

COMMAND DECISION MODELING TECHNOLOGY ASSESSMENT

31 JULY, 1996

COMPILED BY THE
US ARMY
ARTIFICIAL INTELLIGENCE CENTER

FOR THE
NATIONAL SIMULATION CENTER

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

DIRECTOR,
NATIONAL SIMULATION CENTER
ATTN: ATZL-NSC
410 KEARNY AVENUE, BLDG 45
FORT LEAVENWORTH, KS 66027-1306

DIRECTOR,
US ARMY AI CENTER
ATTN: SAIS-AI (RM 1D659)
107 ARMY PENTAGON
WASHINGTON, DC 20310-0107

[Return to the Top of the Page](#)
[Return to the Table of Contents](#)
[Return to the SCC Command Decision Modeling Page](#)
[Return to the WARSIM 2000 Page](#)
[Return to the NSC Home Page](#)

DTIC QUALITY INSPECTED 3

19980121 085

AD
JR

New Text Document.txt

14 January 1998

This paper was downloaded from the Internet.

Distribution Statement A: Approved for public release;
distribution is unlimited.

POC: US Army AI Center
Date: 31 July 1996

Executive Summary

BACKGROUND.

This assessment was prepared at the request of the Director of the National Simulation Center (NSC). The purpose of this assessment is to provide executives and managers a concise overview of the strengths and weaknesses of a number of proposed technical approaches to modeling the command decision process. The specific and most immediate area of application is automated augmentation of training simulations such as WARSIM 2000. However, command decision models have broad applicability to state-of-the-art and future command and control systems as well as to analytical simulations in support of program, budget, and materiel development.

METHOD.

NSC identified a number of proposed technologies to automate command decision modeling. The US Army Artificial Intelligence Center (USAAIC) identified recognized experts in each technology and arranged for them to provide assessments of the strengths and weaknesses of each technology. In addition, NSC identified several implemented systems that incorporate some level of command decision modeling for technical assessment of approach and potential for extension as more comprehensive command models. USAAIC developed a recommended format which specified key technical issues for the assessors to address, and each assessment has undergone a thorough cycle of technical review and edit. However, all revisions were reviewed by the original assessor, and in all cases, differences of opinion were resolved in the assessor's favor, with foot notes indicating alternative or dissenting views.

RESULTS.

Based on Wohl's SHOR framework, described in "Command Decision Modeling" (paper BG 1), command decision modeling is divided into four subprocesses, with a fifth subprocess, "Explain", added to address this important requirement of training systems. The strength of each technology and system to automate each command decision subprocess has been estimated based on the technical and systems assessments. Table ES-1, below,

displays these ratings, using $\ddot{+}$ to indicate a strong potential for modeling a function, and $\ddot{-}$ to indicate a weak capability. The reader is directed to the assessments for additional details.

COMMAND DECISION MODEL FUNCTION

TECHNOLOGY	STIMULUS	HYPOTHESIS	OPTIONS	RESPONSE	EXPLAIN
RULE-BASED	-	+	-	+	+
FUZZY LOGIC	+	-	-	+	-
CASE-BASED	-	+	-	+	-
GENETIC ALGORITHMS	+	+	+	-	-
NEURAL NETS	+	-	-	+	-
LATTICE AUTOMATA	-	-	+	+	-
PLANNING	-	+	+	+	+
PETRI NET	-	-	+	+	-
SYSTEMS					
EAGLE A.P.	-	+	+	+	+
GEKNOFLEXE	TBD	TBD	TBD	TBD	TBD
SOAR	-	+	+	+	+

TABLE ES-1. SUMMARY OF ASSESSMENT RESULTS.

[Return to the Top of the Page](#)
[Return to the Table of Contents](#)
[Return to the SCC Command Decision Modeling Page](#)
[Return to the WARSIM 2000 Page](#)
[Return to the NSC Home Page](#)

COMMAND DECISION MODELING
TECHNOLOGY ASSESSMENT

Section 1

INTRODUCTION:

*** ABOUT THIS ASSESSMENT**

INTRODUCTION:

ABOUT THIS ASSESSMENT

EDITOR:

Major David Payne

Chief Technology Officer

US Army Artificial Intelligence Center

ATTN: SAIS-AI, Rm 1D659

107 Army Pentagon

Washington DC 20310-0107

Phone: 703-614-6900

FAX 703-614-6908

email:

payned@pentagon-ai.army.mil

BACKGROUND.

This assessment was prepared at the request of the Director of the National Simulation Center (NSC). The purpose of this assessment is to provide executives and managers a concise overview of the strengths and weaknesses of a number of proposed technical approaches to modeling the command decision process. The specific and most immediate area of application is automated augmentation of training simulations such as WARSIM 2000. However, command decision models have broad applicability to state-of-the-art and future command and control systems as well as to analytical simulations in support of program, budget, and materiel development.

METHODOLOGY.

NSC identified a number of proposed technologies to automate command decision modeling. The US Army Artificial Intelligence Center (USAAIC) identified recognized experts in each technology and arranged for them to provide assessments of the strengths and weaknesses of each technology. In addition, NSC identified several implemented systems that incorporate some level of command decision modeling for technical assessment of approach and potential for extension as more comprehensive command models. USAAIC developed a recommended format which specified key technical issues for the assessors to address, and each assessment has undergone a thorough cycle of technical review and edit. However, all revisions were reviewed by the original assessor, and in all cases, differences of opinion were resolved in the assessor's favor.

assessor's favor.

OVERVIEW.

In addition to the technology area and system assessments, several introductory papers are included in this first edition of the Command Decision Modeling Technology Assessment. This paper gives the reader an overview of the assessment, including the reasons why the assessment was conducted, the methodology used, and advice on where to find specific results. Dr. Fred Buoni of Florida Tech provides an overview of the process being modeled, the Command Decision Process, in a paper drawing on over a decade of work in this field. In the next paper, Dr. Kent Bimson and his colleagues at SAIC's Orlando Florida office propose a system design for a generic command decision model or command agent, clarifying the basic required functions using a block diagram approach. The final introductory paper is from Dr. David Jenson of Pacific Sierra Research, Inc. (PSR). Dr. Jenson draws on PSR's experience supporting the intelligence community with sophisticated systems to help discern and predict the actions of opposing force commanders to point out some efficient technical approaches to modeling opponents when specific command agent functions, such as communications with human commanders or players during the exercise and interactive participation in after action reviews, are not required to accomplish the training goals of a system or exercise.

Following the Introductory papers are the Technology Area Assessments. These papers address a specific technology thought to be helpful in building command decision models and/or command agents. The authors follow a standard format, which is intended to help the reader quickly understand the key details, strengths, and weaknesses of each technical approach.

Similarly, the System Assessments, in the next section, give the reader a concise overview of several implemented training systems that model the command decision process. Once again, the format is intended to speed understanding, and wherever appropriate, follows the format of the Technology Area Assessments.

RESULTS.

Based on Wohl's SHOR framework, described in "Command Decision Modeling" (paper BG 1), command decision modeling is divided into four subprocesses, with a fifth subprocess, "Explain", added to address this important requirement of training systems. The strength of each technology and system to automate each command decision subprocess has been estimated based on the technical and systems assessments. Table 1, below, displays these ratings, using "+" to indicate a strong potential for modeling a function, and "-" to indicate a weak capability. The reader is directed to the assessments for additional details.

COMMAND DECISION MODEL FUNCTION

TECHNOLOGY	STIMULUS	HYPOTHESIS	OPTIONS	RESPONSE	EXPLAIN
RULE-BASED	-	+	-	+	+
FUZZY LOGIC	+	-	-	+	-
CASE-BASED	-	+	-	+	-
GENETIC ALGORITHMS	+	+	+	-	-
NEURAL NETS	+	-	-	+	-
LATTICE AUTOMATA	-	-	+	+	-
PLANNING	-	+	+	+	+
PETRI NET	-	-	+	+	-
SYSTEMS					
EAGLE A.P.	-	+	+	+	+
GEKNOFLEXE	TBD	TBD	TBD	TBD	TBD
SOAR	-	+	+	+	+

SUMMARY OF ASSESSMENT RESULTS.

WEAKNESSES OF THIS ASSESSMENT:

As with most attempts at a comprehensive survey of a complex subject, especially those constrained by time and budget, this Command Decision Modeling Technology Assessment falls short of being a final, definitive document in several ways. First and fundamentally, the subject area itself is very much an open field of active, ongoing, research. Hence no survey of command decision modeling conducted today can be considered the final word--indeed, a paper published next month may reveal breakthroughs or document findings that negate one or more of the assessments provided here. Secondly, the sponsors realized that the funding available would support only a limited survey of the field. The decision here, then, was to do as comprehensive, yet concise, assessment as possible for a limited subset of command decision modeling approaches. Still, funding supported assessments of virtually all technologies for command decision modeling being proposed for systems being designed today for fielding early in the next century, and as it turns out, essentially all Artificial Intelligence approaches to the problem.

What is not assessed are the Operations Research based, numerical, approaches that generally promise high computational efficiency, typically accompanied by lower processor and memory loads and faster solution times. However, all of the proposed mathematical

technologies suggested to the assessment management are not implemented in code for command decision modeling. Most involve reviving lines of inquiry that were set aside in the early 1980s, when automated command decision aids and decision support systems fell out of favor with the tactician-users. However, many of these approaches are based on sound theory, often implemented in analytical systems supporting other application areas. A large problem with the systems in the early 1980s was due to the limitations of available computers at that time. The machines then available, mostly minicomputers, generally lacked the processing speeds and memory to solve the decision problems in the time required to support tactical combat decisions. Moreover, the physical size and support requirements of the machines precluded practical use in ground combat environments. With today's laptops offering level of magnitude and higher improvements in speed and memory, there is a possibility the old techniques have much to offer the command decision modeler.

A final weakness is that the assessments were done concurrently, due to the short time allotted. The authors had to prepare the assessments without knowledge of what other authors were covering in their assessments. Hence there may be actual or implied contradictions between the different assessments. The reader is therefore advised to take each assessment with the proverbial grain of salt, since scientific opinion in most of the technology areas assessed has in fact not yet reached a consensus.

FUTURE WORK IN THIS AREA:

For the reasons stated above, one recommendation to the sponsors is to continue this assessment process at least one more year. This will allow conflicting opinions stated in the assessments to be discussed, debated, and possibly resolved, which in turn will support improved revisions of the assessments done. In addition, other technologies can also be assessed and included in the assessment collection.

Recommendations for technologies to explore include advances in scheduling algorithms, both from Operations Research optimization technology as well as Artificial Intelligence based search and heuristic approaches. Game theory based decision aids may also provide some useful technology for command decision modeling, as well as techniques from modern decision support models. Constructive simulation models could also conceivably help a machine make good command decisions, through playing out courses of action at very high speeds, not unlike modern chess problems do to project implications of moves. Finally, influence diagram solution engines that have solved some operational problems in distribution systems and other complex domains may hold some useful tricks for the command decision modular.

In the system assessments, a follow-on assessment should include looks at the two industry entries in the Command Forces (CFOR) arena, both of which are producing some emerging results as well as lessons learned.

A WORD OF THANKS:

As Editor of this assessment, I would like to extend a sincere thank you to the authors and their managers, numerous contracting officials at US Army Simulation and Training

Command and Defense Supply Service Washington, and the staff of the Army Model Improvement Program for the support they have provided me in coordinating and producing this assessment.

[Return to the Top of the Page](#)

[Return to the Table of Contents](#)

[Return to the SCC Command Decision Modeling Page](#)

[Return to the WARSIM 2000 Page](#)

[Return to the NSC Home Page](#)

COMMAND DECISION MODELING TECHNOLOGY ASSESSMENT

Section 2

BACKGROUND:

- . COMMAND DECISION MODELING: An Overview
- . PROBLEM DEFINITION: Functional Description of a Command Agent
- . OBSERVATIONS REGARDING THREAT COMMAND AGENTS

BG1 Command Decision Modeling:

AN OVERVIEW

Frederick B. Buoni, Ph.D.

Florida Institute of Technology
Melbourne, Florida 32901
(407) 768-8000, ext. 7390
eMail: buoni@zach.fit.edu

SUMMARY

An architecture of a general decision process is constructed within the SHOR framework (Stimulus, Hypothesis, Options, Response) of Wohl. The process is decomposed into situation assessment and option assessment phases. The situation assessment process is composed of preprocessor, hypothesis generation, and hypothesis assessment subprocesses, while the

option assessment process is composed of criteria, option generation, option processing, and decider subprocesses. The context for the overall decision process is a command and control structure, but the architecture should apply to any decision-making situation.

The decision process architecture provides a framework for understanding the developments which are reported in the papers which follow this one.

SECTION I - INTRODUCTION

In this paper we present a general architecture for an Army tactical command and control (C2) process which will enable the development of computer generated forces (CGF) for use in Distributed Interactive Simulation (DIS). This architecture presents a framework which allows the development of components using the applications of artificial intelligence (AI) and computer-based decision systems. The emphasis is on viewing the C2 process as a decision process.

This work is based on a framework for tactical decision making which the author developed while on a summer faculty fellowship at the Jet Propulsion Laboratory in support of the Airland Battle Advanced Technology (ABAT) program at JPL [BUON83]. While in the original work, the emphasis was on providing decision support to a human decision maker, this effort is directed to support for computer decision making either under the supervision of a human controller or autonomously. While we retain the point of view that the system is computer-aided decision making with the human as the ultimate decision maker, the extension to an autonomous system is merely that there is no human intervention and that the computer could continue to direct the implementation of the decision it recommends. We refer to the system as a decision making system (DMS) whether it is to be a human decision maker or an autonomous computer decision maker.

In this paper we construct an architecture of a general decision process and discuss how it may represent a typical Army command and control process. This discussion is not definitive but is meant as an initial consideration to stimulate further investigation and definition. This work establishes an architecture within which real C2 processes may be considered. The remaining papers in this volume will address the issues involved in using various techniques to implement components within this framework.

SECTION II - DECISION-AIDING NEEDS

The decision process has been studied by behavioral psychologists [JANI77], mathematicians [VONN44], operations researchers [VONW86, ZELE81], computer scientists [SIMO81], system engineers [GREE86, CACC92, REN 95, SAGE81, WOHL81], and military analysts [BEN-82, VRAN92]. The interest has broadened to include artificial intelligence researchers as we try to design systems for computer aiding of the decision process to respond to the needs of the military, business and the non-military governmental agencies [AZAR86, INGR92, POST90].

In today's world, a decision-making system (DMS) has a continually increasing amount of data available. Exotic sensor systems, global communications systems, and large data base systems provide the DMS with vast amounts of data. These same technologies also produce a

quicken world. The decision-maker must respond more quickly; there is less time to make decisions. This presents great pressure for the human decision-maker

[BARN81, ATHA83]. These pressures were exemplified by the Vincennes incident [BARR92].

The basic human decision-making capabilities remain limited. Short-term memory, the basis for perception and processing, is limited to two to four chunks of information, and it appears to take five seconds to place a chunk in long-term memory [SIMO81]. There are further difficulties: in retrieving information from long-term memory, response time is variable and cannot be guaranteed, and the information retrieved may be incomplete or altered.

The decision process itself can be conducted in various modes, some of which may be pathological [SAGE81, JANI77]. In a stressful situation it may be desirable to have a normative guide to assist in ensuring properly considered decisions.

Thus, the context for a DMS is established. Computer-based systems may be used to overcome human limitations in short-term memory, in rapid and accurate recall of data and information, and conducting the decision-making process. A framework for performing these tasks is the subject of the remainder of the paper.

An initial work for the Advanced Research Projects Agency (ARPA) examined the opportunity for decision-aiding for Command, Control, and Communication (C3) environments in [NICK77]. They also examined the tactical communication effort [MADN82]. The Army Research Institute developed the Tactical Operations System [LEVI77, WITU80] and considered AI in the context of situation assessment [BEN-82, SAGE82]. The Air Force has examined decision-making models in tactical C2 systems [BOET82, LEED79, LEVI82, PATT83, WOHL81], and in airborne information systems [CHU 82]. The Navy has examined the tactical C2 process from the anti-submarine warfare (ASW) view [WOHL83a, WOHL83b], the undersea warfare view [DAVI82, DEGR82], the surface warfare view [FUNK82], the airborne view [BARN81], and a general discussion of man-machine task allocation [RIEG82]. The Office of Naval Research (ONR) sponsored Sage's general survey of decision-aiding processes [SAGE81]. NASA has been interested in the process monitoring function [GREE82, HAMM82, ROUS81], and in decision making in the cockpit of an aircraft [CHAP93]. This is not meant to be a thorough review of current work in the field; it only presents some readily available references in the unclassified literature.

