. In *The Proceedings of the Symposium on Autonomous Underwater Vehicle Technology – AUV'90*. Washington, D.C.,1990, pages 9–22.