The early significant thread leads from the ARPA survey [NICK77] to the Sage review [SAGE81] to Wohl's establishment of a framework for decision process analysis [WOHL81]. We note that much work in this area was done in the 1980's, but much less in the 1990's. There now appears to be significant interest in applications of AI, but much less in normative decision modeling.

SECTION III - DECISION PROCESS ARCHITECTURE

GENERAL CONCEPT

The focus of this analysis is on the decision-making process performed by a decision-making unit, which is called the decision-making system or DMS. The unit would normally consist of a

leader or commander and one or more assistants or operators and associated equipment. At this stage of the analysis the decision process will be examined without attributing functions or tasks to individuals or machines within the decision unit.

The decision unit is considered to be one subsystem within a hierarchical, multilevel system as shown in Figure 1 [MESA70]. The decision unit receives guidance, represented by h , from its supervisory unit or higher headquarters, and returns status information s . It issues instructions and orders u to its control units which are in contact with the external environment, and receives responses and progress reports r . The control unit actually attempts to influence the external states of nature through efforts m , and receives observables q . The input to the actual contact with the environment is w and the ensuing state of the situation is y . While this model represents all of the elements of the decision situation, it is too complex for consideration at this stage. It does, however, establish the context for the view of the decision unit, and the types of interactions with other subsystems and the external environment. Contact with an opposing or enemy force is considered as part of the environment and is represented within the states of nature w and y . Similar views are presented in [LAWS79, ATHA83].

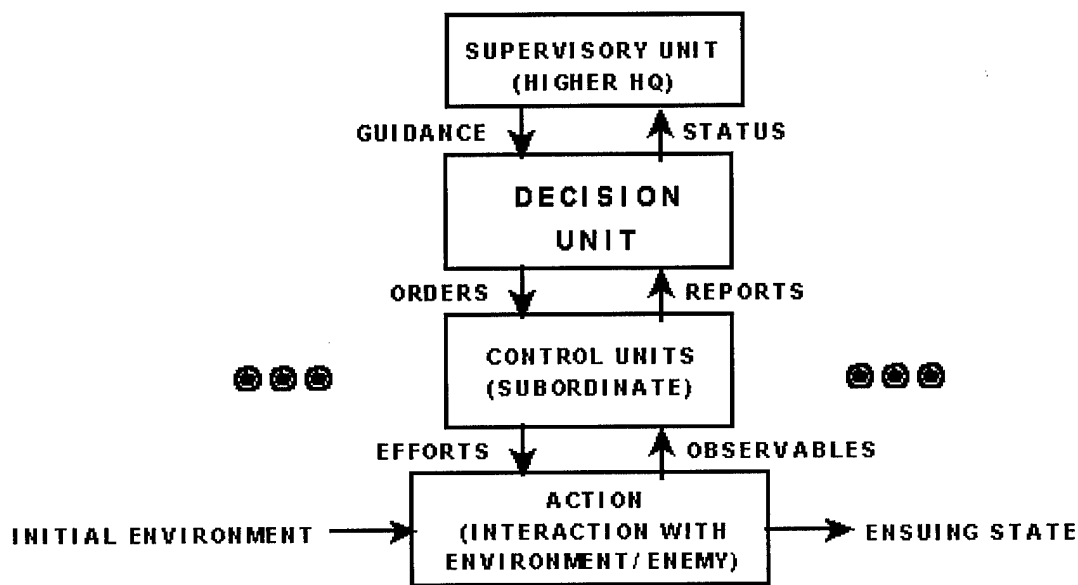
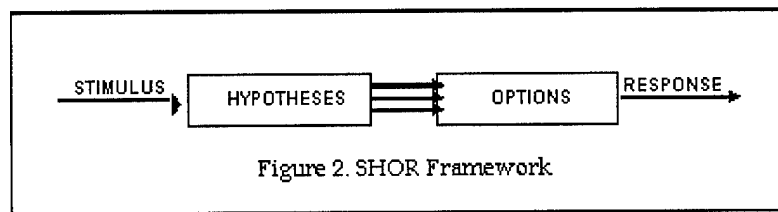


Figure 1. Hierarchical, Multilevel System

The model of the decision process is based upon the SHOR framework of Wohl, [WOHL81, WOHL83a, WOHL83b] shown in Figure 2. The acronym SHOR stands for stimulus, hypothesis, options, and response. A stimulus (S) is received thereby initiating the decision process. Based upon the stimulus and other information available at that time, various hypotheses (H) are generated concerning the actual situation or the actual state-of-the-world faced by the DMS. In consideration of the various possible situations it may be facing (the hypotheses), the DMS generates a set of options (O) or possible actions which it could direct to be taken. The effects or outcomes of the options are evaluated in view of the uncertain nature of the situation, and the DMS selects the appropriate action or response (R). The response then can be considered to interact with the external system generating additional stimuli which

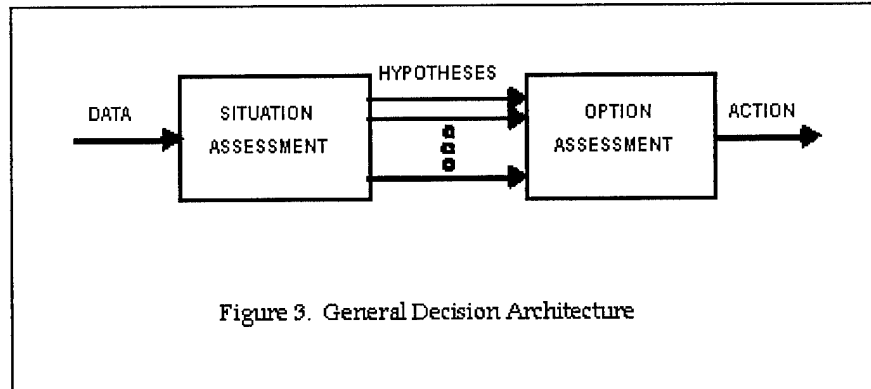
may lead to additional iterations of the process.



Similar frameworks have been developed by other investigators [LAWS79, RASM80, BOET82, DAVI82, ATHA83, LEVI83, RASM83]. While they are based on an assessment-response type of structure, they do include features germane to their investigations and do not provide the same architecture developed in this work.

THE GENERAL DECISION ARCHITECTURE

For this study the SHOR framework is adapted to produce the decision process architecture illustrated in Figure 3. In this view, data is received by the decision unit or DMS. The DMS conducts a situation assessment process (or sub-process) generating one or more hypotheses of the current situation, H_i . The DMS considers various options in the option assessment process, resulting in an action selected for implementation. For this current effort, no further interaction with the external environment is considered. Consideration will be concentrated on the situation assessment and option assessment subprocesses.



Additionally, the decision process is being modeled as an orderly sequential process. There are obvious opportunities for feedback, simultaneous processing, and jumping ahead in the real process. Consideration of these complications is deferred for subsequent studies.

In the following sections, we will further consider the Situation Assessment and Option Assessment subprocesses.

SITUATION ASSESSMENT

The situation assessment process considers newly acquired data in conjunction with currently available information and generates estimates of the current state of the environment or outside world [BEN-82]. Hypotheses, H_i , are developed concerning what is happening. Each of

these hypotheses, in turn, generates an interpretation of the state-of-the-world reflected in an estimated state vector x_i . The process also generates a measure of the likelihood of the hypothesis being true which is reflected as $P(H_i | Z)$. This is shown in the form of a conditional probability implying a Bayesian representation, but one may use other paradigms such as a possibility function [DOCK82], or Dempster-Shafer theory.

The concept of the situation assessment subprocess is shown in Figure 4. Incoming data are preprocessed by the Perception Processor to generate a vector Z which is suitable for further processing. Elements of the vector Z may be numerical, logical, or symbolic; whatever is appropriate to represent the input of the external world to the decision-making process. Operations such as calibration, transformation, aggregation, or correlation may occur within the preprocessor. It may use currently active hypotheses to establish the context for the processing. This processor may also include a process to assess the urgency with which the system should consider the situation, or to provide an alerting or bell-ringing function [PATT83, GREE82, WITU82, WOHL83b].

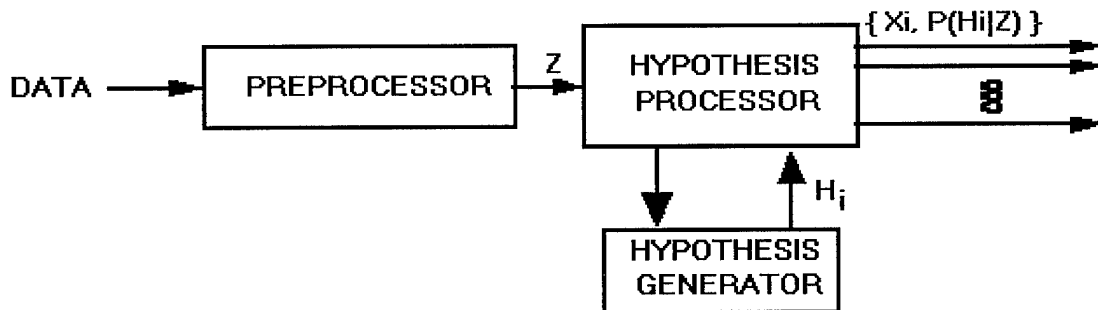


Figure 4. Situation Assessment Subprocess

The stimulus or initiating event for the decision process may be data-driven or concept-driven [WOHL83b]. The data-driven process occurs when specific received data are input to the system and are recognized as a stimuli. The concept-driven process occurs when a hypothesis or concept is presented for consideration, and then data or information is sought to evaluate the concept. Not much is definitely known about how hypotheses are generated. Some are already in memory and can be brought into active consideration [WOHL83b]. Others may be stimulated by specific features of the data Z . Still others may be created by mental processes leading to possible application of neural networks.

The Hypothesis Processor may be viewed as performing a pattern-matching process (see Figure 5). Each of the current active hypotheses in addition to forming an expected state vector X_i also form an expected view of what the incoming data should be. This expected value of Z under H_i or X_i can be represented as z_i . A pattern-matching process may then generate a set of variations and the decision-maker assess if the matches are satisfactory. If not, then the DMS generates additional hypotheses which are placed in the current hypothesis short-term memory until the DMS is satisfied with the set of hypotheses and state vectors under active consideration.

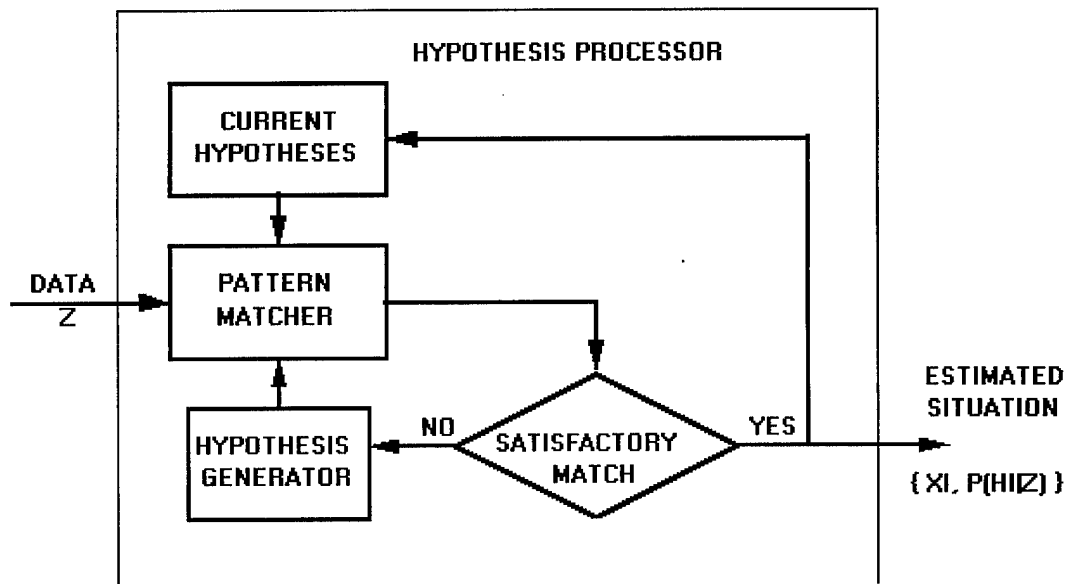


Figure 5. Hypothesis Processor

OPTION ASSESSMENT

The option assessment subprocess uses the hypotheses and the estimates of the situation facing the decision maker system as input, generates and evaluates alternative courses of action available to the DMS, and then selects a course of action for implementation.

This subprocess can be further decomposed into the subprocesses shown in Figure 6. Objectives and criteria are provided to the DMS from higher headquarters, from manuals, procedures, and doctrine, and from the DMS's experience and wisdom. It is assumed that some articulation of these is possible. It is also valuable to have some assessment of the relative importance or value of the objectives, goals, and criteria.

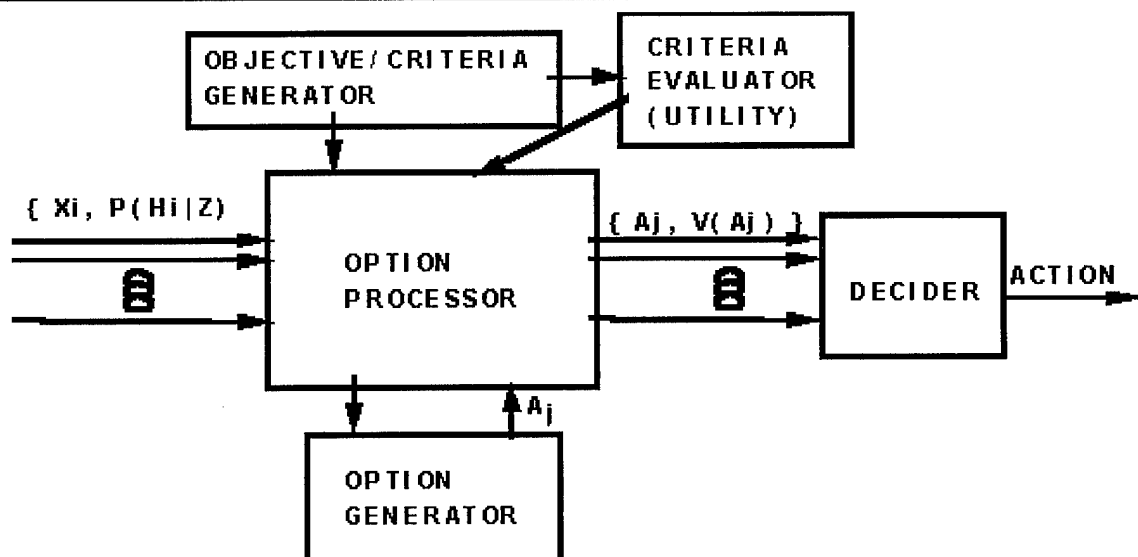


Figure 6, Option Assessment Subprocess

Options or courses of action open to the DMS are available within the memory structure or are provided by the option generator process. Outcomes of the actions under active consideration, a_j , will depend upon the true state-of-the-world, but can be estimated for each of the expected state vectors X_i . The degree to which the possible outcomes of each action meet the goals and objectives may be evaluated considering the relative value of the goals and objectives, and the degrees of belief, probability or possibility of the hypotheses. The nature of this evaluation process cannot be *a priori* defined for any decision-maker, but should allow choice. For this stage of the development it is assumed that some value $V(a_j)$ is available for each action a_j under consideration. This process can be viewed as a continuum of complexity from a simple single-criterion problem to a multi-objective, multi-criteria, nonlinear problem which could be analytically intractable. Whatever the nature of the evaluation process, it is assumed to be captured within the model.

At this point it may be useful to consider the modes of decision-making behavior developed by Rasmussen [HOLL83, RASM83, WOHL83b]. He postulated three decision-making modes: the skill-based mode, the rule-based mode, and the knowledge-based mode. The skill-based mode is characterized by strong habitual behavior. Actions are taken in response to stimuli with little or no conscious effort. This is the reflex action of a skilled operator or practitioner. Actions or decisions made in the rule-based mode are governed by procedures or doctrines. In this case, more effort is required than in the skill-based mode, to find the governing rule or procedure. Once found, however, the appropriate action is determined. Both the skill-based and rule-based modes depend on recognizing the triggering state-of-the-world with sufficient certitude given the urgency of the situation. The knowledge-based mode requires the DMS to extract appropriate information to deduce the appropriate action. This mode requires the full benefit of this decision model. The expanded Option Processor is shown in Figure 7.

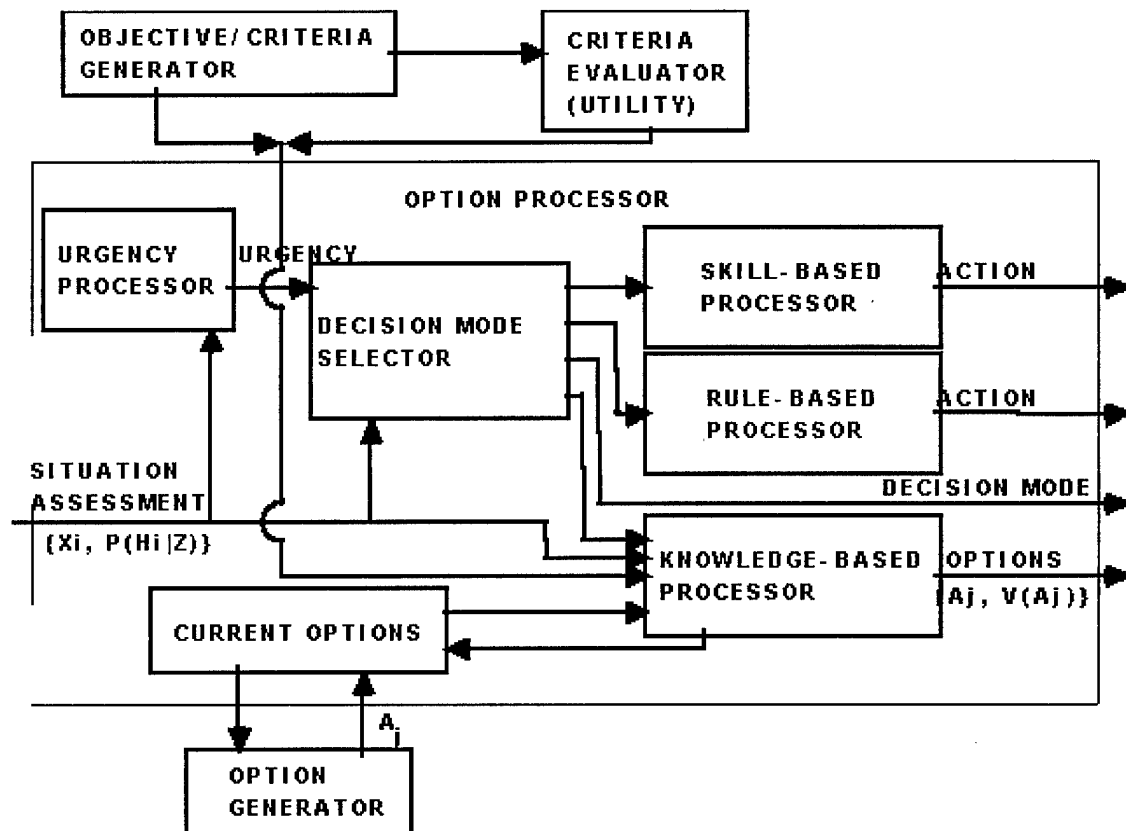


Figure 7. The Option Processor

We should realize that the terms used by Rasmussen do not carry the intention of meaning identical to those which the AI community uses. The blocks in Figure 7 called Rule-Based Processor and Knowledge-Based Processor could be implemented by any appropriate representational model. Not only should rule-based expert systems be considered, but neural networks, model-based systems, genetic algorithms, or others are appropriate candidates.

THE DECIDER

The processes to this point have been concerned with providing the DMS with the informational and procedural resources to make the decision. The actual selection of an action or the decision is made in the Decider process. This process is represented in Figure 8.

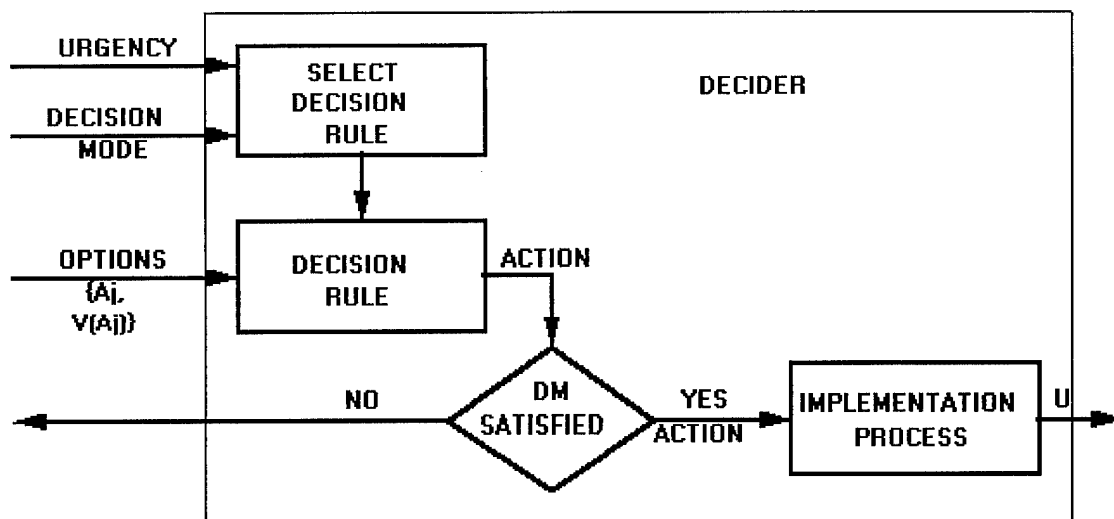


Figure 8. The Decider

In his excellent review article, Sage describes various models of decision-making behavior and effects of stress and complexity [SAGE81]. For this model, it is assumed that as a result of some interaction, a choice is made of decision rule to use, and then the decision rule is applied to select the appropriate action for implementation. In the case of skill-based or rule-based behavior, that action will be already determined. In the case of knowledge-based behavior, one may still select a satisfying decision rule [SIMO81] (also called bounded rationality), a rational actor model, a garbage can model, or some model for making the action selection decision.

Even once this is done, the DMS has one last chance to review the situation and decide if he is really willing to make this decision. If not, the process can cycle back to an earlier stage and go through another iteration. If the DMS is satisfied, it can proceed with implementation. One can go through the decision process and get so caught up with it that some perspective may be lost. This last decision acts as a final check, and, by personal experience, often results in NO result.

We have shown an Urgency variable with input into the Select Decision Mode process. Based on the current hypothesized situation assessment, the Urgency variable will assess a time by which an action should be selected and implemented. The three option processors, the Skill-Based, Rule-Based, and Knowledge-Based Processors will all operate in parallel through blackboard structures so that a "best" action may be selected by the time that it is required.

IMPLEMENTATION

The implementation process represents converting the selected action into an operable instruction u which will cause some interaction with the external environment. It could represent the issuance of an order or communication. In a complete modeling sense, there may be some further planning necessary to translate the selected action into an executable instruction.

EXTERNAL ENVIRONMENT

Once the decision is made and appropriate executable instructions issued, there is some

interaction with the external environment. The results of that interaction may be sensed and that data input to the process to begin another decision. In that sense the process may be a control process. Of course, the process may be open-ended with no subsequent processing.

At several stages in this model, decisions were made as part of the decision process. Since the model, as developed, is general in scope, it can be nested so that each of the individual decisions could be represented in terms of this model.

SECTION IV - CONCLUSIONS

Department of Defense research agencies, ARPA, NRL, AFOSR, and ARI, have begun to examine decision-making and ways AI can be used to assist in the decision-making process. This work developed a general architecture of a decision process using Wohl's SHOR paradigm (Stimulus, Hypothesis, Options, Response) and carrying the decomposition to more detail than previously done, but consistent with the thrust of the DOD work. The possible use of AI techniques to support the decision processes in the model structure are left to others to discuss. AI provides the promise of tools to help the military decision-maker cope with the increasing mass of data and information and of the decreasing response times of the future battlefield. It may also help to provide the decision-makers the capacity for making better decisions. It will, however, provide capability for autonomous agents for use in simulation systems.

The architecture was constructed within the SHOR paradigm. The process was decomposed into situation assessment and option assessment phases. The situation assessment phase is composed of preprocessor, hypothesis generation, and hypothesis assessment subprocesses. The option assessment phase is composed of criteria, option generation, option processing, and decider subprocesses. The context for the overall decision process is a command and control structure, but the architecture applies to any decision-making situation.

REFERENCES:

ATHA83 Michael Athans, "System Theoretic Challenges in Military C₃ Systems," Naval Research Reviews, Vol. XXXV, No. 2, 1983, pp. 18-28.

AZAR86 Jerome Azarewicz, "Plan Recognition for Airborne Tactical Decision Making," AAAI-86, pp. 805-811.

BAIL82 Robert W. Bailey, Human Performance Engineering, Prentice-Hall, 1982.

BARN81 M.J. Barnes, Human Information Processing Guidelines for Decision-Aiding Displays, Naval Weapons Center, China Lake, CA, NWC, Technical Memorandum 4605, December 1981. AD-A124858. N83-02023105.

BARR92 John Barry and Roger Charles, Newsweek, July 13, 1992, pp. 29-39.

BASA81 Tamer Basar and Jose B. Cruz, Jr., Concepts and Methods in Multi-Person Coordination and Control, University of Illinois at Urbana-Champaign, Technical Report UILU-ENG-81-2251, R-920(Dc-49), October 1981, 77 pages. AD-A124386.

BEN-82 Moshe Ben-Bassat and Amos Freedy, "Knowledge Requirements and Management in Expert Decision Support Systems for (Military) Situation Assessment," IEEE Trans. Syst., Man, Cybern., Vol. SMC-12, No. 4, pp.479-490, July/ August 1982.

BOET82 Kevin L. Boettcher and Alexander H. Levis, "Modeling the Interacting Decision Maker with Bounded Rationality," IEEE Trans. Syst., Man, Cybern., Vol. SMC-12, No. 3, pp. 334-344, May/June 1982.

BUON82 Frederick B. Buoni, Decision Support Systems, Appendix L of Computer Science: Key to a Space Program Renaissance, Final Report of the 1981 NASA/ ASEE Summer Study on the Use of Computer Science and Technology in NASA, University of Maryland Technical Report 1168, January 15, 1982.

BUON83 Frederick B. Buoni, A Process Architecture for Computer-Based Decision Support, Florida Institute of Technology working paper, August 1983.

CACC92 Pietro Carlo Cacciabue, Francoise Decortis, Bartolome Drozdowicz, Michel Masson, and Jean-Pierre Nordvik, "COSIMO: A Cognitive Simulation Model of Human Decision Making and Behaviour in Accident Management of Complex Plants," IEEE Trans. Syst., Man, Cybern., Vol. 22, No. 5, pp. 1058-1074, September/October 1992.

CHAP93 Alan R. Chappell, Knowledge-Based Reasoning in the Paladin Tactical Decision Generation System, NASA Contractor Report 4507, Lockheed Engineering and Sciences Company, 1993.

CHU 82 Yee-yeen Chu and Amos Freedy, "Computer-Aided Information Handling in Supervisory Control of Airborne Systems," IEEE 1982 Proc. Conf. Cybern. Soc., pp. 425-429, 1982.

COMP82 Michael A. Companion and Gregory M. Corso, "Task Taxonomies: A General Review and Evaluation," Int. J. Man-Machine Studies, Vol. 17, No. 8, pp. 459-472, 1982.

DAVI82 M. Dianne Davis, "Cybernetics in the Submarine Society: The Human Use of High Technology," 1982 Proc. Int. Conf. Cybern. and Soc., pp. 223-227, 1982.

DEGR82 Edward DeGregorio, Ann Silva, and David Brazil, "Applied Artificial Intelligence in the Submarine Combat Control Environment," 1982 Proc. Int. Conf. Cybern. Soc., pp. 506-510, 1982.

DOCK82 John T. Dockery, "Fuzzy Design of Military Information Systems," Int. J. Man-Machine Studies, Vol. 16, pp. 1-38, 1982.

FITT80 Mike Fitter and Max Sime, "Creating Responsive Computers: Responsibility and Shared Decision-Making," in Smith and Green, Ed's, Human Interaction With Computers, Academic Press, 1980, pp. 39-66.

FUNK82 Ken Funk, "Symbolic Structures and the Human Operator," IEEE 1982 Proceedings of

the International Conference on Cybernetics and Society, October 28-30, 1982, pp. 156-160.

GREE86 Joel S. Greenstein and Mark E. Revesman, "Development and Validation of a Mathematical Model of Human Decisionmaking for Human-Computer Communication," IEEE Trans. Syst., Man, Cybern., Vol. SMC-16, No. 1, pp. 148-154, 1986.

GREE82 Joel S. Greenstein, and William B. Rouse, A Model of Human Decision-making in Multiple Process Monitoring Situations, IEEE Trans. Syst., Man, Cybern., Vol. SMC-12, No. 2, pp. 182-193, 1982.

HAMM82 John M. Hammer and William B. Rouse, "Design of an Intelligent Computer-Aided Cockpit," 1982 Int. Proc. Conf. Cybern. Soc., pp. 449-453.

HOLL83 Erik Hollnagel, "What We Do Not Know About Man-Machine Systems," Int. J. Man-Machine Studies, Vol. 18, No. 2, pp. 135-143, 1983.

INGR92 Francois F. Ingrand, Michael P. Georgeff and Anand S. Rao, "An Architecture for Real-Time Reasoning and System Control," IEEE Expert, pp. 34-44, December 1992.

JANI77 Irving L. Janis and Leo Mann, Decision Making: A Psychological Analysis of Conflict, Choice and Commitment, Free Press (MacMillan) 1977.

JOHN82 Scott Kevin Johnson, An Investigation of a Land Combat Tactical Commander's Decision-Making Process, Naval Postgraduate School Thesis, October 1982. AD A124990.

LAWS79 Joel S. Lawson, Jr., "Naval Tactical C3 Architecture 1985-1995," Signal, Vol. 33, No. 10, August 1979, pp. 71-76.

LEED79 D. K. Leedom, "Representing Human Thought and Response in Military Conflict Simulation Models," in Symp. on Modelling and Simulation of Avionics Syst. and Command, Contr. and Com. Syst., AGARD (NATO) Conf. Proc., No. 268, Nat. Tech. Inform. Ser., Oct. 15-19, 1979.

LEVI77 Robert A. Levit, Berton J. Heaton, David G. Alden, "Development and Application of Decision Aids for Tactical Control of Battlefield Operations: Decision Support in a Simulated Tactical Operations System (SIMTOS)," Honeywell Systems and Research Center, Minneapolis, MN, ARI Technical Report TR-77-A13, December 1977, AD A121413.

LEVI82 Alexander H. Levis, Elizabeth R. Ducot, Michael Athans, Distributed Decision and Communication Problems in Tactical USAF Command and Control, Laboratory for Information and Decision Systems (MIT), July 30, 1982. Technical Report LIDS-IR-1226, AFOSR-TR-82-0952. AD A121413.

LEVI83 Alexander H. Levis and Kevin L. Boettcher, "Decisionmaking Organizations with Acyclical Information Structures," IEEE Trans. Syst., Man, Cybern., Vol. SMC-13, No. 3, May/June 1983, pp. 384-391.

MADN82 Azad M. Madni, Michael G. Samet, and Amos Freedy, "A Trainable On-Line Model

of the Human Operator in Information Acquisition Tasks," IEEE Trans. Syst., Man, Cybern., Vol. SMC-12, No. 4, pp. 504-511, July/ August 1982.

MESA70 M. D. Mesarovic, D. Macko, and Y. Takahara, Theory of Hierarchical, Multilevel Systems, Academic Press, 1970.

NICK77 R. S. Nickerson, M. J. Adams, R. W. Pew, J. A. Swets, S. A. Fidell, C. E. Feehrer, D. B. Yntema, D. M. Green. The C3 System User Vol. I: A Review of Research on Human Performance as it Relates to the Design and Operation of Command, Control and Communication Systems. Bolt Beranek and Newman, Inc. Technical Report 3459, Feb. 1977, AD-A126633.

PATT83 Krishna R. Pattipati, David L. Kleinman, and Arye R. Ephraim, "A Dynamic Decision Model of Human Task Selection Performance," IEEE Trans. Syst., Man, Cybern., Vol. SMC-13, No. 3, pp. 145-166, March/ April 1983.

POST90 Stephen Post and Andrew P. Sage, "An Overview of Automated Reasoning," IEEE Trans. Syst., Man, Cybern., Vol. 20, No. 1, pp. 202-224, January/ February 1990.

RASM80 Jens Rasmussen, "The Human as a Systems Component," in Smith and Greed (Ed's), Human Interaction with Computers, Academic Press, 1980.

RASM83 Jens Rasmussen, "Skills, Rules, and Knowledge: Signals, Signs, and Symbols, and Other Distinctions in Human Performance Models," IEEE Trans. Syst., Man, Cybern., Vol. SMC-13, No. 3, pp. 257-266.

REN 95 Jie Ren and Thomas B. Sheridan, "An Adaptive Decision Aid for Real Environments," IEEE Trans. Syst., Man, Cybern., Vol. 25, No. 10, pp. 1384-1391, October 1995.

RIEG82 Christine A. Rieger and Joel S. Greenstein, "The Allocation of Tasks Between the Human and Computer in Automated Systems," in IEEE 1982 Proceedings of the International Conference on Cybernetics and Society, October 28-30, 1982, pp. 204-208.

ROUS81 William B. Rouse, "Human-Computer Interaction in the Control of Dynamic Systems," Computing Surveys, Vol. 13, No. 1, pp. 71-99, March 1981.

SAGE81 Andrew P. Sage, "Behavioral and Organizational Considerations in the Design of Information Systems and Processes for Planning and Decision Support," IEEE Trans. Syst., Man, Cybern., Vol. SMC-11, No. 9, pp. 640-678, September 1981.

SAGE82 Andrew P. Sage and Adolfo Lagomasino, "Knowledge Representation and Interpretation in Decision Support Systems," IEEE 1982 Proc. Conf. Cybern. Soc., pp. 658-662, 1982.

SIMO81 Herbert A. Simon, The Sciences of the Artificial, Second Edition, MIT Press, 1981.

VONN44 John Von Neumann and Oskar Morgenstern, The Theory of Games and Economic Behavior, Princeton, 1944.

VONW86 Detlof Von Winterfeldt and Ward Edwards, Decision Analysis and Behavioral Research, Cambridge University Press, 1986.

VRAN92 Sanja Vranes, Mario Lucin, Mladen Stanojevic, Violeta Stevanovic and Pero Subasic, "Blackboard Metaphor in Tactical Decision Making," European Journal of Operational Research, Vol. 61, pp. 86-97, 1992.

WITU80 Gary Witus, Robert W. Blum, Mark Graulich, Description of the Tactical Operations System Information Flow Model, Vector Research, Inc. Report VRI-ARI-3-FR79-3, 30 November 1979, ARI Research Note 80-13, AD-A092206, May 1980.

WITU82 Gary Witus, Robert W. Blum, Information Flow Management for Distributed Tactical Command Control Systems, IEEE Trans. Syst., Man, Cybern., Vol. SMC-12, No. 4, pp. 512-518, 1982.

WOHL81 Joseph G. Wohl, "Force Management Decision Requirements for Air Force Tactical Command and Control," IEEE Trans. Syst., Man, Cybern., Vol. SMC-11, No. 9, pp. 618-639, September 1981.

WOHL83a Joseph G. Wohl, E. E. Entin, M. G. Alexandridis, J. S. Eterno, Toward a Unified Approach to Combat System Analysis, Alphatech, Inc. Technical Report TR-151, January 1983, ADA124570.

WOHL83b Joseph G. Wohl, E. E. Entin, J. S. Eterno, Modeling Human Decision Processes in Command and Control, Alphatech, Inc. Technical Report TR-137, January 1983, AD A125218.

ZELE81 Milan Zeleny, Multiple Criteria Decision Making, McGraw-Hill, 1981.

ACKNOWLEDGMENTS

This work was conducted while the author was a NASA/ASEE Summer Faculty Fellow assigned to the Jet Propulsion Laboratory (JPL). The ten-week term of the fellowship was spent with the Airland Battle Advanced Technology (ABAT) program in association with Dr. Victor J. Anselmo.

The author may be contacted at the following address:

Dr. Frederick B. Buoni
College of Engineering
Florida Institute of Technology
Melbourne, FL 32901

(407) 768-8000, ext. 7390
E-mail: buoni@zach.fit.edu

[Return to the Top of this Section](#)

BG2 *PROBLEM DEFINITION:*

A Functional Description of a Command Agent

Howard Mall, M.S.C.S.; Rick McKenzie, Ph.D.; and
Jenifer McCormack, Ph.D.

Science Applications International Corporation
3045 Technology Parkway
Orlando, Florida 32826-3299
(407) 282-6700
Howard_Mall@cpqm.saic.com
Rick_McKenzie@cpqm.saic.com

OVERVIEW:

A **command agent** is an intelligent agent that will act as a surrogate for a human commander in a military simulation. To develop command agents that are designed for change within the evolving simulation domain, requires an architecture which will allow the heterogeneous application of decision-making strategies, the management of myriad interfaces to simulation services and knowledge sources, and the proper expression of command behavior. The approach suggested is to develop an integrated collection of sub-agent **advisors** which is arbitrated and controlled by an overwatching **command agent**. This architecture can be applied to many different domains and be used to model different approaches to command decision making.

THE COMMAND DECISION-MAKING DOMAIN

Army commanders, by their very definition, make decisions. They make decisions on how to act in a given situation. They then formulate orders that are given to their subordinates who in turn can make decisions about how to act and give orders to their subordinates. This is the hierarchical structure of the military (see Fig. 1, BG 1) that allows a commander to control a large number of men and materiel without being overwhelmed with information. It also allows specialization in which one individual can gain deep knowledge about a specific aspect of the military domain, such as intelligence, logistics, artillery, etc.

Computer Generated Forces (CGF) must demonstrate realistic and valid behaviors of military entities in order to provide an effective training environment or predictive capability. In order to exhibit advanced coordination and cooperation of military units (which are found on a real battlefield) simulations turn toward modeling the commander of those forces to provides the Command, Control, and Communication (C³) required.

The implementation of a model utilized to simulate the command decision making process will hereafter be referred to as a **command agent**. "An agent is a surrogate for a person or process that fulfills a stated need or action. []" A command agent provides a framework for the aggregation and abstraction of behaviors of various military echelons. It also allows the C³ process to be closely patterned after the one utilized in the real world. The SHOR framework (see Figure 2, BG 1) indicates a proven model for initiating and executing decision making processes.

Figure 3 (BG 1) shows the SHOR framework as applied within CGF. Decision making is divided into two phases: one for situation assessment followed by the assessment of options. Data received from the simulation environment stimulates the system to carry out assessments of situations and generate hypotheses. These hypotheses represent assessments of the situation. The option assessment phase uses these hypotheses to generate options (in military parlance: courses of action). Some decision criteria are generated and used to select an option for execution.

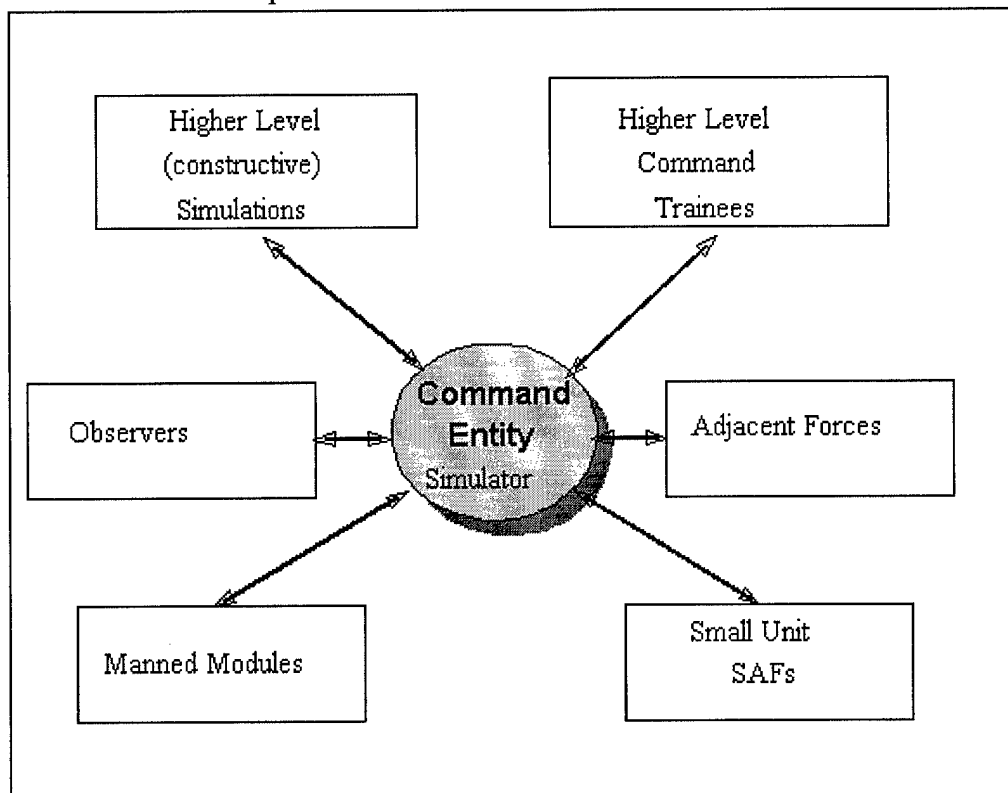


Figure 1: Command Agent Interfaces

Figure 1: Command Agent Interfaces

IMPLEMENTATION IMPLICATIONS

The general description of the command decision making domain described above and in [task1] captures the process appropriately. However, for military simulations, there are many different implementation issues that must be addressed. In a military simulation there are a variety of interfaces that a command agent must be capable of utilizing. Figure 1 shows a representative collection of systems that are involved in a modern Distributed Interactive Simulation (DIS) environment.

The growing trend in the military simulation community is for greater degrees of system interoperability, computer resource distribution, and entity and aggregate behavioral complexity. Simulations of today must find some optimal solution to balance these contentious goals. A command agent must also be designed and implemented for these varying degrees of integration.

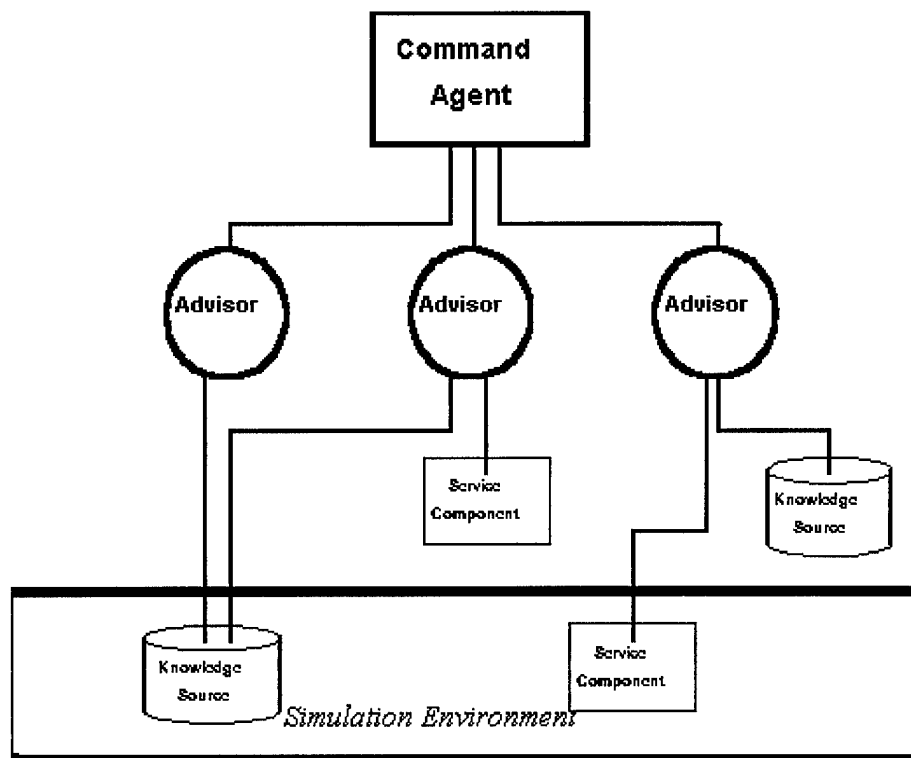


Figure 2: Command Agent Architecture

Complex decision making systems are typically a hybrid of artificial intelligence techniques. They deal with diverse types of information and knowledge that can be used in different contexts. A result of the work reported in [] was the need for a command agent to deal with environments rich with variegated information. The design of a command agent should reflect the need for heterogeneous integration both internally and externally. Figure 2 shows the generic layered approach suggested for constructing a command agent. A detailed application of this approach can be found in [] where "many different reasoning processes, societies of agents, are integrated in order to realize a software assistant capable of performing a broad range of tasks."

This design allows the implementation of different artificial intelligence techniques to be applied to their appropriate categories of problems. It also provides facilities to abstract the interfaces to external knowledge sources and simulations. The key to this approach is the implementation of advisors which are sub-agents constructed to monitor, interpret, and reason about information flowing into the command agent from various sources (i.e. terrain databases, sensor objects within the simulation, reports from subordinate units, etc...) The process modeled within the command agent can remain stable while new interfaces to various simulation components can be managed and translated by the advisors.

To correlate with the SHOR framework (Figure 3, BG1), stimulus is communicated from the simulation environment through the command agent's advisors. The advisors handle the API's to heterogeneous components within the simulation, which provide information and accept direction. Advisors receive information, either passively or through query, and synthesize the information into situation hypotheses. These hypotheses are translated into the command agent's internal representation. The command agent acts to arbitrate advisor input and to control the services and interfaces that the advisors provide. The command agent has knowledge of how to utilize the analyses of its advisors to make decisions.

This implementation corresponds to a model that is very close to the way modern command decision-making is done. The commander receives knowledge through his aides or advisors that analyze and interpret with specialized expert knowledge. The information is then presented to the commander as icons on a map or some other abstract representation that saves the commander from processing large amounts of irrelevant information. The commander receives only information appropriate to his decision-making process.

The command agent structure not only coincides with real-world models but it also provides many software engineering advantages as well. In order to add new services or other capabilities to the command agent, the developer need only extend an existing advisor or add a new one. This extension would not significantly impact the other components due to the layer of abstraction defined between the advisors and the command agent. This advantage applies, also, to advisors; whereby, they can supply the same services to the command agent but utilize new sources of information by switching APIs.

The command agent design provides a flexible architecture that allows modular functionality, extensibility, and interoperability. These requirements are necessary to perform within the diverse and plentiful simulation federations being fielded today. The implementation of a command agent can take many forms and depends on many factors derived from the environment in which command agent is intended to operate.

COMMAND AGENT CONSTRUCTION:

The command agent is composed of a set of sub-agent advisors that specialize in generating hypotheses about the situation environment and the decision making component that supervises the advisors and makes decisions based on the information they produce.

COMMAND AGENT:

The command agent's main role is to control the operation of its advisors and make decisions based on the results of these advisors. The command agent is made up of the following components:

- Internal Representations
- Inference Engine
- Command Behavior Model
- Agent Arbitration and Control Knowledge
- General Sub-Agent Interface

The command agent is an intelligent agent. "An intelligent agent is composed of the represented knowledge that drives its functions. This knowledge is described in the form of goal(s) and process(es) that lay out both the objectives for the agent and manner in which it will perform its tasks." []

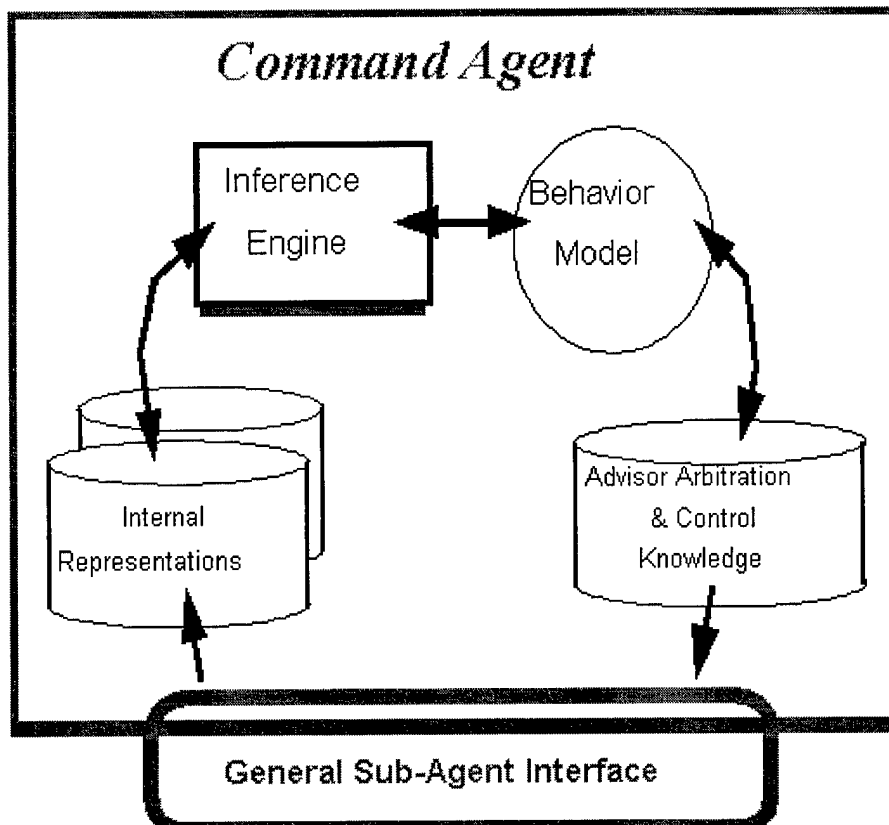


FIGURE 3: COMMAND AGENT INTERNALS

Internal Representations and Inference Engine

The Command Agent retains internal representations that represent its perceptions of the battlefield environment. These internal representations are updated and maintained by the flow of information from the Command Agent's Advisor sub-agents. Within [ii] the authors chose to use two representations to store battlefield knowledge: a semantic network and a tessellated terrain map. The choice of internal structures can depend on the level of abstraction required in order to accomplish the command decision-making process efficiently and effectively.

Frame-based systems appear to be the best representation for the internal knowledge of the command agent. This technique follows an object-oriented strategy to depict the concepts and relationships that the command agent must handle. It provides a degree of flexibility and resolution for organizing battlefield information, and it requires a simple inference engine processing control rules to utilize the information. This corresponds to the human commander who does not cognitively handle the vast amounts of data that flows around a battlefield. Instead, he relies on trusted advisors to fuse, filter, and extrapolate important information from this flow of data. Decisions are made based on these abstract views.

The frame-based system is recommended because it provides a general structured approach to modeling battlefield knowledge that applies at different echelon levels and different levels of complexity; it can be as simple or complex as the developer wants to make it. It is, as pointed out above, cognitively similar to the way human commanders conduct their decision-making process. Inference engines derived from expert system shells are also readily available, and there is prevalent legacy experience in the engineering of these kinds of systems for decision-making. Other features such as decision explanation, case-based reasoning, and machine learning can be easily value-added to a frame-based system.

Behavior Model

The Command Agent has a model of behavior which it uses to control its operation. These can be represented as constraints on its decision-making capability. These are usually defined by the higher-level controller of the command agent: a human operator or higher-echelon command agent.

These constraints can be represented in a variety of ways, but the easiest to implement would be as meta-rules within the inference engine. These rules would be differentiated from the static doctrinal rules by their dynamism. The implementation would be based on the capabilities of the expert system shell, but could include either newly written (marked) rules or existing rules that depend on defined data within the command agent's representation.

Advisor Arbitration and Control Knowledge

Arbitration and Control Knowledge defined for the command agent's advisors is of extreme importance to the design of the command agent. This component is where the command agent's knowledge of its advisors is stored. Utilizing the arbitration and control knowledge, the command agent is able to handle inconsistencies that may occur when two or more agent's produce conflicting hypotheses about the environment. In spite of the very careful compartmentalization of each agent's role in the simulation domain, agents can produce information that does not correlate either directly or by inference from that of other agents, especially as the system's components grow in complexity.

Arbitration and Control Knowledge also aid the command agent in directing the operation of its advisor sub-agents. This stores knowledge for the command agent to direct when and how the advisors should be utilized for efficiency. The behavior model greatly affects the utilization of this meta-knowledge for directing the decision making process. This knowledge could be stored as a procedural construct (e.g. a finite-state machine) in which advisor management strategies would be based on the context or node of the command agent. This provides a straight-forward and simple approach to handling advisor control and arbitration.

General Sub-Agent Interface

The General Sub-Agent Interface provides a layer of abstraction between the advisors and the command agent. The agent can call on the services encapsulated by the advisors through the sub-agent interface. The advisors also provide information to the command entity through this interface. This layer of abstraction allows the different pieces of the command agent to be functionally distinct but their combined behavior to act homogeneously.

A layered approach allows for independent and concurrent development of the interior processes of the advisors and the command agent. The different advisors and command agent can be tested without the other components being complete by simulating (stubbing) their behavior. The interface through which these pieces communicate must be maintained as well, but it provides flexibility required for the evolution of the design and implementation.

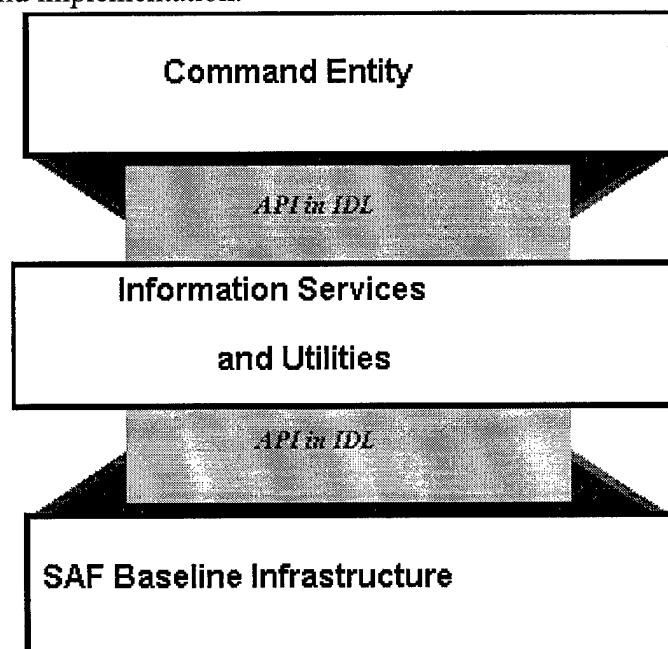


Figure 4: CFOR Reference Model

This layered approach can be seen in the technical reference model and implementation of the CFOR system [1]. Figure 4 shows the high-level layered model used as a basis for the CFOR design. There is a defined Application Programmers Interface (API), composed in an Interface Design Language (IDL), between the command entity application and decision processes, the information services and utilities, and the SAF baseline infrastructure (which in this implementation contains ModSAF and the CCSIL network protocols). This approach is extended in that advisor agents are the shepherds of a myriad number of interfaces and services and utilities that will support the command decision-making process.

ADVISORS

Advisors are sub-agents of the command agent that sift the extreme amount of battlefield data in a simulation and synthesize the data into information useful to the command agent's decision-making process. The agent's compartmentalized operation allows its internal process to be implemented using heterogeneous methodologies that are best-suited for that particular agent's specialized sub-domain of the battlefield environment. "In general, multiagent systems are computational systems in which several semi-autonomous agents interact or work together to perform some set of tasks or satisfy some set of goals." [1]

As can be seen from Figure 5 (next page), the advisor sub-agent consists of the following pieces:

- Interfaces
- Command Agent Directives
- Domain Knowledge
- The Reasoning Mechanism
- Provided Services.

The advisor controls the interfaces to components within the simulation that provide data and services. This can be terrain algorithms, C³ equipment, or vehicle primitives, etc. The advisor also receives and stores command agent directives that controls the advisor's operation. This can take the form of queries of data or services the command agent requests, constraints on the services of the advisor sub-agent, or tasks that the command agent has set for the advisor.

The domain knowledge element denotes the advisor's special capabilities regarding the context of its specialty. Domain knowledge can take many forms, some that encodes knowledge implicitly. Domain knowledge can be captured within rules, a training of a neural net, or within the cases of a case-base. "Agents could represent the same knowledge differently to optimize their particular use of it, or agents could obtain knowledge from different sources..." [1]

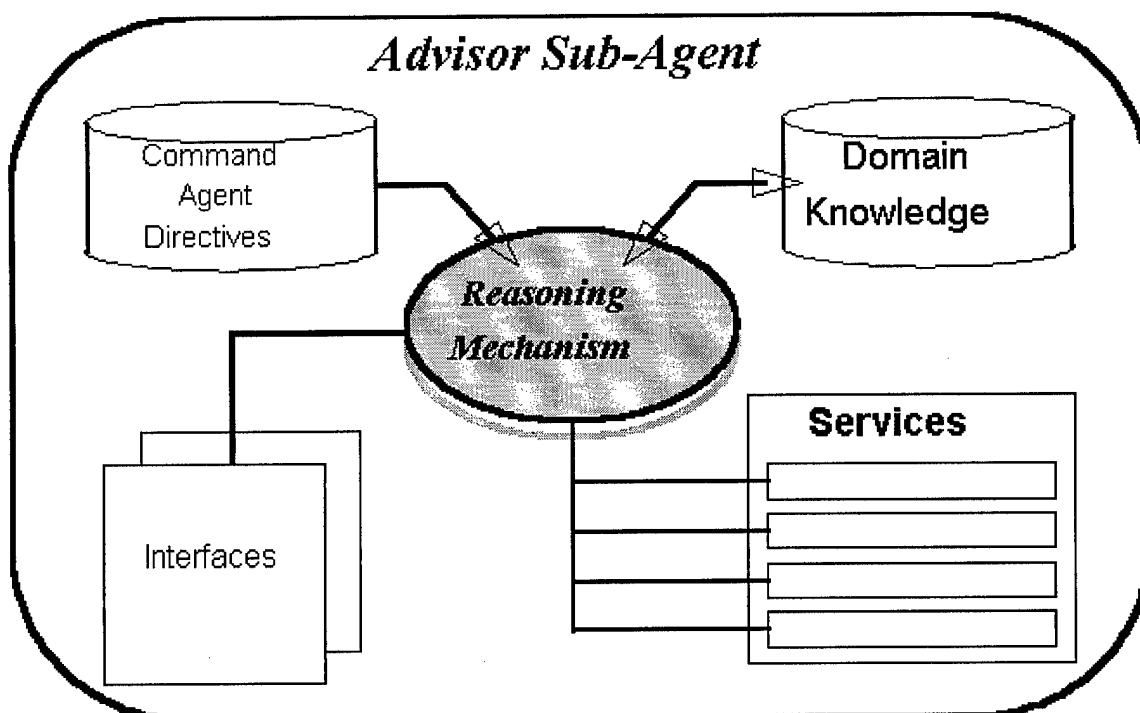


Figure 5: Advisor Sub-Agent Model

The reasoning mechanism of the advisor describes the operations that are performed utilizing the encoded domain knowledge. The reasoning mechanism could be a forward- or backward-chaining inference engine or neural nets in the form of the training and retrieval algorithms with which the net has been constructed. Case-based systems would employ key-matching and case adaptation as its reasoning mechanism.

Finally, the services that the advisor may be defined as part of its interface to the command agent. Data-driven techniques could be employed to allow the advisors to "register" its services with the command agent describing its capabilities and purpose. If the advisors conform to the protocol of the General Advisor Interface then this could facilitate an automated way of integrating altered or new advisors to the command agent's repertoire. This concept is a smaller-scale implementation of the ideas set down in the High-Level Architecture (HLA) specification.

COMMAND AGENT IMPLEMENTATION

Employing the layered concepts above allows the implementation of the command agent to take many forms while retaining the same decision-making process described in the SHOR framework. The flexibility of the design allows the implementation of command decision making agents to take any form which is appropriate to the problem domain. The developer is constructing an ontology of SHOR components. The following sections describe examples of command agents that may be constructed for different echelon levels and different strategies of conducting decision-making.

JUDGEMENTAL METT-T COMMAND AGENT

Command agents must be capable of rapid decision-making in constantly changing domains. Knowledge acquisition to apply to automated decision-making is of great difficulty because of the lack of canonical literature. To aid in this process the command agent can be modeled around doctrinal epistemology.

Situational awareness is paramount to the a battlefield commander. Army doctrine captured from Subject Matter Experts (SMEs) and training literature promotes the acronym METT-T to describe a process for an army commander to analyze the battlefield environment. METT-T stands for Mission, Enemy, Terrain, Troops, and Time, and it is utilized at all levels of the U.S. Army command structure.

As can be seen in Figure 6 (next page), the construction of the Command Agent's advisors corresponds with these subjective areas. As described above, the advisors act as the command agent's proxies and filters to the information sources that exist inside and outside of the simulation environment.

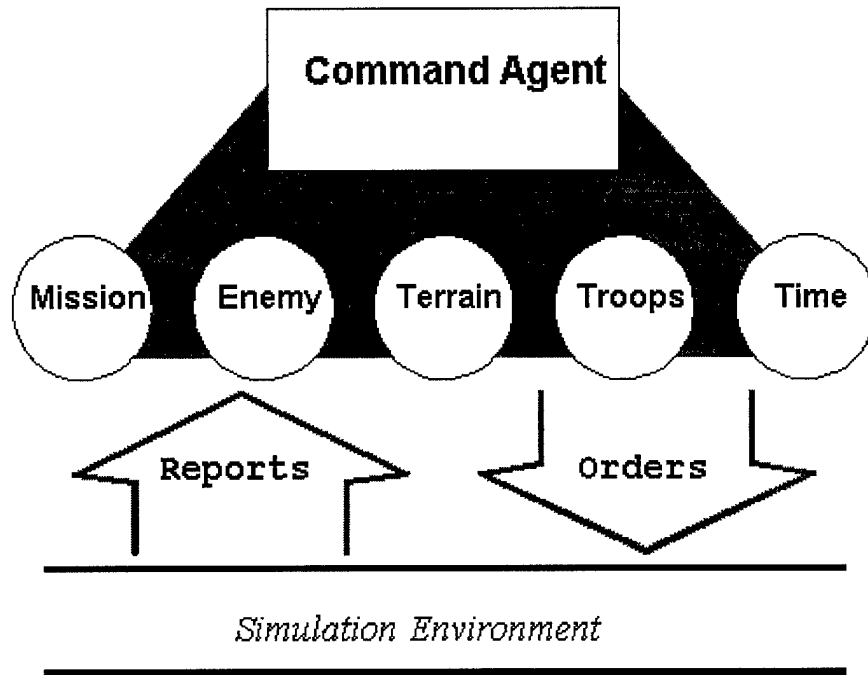


Figure 6: Command Agent with METT-T Advisors

For example, the Terrain Advisor controls the interface to the terrain database of the battlefield. The Terrain Advisor has rational methods of analyzing this data and deriving knowledge that is useful to the Company Team Command Agent. The other Advisors have similar interfaces to other information sources necessary to conduct their specialized purposes. The strength of this approach is that various AI techniques may be encapsulated within these Advisors providing a flexible, extensible framework in which to construct hybrid systems. On top of this design is the Command Agent that acts as arbiter and director of the outputs of each advisor's analyses.

A detailed description of this model as applied to a Company Team Command Agent utilizing METT-T advisor sub-agents can be found in [ii]. Table 1 shows some possible AI techniques that may be applied to the different areas of METT-T.

Table 1: Techniques applied to METT-T

Mission	<i>Objectives-based Planning</i>
Enemy	<i>Case-Based Reasoning (CBR)</i>
Terrain	<i>Neural Nets</i>
Troops	<i>Rule-based Systems</i>
Time	<i>Temporal Reasoning</i>

BOS COMMAND AGENT

The Battlefield Operating System (BOS) is a large connection of computers and intercommunication components that is used to disseminate and fuse large amounts of battlefield information. The BOS is divided into several different areas of command and control that is under a high-level echelon commander's purview. The example, shown in Figure 7, (next page) is meant to illustrate how a domain model of decision-making can be constructed for command agents that differs from the one presented above.

The same SHOR framework (see above) concept is employed, distributed through the different advisors. Data flows from the simulation environment and spawns one or a concurrent number of stimuli within the advisor environ. The advisors generate hypotheses that are incorporated into the BOS Command Agent's internal representations. The BOS Command Agent determines how to use these hypotheses. It can call on advisor services in order to make a decision or continue to collect information.

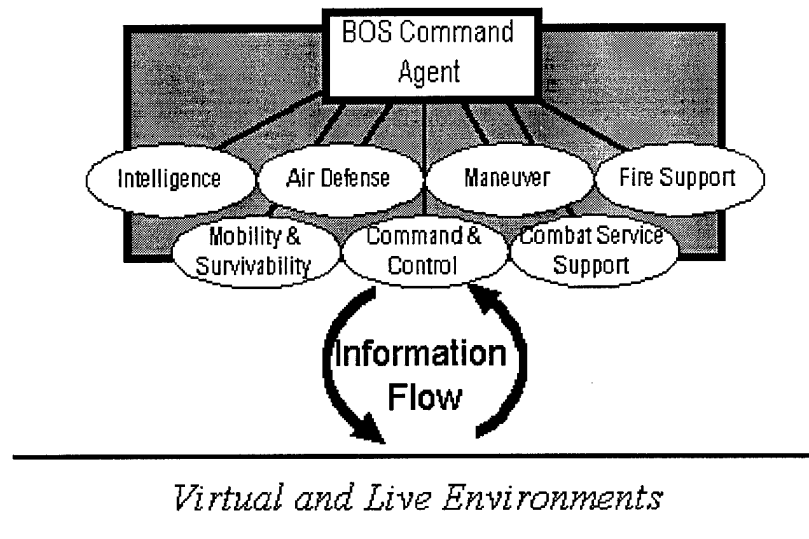


Figure 7: BOS Command Agent with Critical Combat Function Advisors

The advisors specialize in critical combat functions necessary to particular areas of a military operation. The significant aspect of this engineering structure is that each advisor controls the information flow from one channel implemented within the BOS. The interfaces to these different parts of the BOS are encapsulated within each specialized BOS advisor sub-agent. In this way, the construction of the command agent can be incremental and only concern specific aspects the designer wishes to implement. This design also shows how this approach provides an easy facility to integrating with existing military systems while employing hybrid AI techniques.

COMMANDING OFFICER AGENT

Commanders from Battalion and above have staff positions held by subordinate officers that are responsible for different areas of an Army organization. At the battalion and brigade levels these positions are called Staff (S) positions. At division to army level they are held by General (G) Officers. The positions and their designations are described in Table 2.

Table 2: Staff/General Positions

Designation	Position
S/G 1	Personnel
S/G 2	Intelligence
S/G 3	Operations
S/G 4	Logistics
S/G 5	Civil Liaison

The S/G1 (Personnel) handles discipline, awards, and assignments. This position assures that men are available and placed properly within the organization, whether it is a battalion or an army.

The S/G2 is the intelligence officer responsible for security, counter-intelligence, and the Intelligence Preparation of the Battlefield (IPB). The IPB is basically the proper presentation of mission materials to indicate where enemy emplacements are believed to be and how the enemy is expected to behave.

The S/G3 is the highly important operations officer. His responsibilities encompasses the planning of the movement and disposition of all forces in the organization. This includes training, detailed mission plans, and movement of all equipment, men, and their support.

The S/G4 deals with the logistics of his unit. He facilitates the supply and maintenance of all things the organization will require.

The S5 is a position that is generated based on mission requirements. This position manages the politics, public relations, and interface to civil authorities sometimes required by Operations Other Than War (OOTW), such as humanitarian aid. At higher echelons this position becomes a necessary part of the organization otherwise it is filled as needed.

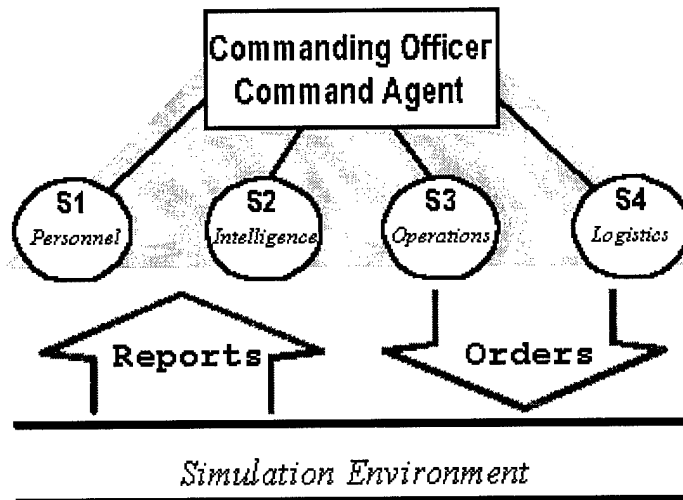


Figure 8: Commanding Officer Agent with Staff Advisors

Figure 8 shows the organization of the command agent architecture based on the commander and staff model. This shows the advisors being divided along the staff specializations: personnel, intelligence, operations, and logistics. The civil liaison duties would not commonly be required in a battle simulation. Figure 8 again shows the common theme of information flowing to and from the simulation environment which results in command behavior interaction with the simulation entities.

COMMAND AGENTS AT MULTIPLE ECHELONS

An interesting aspect of this problem is the creation of multiple levels of command agent interaction along echelon levels. The developer can create a hierarchy of command agents that act as both advisor and command agent. However, creating a stovepipe connection between command agents, as in Figure 9, can be detrimental to efficiency. This is true for real life military operations where informal, non-doctrinal communication relationships arise to assure the effective accomplishment of objectives, as in Figure 9.

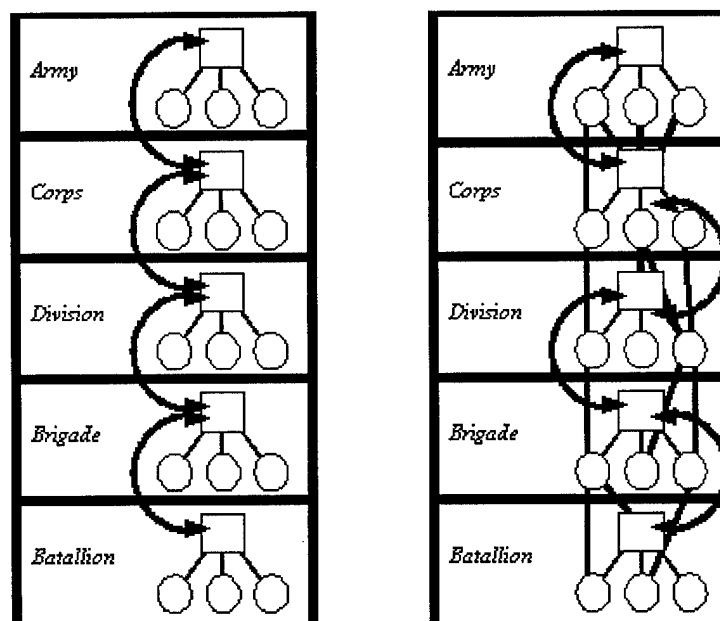


Figure 9: Multi-echelon Command Agents

CONCLUSION:

The implementation of a command agent must be designed for evolution in the changing simulation environment in which it will be employed. This requires a special architecture that can provide flexibility and scalability. The architecture described above can be used to design a command decision-maker that accounts for the highly dynamic domains of a simulated battlefield environment, but also allows for the implementation of heterogeneous strategies for reasoning about the environment. This architecture also allows for different models of command decision-making to be implemented. The proper implementation of this architecture can develop a command agent that functionally acts as a surrogate for a human commander in a military simulation.

[Return to the Top of this Section](#)

[Return to the Section 2 Index](#)

[Return to the Table of Contents](#)

[Return to the SCC Command Decision Modeling Page](#)

[Return to the WARSIM 2000 Page](#)

[Return to the NSC Home Page](#)

COMMAND DECISION MODELING TECHNOLOGY ASSESSMENT

Section 2

BACKGROUND:

- . COMMAND DECISION MODELING: An Overview
- . PROBLEM DEFINITION: Functional Description of a Command Agent
- . OBSERVATIONS REGARDING THREAT COMMAND AGENTS

BG1 Command Decision Modeling:

AN OVERVIEW

Frederick B. Buoni, Ph.D.

Florida Institute of Technology
Melbourne, Florida 32901
(407) 768-8000, ext. 7390
eMail: buoni@zach.fit.edu

SUMMARY

An architecture of a general decision process is constructed within the SHOR framework (Stimulus, Hypothesis, Options, Response) of Wohl. The process is decomposed into situation assessment and option assessment phases. The situation assessment process is composed of preprocessor, hypothesis generation, and hypothesis assessment subprocesses, while the

option assessment process is composed of criteria, option generation, option processing, and decider subprocesses. The context for the overall decision process is a command and control structure, but the architecture should apply to any decision-making situation.

The decision process architecture provides a framework for understanding the developments which are reported in the papers which follow this one.

SECTION I - INTRODUCTION

In this paper we present a general architecture for an Army tactical command and control (C₂) process which will enable the development of computer generated forces (CGF) for use in Distributed Interactive Simulation (DIS). This architecture presents a framework which allows the development of components using the applications of artificial intelligence (AI) and computer-based decision systems. The emphasis is on viewing the C₂ process as a decision process.

This work is based on a framework for tactical decision making which the author developed while on a summer faculty fellowship at the Jet Propulsion Laboratory in support of the Airland Battle Advanced Technology (ABAT) program at JPL [BUON83]. While in the original work, the emphasis was on providing decision support to a human decision maker, this effort is directed to support for computer decision making either under the supervision of a human controller or autonomously. While we retain the point of view that the system is computer-aided decision making with the human as the ultimate decision maker, the extension to an autonomous system is merely that there is no human intervention and that the computer could continue to direct the implementation of the decision it recommends. We refer to the system as a decision making system (DMS) whether it is to be a human decision maker or an autonomous computer decision maker.

In this paper we construct an architecture of a general decision process and discuss how it may represent a typical Army command and control process. This discussion is not definitive but is meant as an initial consideration to stimulate further investigation and definition. This work establishes an architecture within which real C₂ processes may be considered. The remaining papers in this volume will address the issues involved in using various techniques to implement components within this framework.

SECTION II - DECISION-AIDING NEEDS

The decision process has been studied by behavioral psychologists [JANI77], mathematicians [VONN44], operations researchers [VONW86, ZELE81], computer scientists [SIMO81], system engineers [GREE86, CACC92, REN 95, SAGE81, WOHL81], and military analysts [BEN-82, VRAN92]. The interest has broadened to include artificial intelligence researchers as we try to design systems for computer aiding of the decision process to respond to the needs of the military, business and the non-military governmental agencies [AZAR86, INGR92, POST90].

In today's world, a decision-making system (DMS) has a continually increasing amount of data available. Exotic sensor systems, global communications systems, and large data base systems provide the DMS with vast amounts of data. These same technologies also produce a

quickened world. The decision-maker must respond more quickly; there is less time to make decisions. This presents great pressure for the human decision-maker

[BARN81, ATHA83]. These pressures were exemplified by the Vincennes incident [BARR92].

The basic human decision-making capabilities remain limited. Short-term memory, the basis for perception and processing, is limited to two to four chunks of information, and it appears to take five seconds to place a chunk in long-term memory [SIMO81]. There are further difficulties: in retrieving information from long-term memory, response time is variable and cannot be guaranteed, and the information retrieved may be incomplete or altered.

The decision process itself can be conducted in various modes, some of which may be pathological [SAGE81, JANI77]. In a stressful situation it may be desirable to have a normative guide to assist in ensuring properly considered decisions.

Thus, the context for a DMS is established. Computer-based systems may be used to overcome human limitations in short-term memory, in rapid and accurate recall of data and information, and conducting the decision-making process. A framework for performing these tasks is the subject of the remainder of the paper.

An initial work for the Advanced Research Projects Agency (ARPA) examined the opportunity for decision-aiding for Command, Control, and Communication (C₃) environments in [NICK77]. They also examined the tactical communication effort [MADN82]. The Army Research Institute developed the Tactical Operations System [LEVI77, WITU80] and considered AI in the context of situation assessment [BEN-82, SAGE82]. The Air Force has examined decision-making models in tactical C₂ systems [BOET82, LEED79, LEVI82, PATT83, WOHL81], and in airborne information systems [CHU 82]. The Navy has examined the tactical C₂ process from the anti-submarine warfare (ASW) view [WOHL83a, WOHL83b], the undersea warfare view [DAVI82, DEGR82], the surface warfare view [FUNK82], the airborne view [BARN81], and a general discussion of man-machine task allocation [RIEG82]. The Office of Naval Research (ONR) sponsored Sage's general survey of decision-aiding processes [SAGE81]. NASA has been interested in the process monitoring function [GREE82, HAMM82, ROUS81], and in decision making in the cockpit of an aircraft [CHAP93]. This is not meant to be a thorough review of current work in the field; it only presents some readily available references in the unclassified literature.

The early significant thread leads from the ARPA survey [NICK77] to the Sage review [SAGE81] to Wohl's establishment of a framework for decision process analysis [WOHL81]. We note that much work in this area was done in the 1980's, but much less in the 1990's. There now appears to be significant interest in applications of AI, but much less in normative decision modeling.

SECTION III - DECISION PROCESS ARCHITECTURE

GENERAL CONCEPT

The focus of this analysis is on the decision-making process performed by a decision-making unit, which is called the decision-making system or DMS. The unit would normally consist of a

leader or commander and one or more assistants or operators and associated equipment. At this stage of the analysis the decision process will be examined without attributing functions or tasks to individuals or machines within the decision unit.

The decision unit is considered to be one subsystem within a hierarchical, multilevel system as shown in Figure 1 [MESA70]. The decision unit receives guidance, represented by h , from its supervisory unit or higher headquarters, and returns status information s . It issues instructions and orders u to its control units which are in contact with the external environment, and receives responses and progress reports r . The control unit actually attempts to influence the external states of nature through efforts m , and receives observables q . The input to the actual contact with the environment is w and the ensuing state of the situation is y . While this model represents all of the elements of the decision situation, it is too complex for consideration at this stage. It does, however, establish the context for the view of the decision unit, and the types of interactions with other subsystems and the external environment. Contact with an opposing or enemy force is considered as part of the environment and is represented within the states of nature w and y . Similar views are presented in [LAWS79, ATHA83].

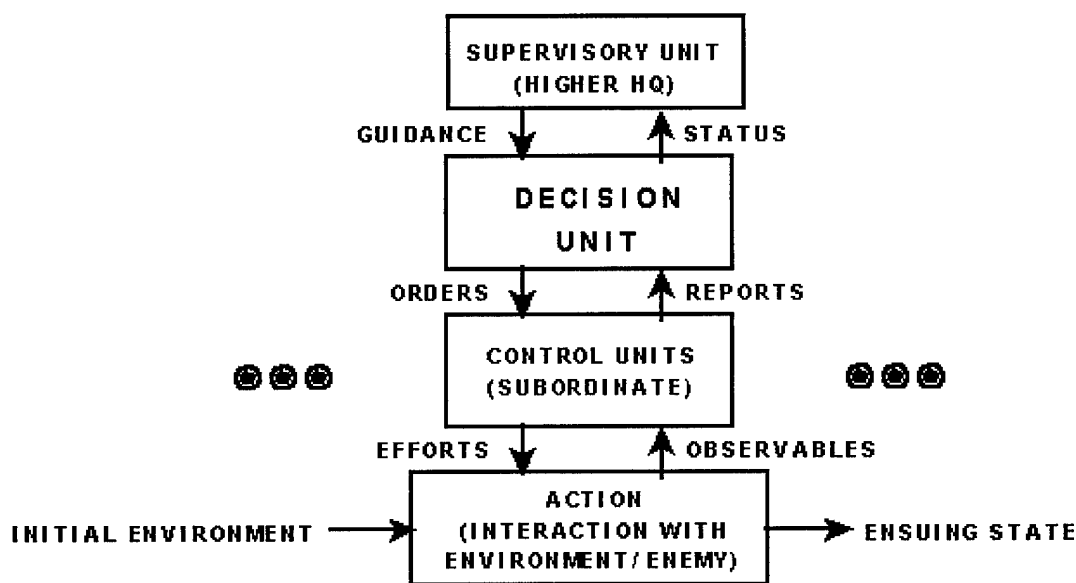
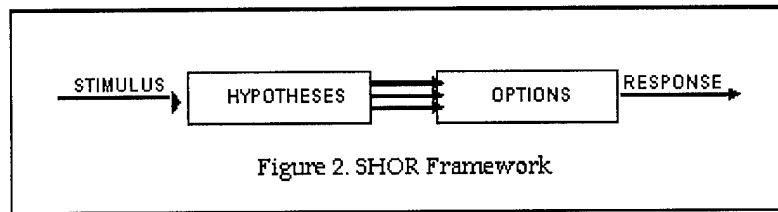


Figure 1. Hierarchical, Multilevel System

The model of the decision process is based upon the SHOR framework of Wohl, [WOHL81, WOHL83a, WOHL83b] shown in Figure 2. The acronym SHOR stands for stimulus, hypothesis, options, and response. A stimulus (S) is received thereby initiating the decision process. Based upon the stimulus and other information available at that time, various hypotheses (H) are generated concerning the actual situation or the actual state-of-the-world faced by the DMS. In consideration of the various possible situations it may be facing (the hypotheses), the DMS generates a set of options (O) or possible actions which it could direct to be taken. The effects or outcomes of the options are evaluated in view of the uncertain nature of the situation, and the DMS selects the appropriate action or response (R). The response then can be considered to interact with the external system generating additional stimuli which

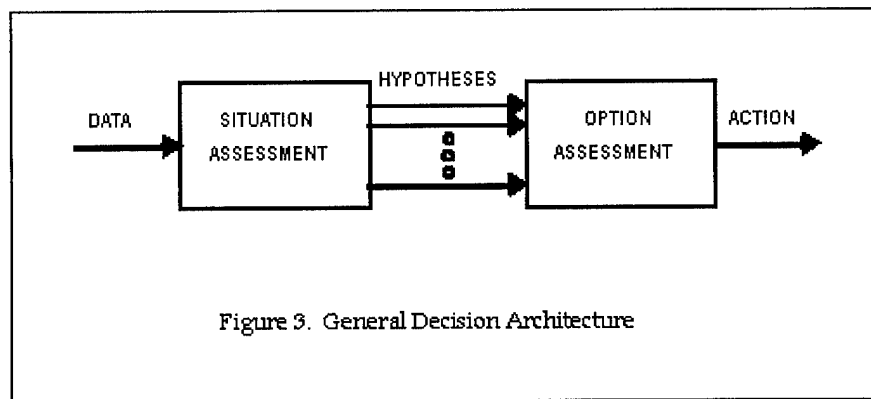
may lead to additional iterations of the process.



Similar frameworks have been developed by other investigators [LAWS79, RASM80, BOET82, DAVI82, ATHA83, LEVI83, RASM83]. While they are based on an assessment-response type of structure, they do include features germane to their investigations and do not provide the same architecture developed in this work.

THE GENERAL DECISION ARCHITECTURE

For this study the SHOR framework is adapted to produce the decision process architecture illustrated in Figure 3. In this view, data is received by the decision unit or DMS. The DMS conducts a situation assessment process (or sub-process) generating one or more hypotheses of the current situation, H_i . The DMS considers various options in the option assessment process, resulting in an action selected for implementation. For this current effort, no further interaction with the external environment is considered. Consideration will be concentrated on the situation assessment and option assessment subprocesses.



Additionally, the decision process is being modeled as an orderly sequential process. There are obvious opportunities for feedback, simultaneous processing, and jumping ahead in the real process. Consideration of these complications is deferred for subsequent studies.

In the following sections, we will further consider the Situation Assessment and Option Assessment subprocesses.

SITUATION ASSESSMENT

The situation assessment process considers newly acquired data in conjunction with currently available information and generates estimates of the current state of the environment or outside world [BEN-82]. Hypotheses, H_i , are developed concerning what is happening. Each of

these hypotheses, in turn, generates an interpretation of the state-of-the-world reflected in an estimated state vector x_i . The process also generates a measure of the likelihood of the hypothesis being true which is reflected as $P(H_i | Z)$. This is shown in the form of a conditional probability implying a Bayesian representation, but one may use other paradigms such as a possibility function [DOCK82], or Dempster-Shafer theory.

The concept of the situation assessment subprocess is shown in Figure 4. Incoming data are preprocessed by the Perception Processor to generate a vector Z which is suitable for further processing. Elements of the vector Z may be numerical, logical, or symbolic; whatever is appropriate to represent the input of the external world to the decision-making process. Operations such as calibration, transformation, aggregation, or correlation may occur within the preprocessor. It may use currently active hypotheses to establish the context for the processing. This processor may also include a process to assess the urgency with which the system should consider the situation, or to provide an alerting or bell-ringing function [PATT83, GREE82, WITU82, WOHL83b].

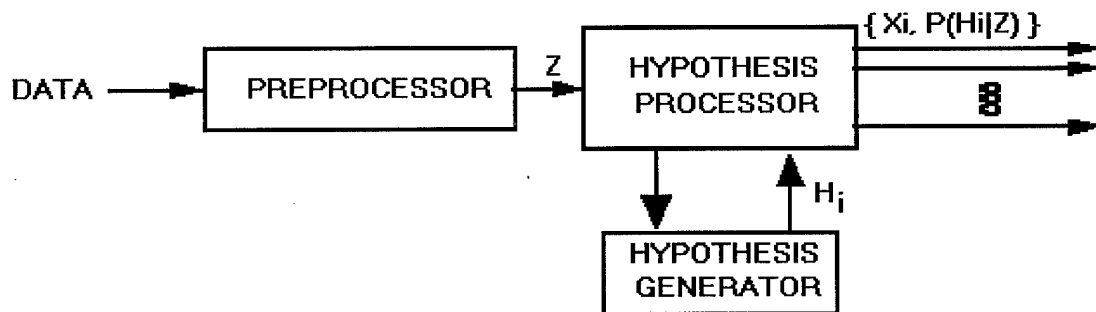


Figure 4. Situation Assessment Subprocess

The stimulus or initiating event for the decision process may be data-driven or concept-driven [WOHL83b]. The data-driven process occurs when specific received data are input to the system and are recognized as a stimuli. The concept-driven process occurs when a hypothesis or concept is presented for consideration, and then data or information is sought to evaluate the concept. Not much is definitely known about how hypotheses are generated. Some are already in memory and can be brought into active consideration [WOHL83b]. Others may be stimulated by specific features of the data Z . Still others may be created by mental processes leading to possible application of neural networks.

The Hypothesis Processor may be viewed as performing a pattern-matching process (see Figure 5). Each of the current active hypotheses in addition to forming an expected state vector X_i also form an expected view of what the incoming data should be. This expected value of Z under H_i or X_i can be represented as z_i . A pattern-matching process may then generate a set of variations and the decision-maker assess if the matches are satisfactory. If not, then the DMS generates additional hypotheses which are placed in the current hypothesis short-term memory until the DMS is satisfied with the set of hypotheses and state vectors under active consideration.

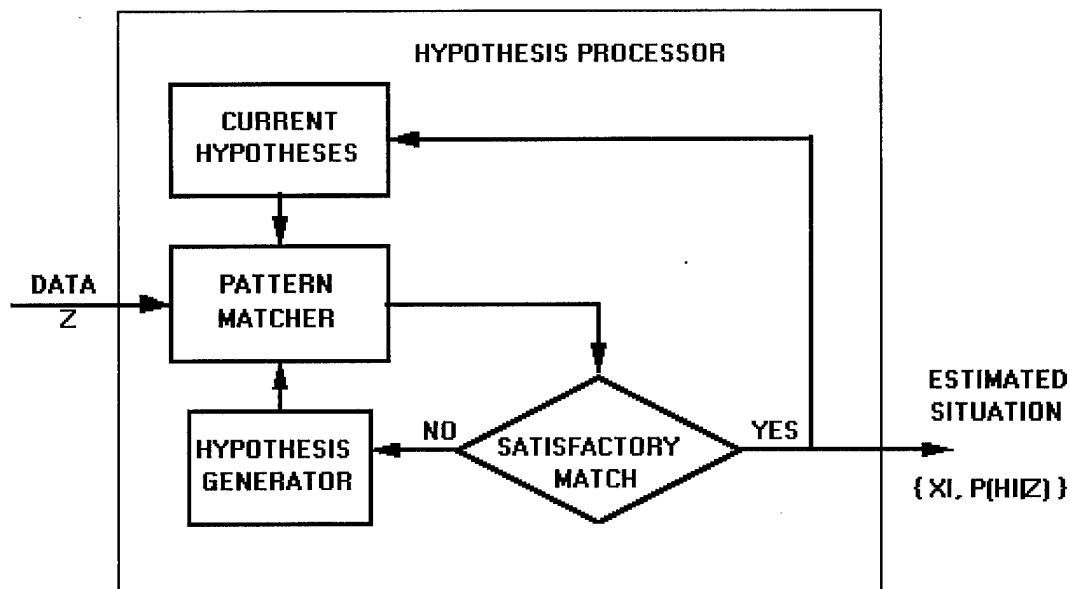


Figure 5. Hypothesis Processor

OPTION ASSESSMENT

The option assessment subprocess uses the hypotheses and the estimates of the situation facing the decision maker system as input, generates and evaluates alternative courses of action available to the DMS, and then selects a course of action for implementation.

This subprocess can be further decomposed into the subprocesses shown in Figure 6. Objectives and criteria are provided to the DMS from higher headquarters, from manuals, procedures, and doctrine, and from the DMS's experience and wisdom. It is assumed that some articulation of these is possible. It is also valuable to have some assessment of the relative importance or value of the objectives, goals, and criteria.

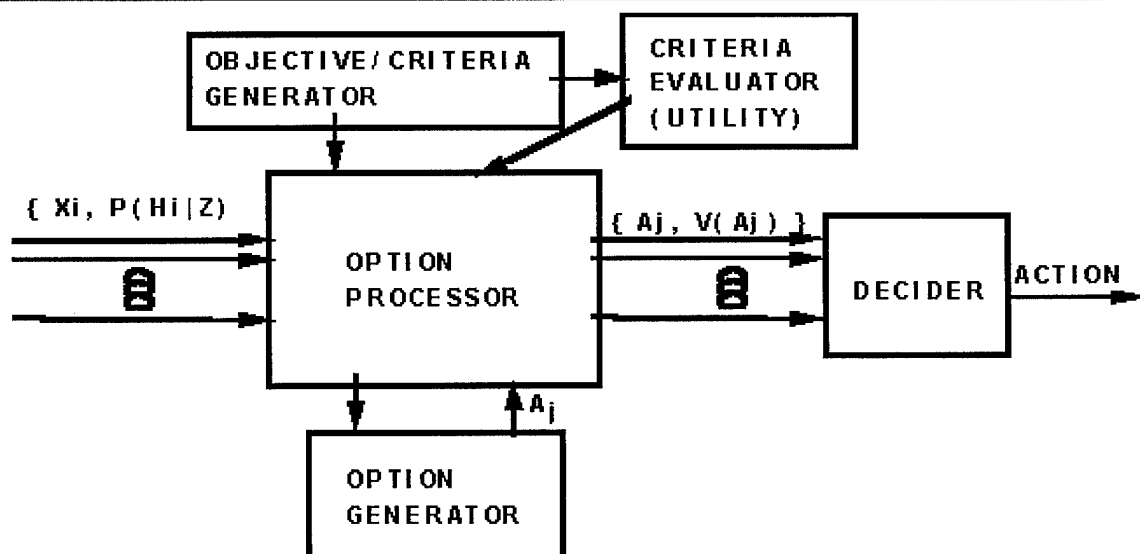


Figure 6, Option Assessment Subprocess

Options or courses of action open to the DMS are available within the memory structure or are provided by the option generator process. Outcomes of the actions under active consideration, a_j , will depend upon the true state-of-the-world, but can be estimated for each of the expected state vectors X_i . The degree to which the possible outcomes of each action meet the goals and objectives may be evaluated considering the relative value of the goals and objectives, and the degrees of belief, probability or possibility of the hypotheses. The nature of this evaluation process cannot be *a priori* defined for any decision-maker, but should allow choice. For this stage of the development it is assumed that some value $V(a_j)$ is available for each action a_j under consideration. This process can be viewed as a continuum of complexity from a simple single-criterion problem to a multi-objective, multi-criteria, nonlinear problem which could be analytically intractable. Whatever the nature of the evaluation process, it is assumed to be captured within the model.

At this point it may be useful to consider the modes of decision-making behavior developed by Rasmussen [HOLL83, RASM83, WOHL83b]. He postulated three decision-making modes: the skill-based mode, the rule-based mode, and the knowledge-based mode. The skill-based mode is characterized by strong habitual behavior. Actions are taken in response to stimuli with little or no conscious effort. This is the reflex action of a skilled operator or practitioner. Actions or decisions made in the rule-based mode are governed by procedures or doctrines. In this case, more effort is required than in the skill-based mode, to find the governing rule or procedure. Once found, however, the appropriate action is determined. Both the skill-based and rule-based modes depend on recognizing the triggering state-of-the-world with sufficient certitude given the urgency of the situation. The knowledge-based mode requires the DMS to extract appropriate information to deduce the appropriate action. This mode requires the full benefit of this decision model. The expanded Option Processor is shown in Figure 7.

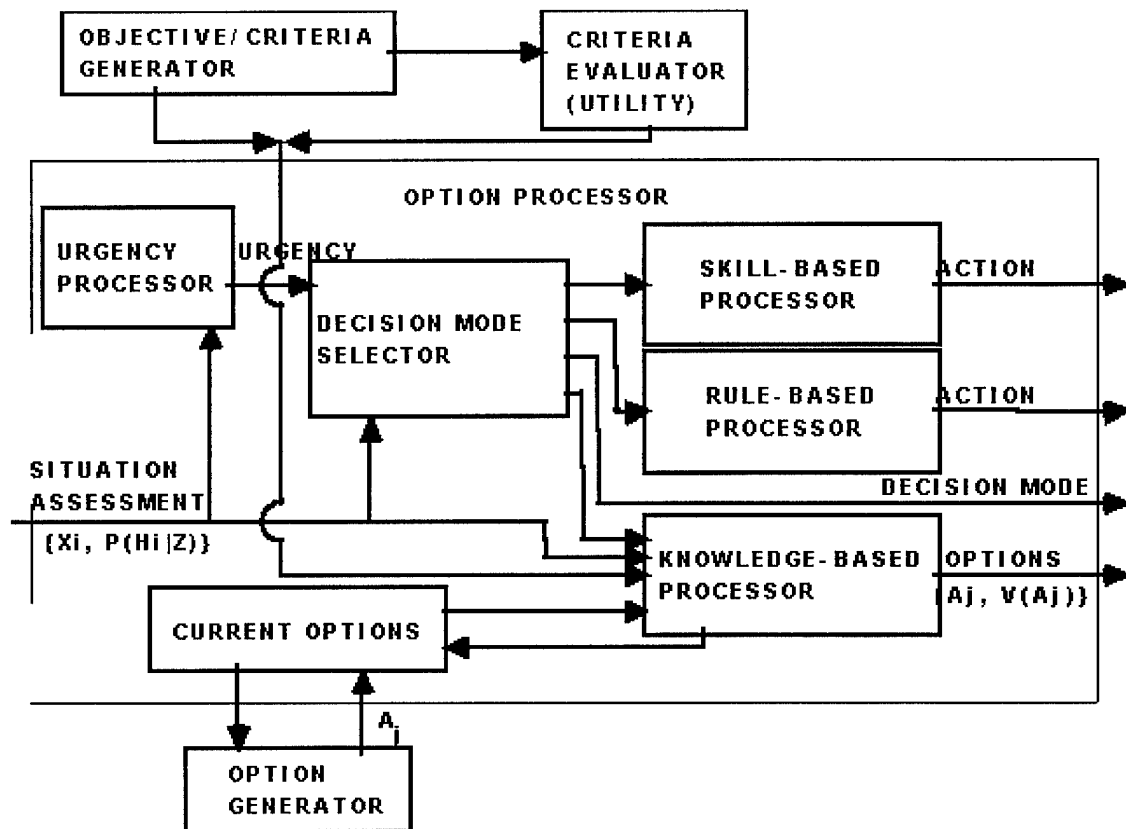


Figure 7. The Option Processor

We should realize that the terms used by Rasmussen do not carry the intention of meaning identical to those which the AI community uses. The blocks in Figure 7 called Rule-Based Processor and Knowledge-Based Processor could be implemented by any appropriate representational model. Not only should rule-based expert systems be considered, but neural networks, model-based systems, genetic algorithms, or others are appropriate candidates.

THE DECIDER

The processes to this point have been concerned with providing the DMS with the informational and procedural resources to make the decision. The actual selection of an action or the decision is made in the Decider process. This process is represented in Figure 8.

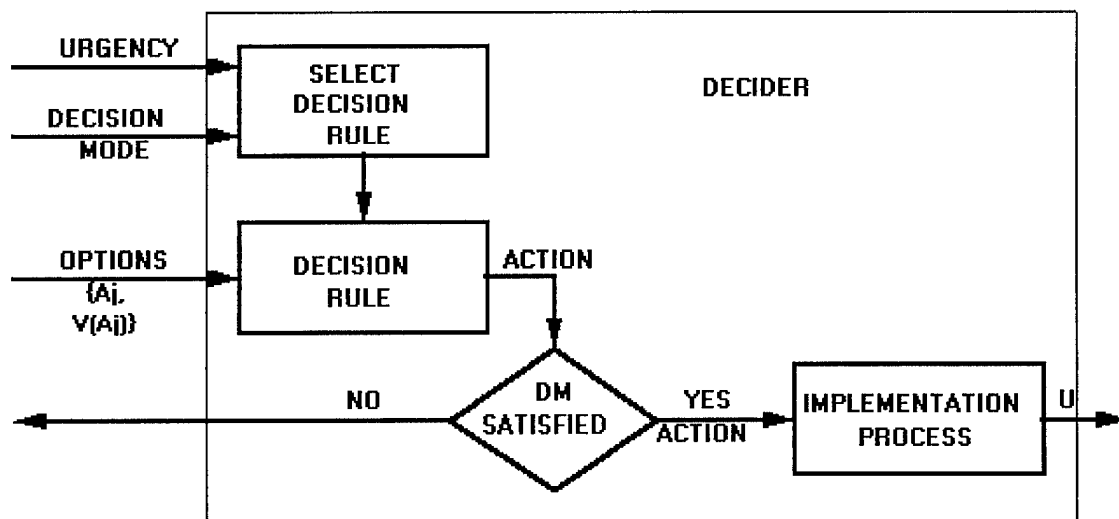


Figure 8. The Decider

In his excellent review article, Sage describes various models of decision-making behavior and effects of stress and complexity [SAGE81]. For this model, it is assumed that as a result of some interaction, a choice is made of decision rule to use, and then the decision rule is applied to select the appropriate action for implementation. In the case of skill-based or rule-based behavior, that action will be already determined. In the case of knowledge-based behavior, one may still select a satisfying decision rule [SIMO81] (also called bounded rationality), a rational actor model, a garbage can model, or some model for making the action selection decision.

Even once this is done, the DMS has one last chance to review the situation and decide if he is really willing to make this decision. If not, the process can cycle back to an earlier stage and go through another iteration. If the DMS is satisfied, it can proceed with implementation. One can go through the decision process and get so caught up with it that some perspective may be lost. This last decision acts as a final check, and, by personal experience, often results in NO result.

We have shown an Urgency variable with input into the Select Decision Mode process. Based on the current hypothesized situation assessment, the Urgency variable will assess a time by which an action should be selected and implemented. The three option processors, the Skill-Based, Rule-Based, and Knowledge-Based Processors will all operate in parallel through blackboard structures so that a "best" action may be selected by the time that it is required.

IMPLEMENTATION

The implementation process represents converting the selected action into an operable instruction u which will cause some interaction with the external environment. It could represent the issuance of an order or communication. In a complete modeling sense, there may be some further planning necessary to translate the selected action into an executable instruction.

EXTERNAL ENVIRONMENT

Once the decision is made and appropriate executable instructions issued, there is some

interaction with the external environment. The results of that interaction may be sensed and that data input to the process to begin another decision. In that sense the process may be a control process. Of course, the process may be open-ended with no subsequent processing.

At several stages in this model, decisions were made as part of the decision process. Since the model, as developed, is general in scope, it can be nested so that each of the individual decisions could be represented in terms of this model.

SECTION IV - CONCLUSIONS

Department of Defense research agencies, ARPA, NRL, AFOSR, and ARI, have begun to examine decision-making and ways AI can be used to assist in the decision-making process. This work developed a general architecture of a decision process using Wohl's SHOR paradigm (Stimulus, Hypothesis, Options, Response) and carrying the decomposition to more detail than previously done, but consistent with the thrust of the DOD work. The possible use of AI techniques to support the decision processes in the model structure are left to others to discuss. AI provides the promise of tools to help the military decision-maker cope with the increasing mass of data and information and of the decreasing response times of the future battlefield. It may also help to provide the decision-makers the capacity for making better decisions. It will, however, provide capability for autonomous agents for use in simulation systems.

The architecture was constructed within the SHOR paradigm. The process was decomposed into situation assessment and option assessment phases. The situation assessment phase is composed of preprocessor, hypothesis generation, and hypothesis assessment subprocesses. The option assessment phase is composed of criteria, option generation, option processing, and decider subprocesses. The context for the overall decision process is a command and control structure, but the architecture applies to any decision-making situation.

REFERENCES:

- ATHA83 Michael Athans, "System Theoretic Challenges in Military C₃ Systems," Naval Research Reviews, Vol. XXXV, No. 2, 1983, pp. 18-28.
- AZAR86 Jerome Azarewicz, "Plan Recognition for Airborne Tactical Decision Making," AAAI-86, pp. 805-811.
- BAIL82 Robert W. Bailey, Human Performance Engineering, Prentice-Hall, 1982.
- BARN81 M.J. Barnes, Human Information Processing Guidelines for Decision-Aiding Displays, Naval Weapons Center, China Lake, CA, NWC, Technical Memorandum 4605, December 1981. AD-A124858. N83-02023105.
- BARR92 John Barry and Roger Charles, Newsweek, July 13, 1992, pp. 29-39.
- BASA81 Tamer Basar and Jose B. Cruz, Jr., Concepts and Methods in Multi-Person Coordination and Control, University of Illinois at Urbana-Champaign, Technical Report UILU-ENG-81-2251, R-920(Dc-49), October 1981, 77 pages. AD-A124386.

BEN-82 Moshe Ben-Bassat and Amos Freedy, "Knowledge Requirements and Management in Expert Decision Support Systems for (Military) Situation Assessment," IEEE Trans. Syst., Man, Cybern., Vol. SMC-12, No. 4, pp.479-490, July/August 1982.

BOET82 Kevin L. Boettcher and Alexander H. Levis, "Modeling the Interacting Decision Maker with Bounded Rationality," IEEE Trans. Syst., Man, Cybern., Vol. SMC-12, No. 3, pp. 334-344, May/June 1982.

BUON82 Frederick B. Buoni, Decision Support Systems, Appendix L of Computer Science: Key to a Space Program Renaissance, Final Report of the 1981 NASA/ASEE Summer Study on the Use of Computer Science and Technology in NASA, University of Maryland Technical Report 1168, January 15, 1982.

BUON83 Frederick B. Buoni, A Process Architecture for Computer-Based Decision Support, Florida Institute of Technology working paper, August 1983.

CACC92 Pietro Carlo Cacciabue, Françoise Decortis, Bartolome Drozdowicz, Michel Masson, and Jean-Pierre Nordvik, "COSIMO: A Cognitive Simulation Model of Human Decision Making and Behaviour in Accident Management of Complex Plants," IEEE Trans. Syst., Man, Cybern., Vol. 22, No. 5, pp. 1058-1074, September/October 1992.

CHAP93 Alan R. Chappell, Knowledge-Based Reasoning in the Paladin Tactical Decision Generation System, NASA Contractor Report 4507, Lockheed Engineering and Sciences Company, 1993.

CHU 82 Yee-yeen Chu and Amos Freedy, "Computer-Aided Information Handling in Supervisory Control of Airborne Systems," IEEE 1982 Proc. Conf. Cybern. Soc., pp. 425-429, 1982.

COMP82 Michael A. Companion and Gregory M. Corso, "Task Taxonomies: A General Review and Evaluation," Int. J. Man-Machine Studies, Vol. 17, No. 8, pp. 459-472, 1982.

DAVI82 M. Dianne Davis, "Cybernetics in the Submarine Society: The Human Use of High Technology," 1982 Proc. Int. Conf. Cybern. and Soc., pp. 223-227, 1982.

DEGR82 Edward DeGregorio, Ann Silva, and David Brazil, "Applied Artificial Intelligence in the Submarine Combat Control Environment," 1982 Proc. Int. Conf. Cybern. Soc., pp. 506-510, 1982.

DOCK82 John T. Dockery, "Fuzzy Design of Military Information Systems," Int. J. Man-Machine Studies, Vol. 16, pp. 1-38, 1982.

FITT80 Mike Fitter and Max Sime, "Creating Responsive Computers: Responsibility and Shared Decision-Making," in Smith and Green, Ed's, Human Interaction With Computers, Academic Press, 1980, pp. 39-66.

FUNK82 Ken Funk, "Symbolic Structures and the Human Operator," IEEE 1982 Proceedings of

the International Conference on Cybernetics and Society, October 28-30, 1982, pp. 156-160.

GREE86 Joel S. Greenstein and Mark E. Revesman, "Development and Validation of a Mathematical Model of Human Decisionmaking for Human-Computer Communication," IEEE Trans. Syst., Man, Cybern., Vol. SMC-16, No. 1, pp. 148-154, 1986.

GREE82 Joel S. Greenstein, and William B. Rouse, A Model of Human Decision-making in Multiple Process Monitoring Situations, IEEE Trans. Syst., Man, Cybern., Vol. SMC-12, No. 2, pp. 182-193, 1982.

HAMM82 John M. Hammer and William B. Rouse, "Design of an Intelligent Computer-Aided Cockpit," 1982 Int. Proc. Conf. Cybern. Soc., pp. 449-453.

HOLL83 Erik Hollnagel, "What We Do Not Know About Man-Machine Systems," Int. J. Man-Machine Studies, Vol. 18, No. 2, pp. 135-143, 1983.

INGR92 Francois F. Ingrand, Michael P. Georgeff and Anand S. Rao, "An Architecture for Real-Time Reasoning and System Control," IEEE Expert, pp. 34-44, December 1992.

JANI77 Irving L. Janis and Leo Mann, Decision Making: A Psychological Analysis of Conflict, Choice and Commitment, Free Press (MacMillan) 1977.

JOHN82 Scott Kevin Johnson, An Investigation of a Land Combat Tactical Commander's Decision-Making Process, Naval Postgraduate School Thesis, October 1982. AD A124990.

LAWS79 Joel S. Lawson, Jr., "Naval Tactical C3 Architecture 1985-1995," Signal, Vol. 33, No. 10, August 1979, pp. 71-76.

LEED79 D. K. Leedom, "Representing Human Thought and Response in Military Conflict Simulation Models," in Symp. on Modelling and Simulation of Avionics Syst. and Command, Contr. and Com. Syst., AGARD (NATO) Conf. Proc., No. 268, Nat. Tech. Inform. Ser., Oct. 15-19, 1979.

LEVI77 Robert A. Levit, Berton J. Heaton, David G. Alden, "Development and Application of Decision Aids for Tactical Control of Battlefield Operations: Decision Support in a Simulated Tactical Operations System (SIMTOS)," Honeywell Systems and Research Center, Minneapolis, MN, ARI Technical Report TR-77-A13, December 1977, AD A121413.

LEVI82 Alexander H. Levis, Elizabeth R. Ducot, Michael Athans, Distributed Decision and Communication Problems in Tactical USAF Command and Control, Laboratory for Information and Decision Systems (MIT), July 30, 1982. Technical Report LIDS-IR-1226, AFOSR-TR-82-0952. AD A121413.

LEVI83 Alexander H. Levis and Kevin L. Boettcher, "Decisionmaking Organizations with Acyclical Information Structures," IEEE Trans. Syst., Man, Cybern., Vol. SMC-13, No. 3, May/June 1983, pp. 384-391.

MADN82 Azad M. Madni, Michael G. Samet, and Amos Freedy, "A Trainable On-Line Model

of the Human Operator in Information Acquisition Tasks," IEEE Trans. Syst., Man, Cybern., Vol. SMC-12, No. 4, pp. 504-511, July/August 1982.

MESA70 M. D. Mesarovic, D. Macko, and Y. Takahara, Theory of Hierarchical, Multilevel Systems, Academic Press, 1970.

NICK77 R. S. Nickerson, M. J. Adams, R. W. Pew, J. A. Swets, S. A. Fidell, C. E. Feehrer, D. B. Yntema, D. M. Green. The C3 System User Vol. I: A Review of Research on Human Performance as it Relates to the Design and Operation of Command, Control and Communication Systems. Bolt Beranek and Newman, Inc. Technical Report 3459, Feb. 1977, AD-A126633.

PATT83 Krishna R. Pattipati, David L. Kleinman, and Arye R. Ephraim, "A Dynamic Decision Model of Human Task Selection Performance," IEEE Trans. Syst., Man, Cybern., Vol. SMC-13, No. 3, pp. 145-166, March/April 1983.

POST90 Stephen Post and Andrew P. Sage, "An Overview of Automated Reasoning," IEEE Trans. Syst., Man, Cybern., Vol. 20, No. 1, pp. 202-224, January/February 1990.

RASM80 Jens Rasmussen, "The Human as a Systems Component," in Smith and Greed (Ed's), Human Interaction with Computers, Academic Press, 1980.

RASM83 Jens Rasmussen, "Skills, Rules, and Knowledge: Signals, Signs, and Symbols, and Other Distinctions in Human Performance Models," IEEE Trans. Syst., Man, Cybern., Vol. SMC-13, No. 3, pp. 257-266.

REN 95 Jie Ren and Thomas B. Sheridan, "An Adaptive Decision Aid for Real Environments," IEEE Trans. Syst., Man, Cybern., Vol. 25, No. 10, pp. 1384-1391, October 1995.

RIEG82 Christine A. Rieger and Joel S. Greenstein, "The Allocation of Tasks Between the Human and Computer in Automated Systems," in IEEE 1982 Proceedings of the International Conference on Cybernetics and Society, October 28-30, 1982, pp. 204-208.

ROUS81 William B. Rouse, "Human-Computer Interaction in the Control of Dynamic Systems," Computing Surveys, Vol. 13, No. 1, pp. 71-99, March 1981.

SAGE81 Andrew P. Sage, "Behavioral and Organizational Considerations in the Design of Information Systems and Processes for Planning and Decision Support," IEEE Trans. Syst., Man, Cybern., Vol. SMC-11, No. 9, pp. 640-678, September 1981.

SAGE82 Andrew P. Sage and Adolfo Lagomasino, "Knowledge Representation and Interpretation in Decision Support Systems," IEEE 1982 Proc. Conf. Cybern. Soc., pp. 658-662, 1982.

SIMO81 Herbert A. Simon, The Sciences of the Artificial, Second Edition, MIT Press, 1981.

VONN44 John Von Neumann and Oskar Morgenstern, The Theory of Games and Economic Behavior, Princeton, 1944.

VONW86 Detlof Von Winterfeldt and Ward Edwards, Decision Analysis and Behavioral Research, Cambridge University Press, 1986.

VRAN92 Sanja Vranes, Mario Lucin, Mladen Stanojevic, Violeta Stevanovic and Pero Subasic, "Blackboard Metaphor in Tactical Decision Making," European Journal of Operational Research, Vol. 61, pp. 86-97, 1992.

WITU80 Gary Witus, Robert W. Blum, Mark Graulich, Description of the Tactical Operations System Information Flow Model, Vector Research, Inc. Report VRI-ARI-3-FR79-3, 30 November 1979, ARI Research Note 80-13, AD-A092206, May 1980.

WITU82 Gary Witus, Robert W. Blum, Information Flow Management for Distributed Tactical Command Control Systems, IEEE Trans. Syst., Man, Cybern., Vol. SMC-12, No. 4, pp. 512-518, 1982.

WOHL81 Joseph G. Wohl, "Force Management Decision Requirements for Air Force Tactical Command and Control," IEEE Trans. Syst., Man, Cybern., Vol. SMC-11, No. 9, pp. 618-639, September 1981.

WOHL83a Joseph G. Wohl, E. E. Entin, M. G. Alexandridis, J. S. Eterno, Toward a Unified Approach to Combat System Analysis, Alphatech, Inc. Technical Report TR-151, January 1983, ADA124570.

WOHL83b Joseph G. Wohl, E. E. Entin, J. S. Eterno, Modeling Human Decision Processes in Command and Control, Alphatech, Inc. Technical Report TR-137, January 1983, AD A125218.

ZELE81 Milan Zeleny, Multiple Criteria Decision Making, McGraw-Hill, 1981.

ACKNOWLEDGMENTS

This work was conducted while the author was a NASA/ ASEE Summer Faculty Fellow assigned to the Jet Propulsion Laboratory (JPL). The ten-week term of the fellowship was spent with the Airland Battle Advanced Technology (ABAT) program in association with Dr. Victor J. Anselmo.

The author may be contacted at the following address:

Dr. Frederick B. Buoni
College of Engineering
Florida Institute of Technology
Melbourne, FL 32901

(407) 768-8000, ext. 7390
E-mail: buoni@zach.fit.edu

[Return to the Top of this Section](#)

BG2 *PROBLEM DEFINITION:*

A Functional Description of a Command Agent

**Howard Mall, M.S.C.S.; Rick McKenzie, Ph.D.; and
Jenifer McCormack, Ph.D.**

Science Applications International Corporation
3045 Technology Parkway
Orlando, Florida 32826-3299
(407) 282-6700
Howard_Mall@cpqm.saic.com
Rick_McKenzie@cpqm.saic.com

OVERVIEW:

A **command agent** is an intelligent agent that will act as a surrogate for a human commander in a military simulation. To develop command agents that are designed for change within the evolving simulation domain, requires an architecture which will allow the heterogeneous application of decision-making strategies, the management of myriad interfaces to simulation services and knowledge sources, and the proper expression of command behavior. The approach suggested is to develop an integrated collection of sub-agent **advisors** which is arbitrated and controlled by an overwatching **command agent**. This architecture can be applied to many different domains and be used to model different approaches to command decision making.

THE COMMAND DECISION-MAKING DOMAIN

Army commanders, by their very definition, make decisions. They make decisions on how to act in a given situation. They then formulate orders that are given to their subordinates who in turn can make decisions about how to act and give orders to their subordinates. This is the hierarchical structure of the military (see Fig. 1, BG 1) that allows a commander to control a large number of men and materiel without being overwhelmed with information. It also allows specialization in which one individual can gain deep knowledge about a specific aspect of the military domain, such as intelligence, logistics, artillery, etc.

Computer Generated Forces (CGF) must demonstrate realistic and valid behaviors of military entities in order to provide an effective training environment or predictive capability. In order to exhibit advanced coordination and cooperation of military units (which are found on a real battlefield) simulations turn toward modeling the commander of those forces to provides the Command, Control, and Communication (C³) required.

The implementation of a model utilized to simulate the command decision making process will hereafter be referred to as a **command agent**. "An agent is a surrogate for a person or process that fulfills a stated need or action. []" A command agent provides a framework for the aggregation and abstraction of behaviors of various military echelons. It also allows the C³ process to be closely patterned after the one utilized in the real world. The SHOR framework (see Figure 2, BG 1) indicates a proven model for initiating and executing decision making processes.

Figure 3 (BG 1) shows the SHOR framework as applied within CGF. Decision making is divided into two phases: one for situation assessment followed by the assessment of options. Data received from the simulation environment stimulates the system to carry out assessments of situations and generate hypotheses. These hypotheses represent assessments of the situation. The option assessment phase uses these hypotheses to generate options (in military parlance: courses of action). Some decision criteria are generated and used to select an option for execution.

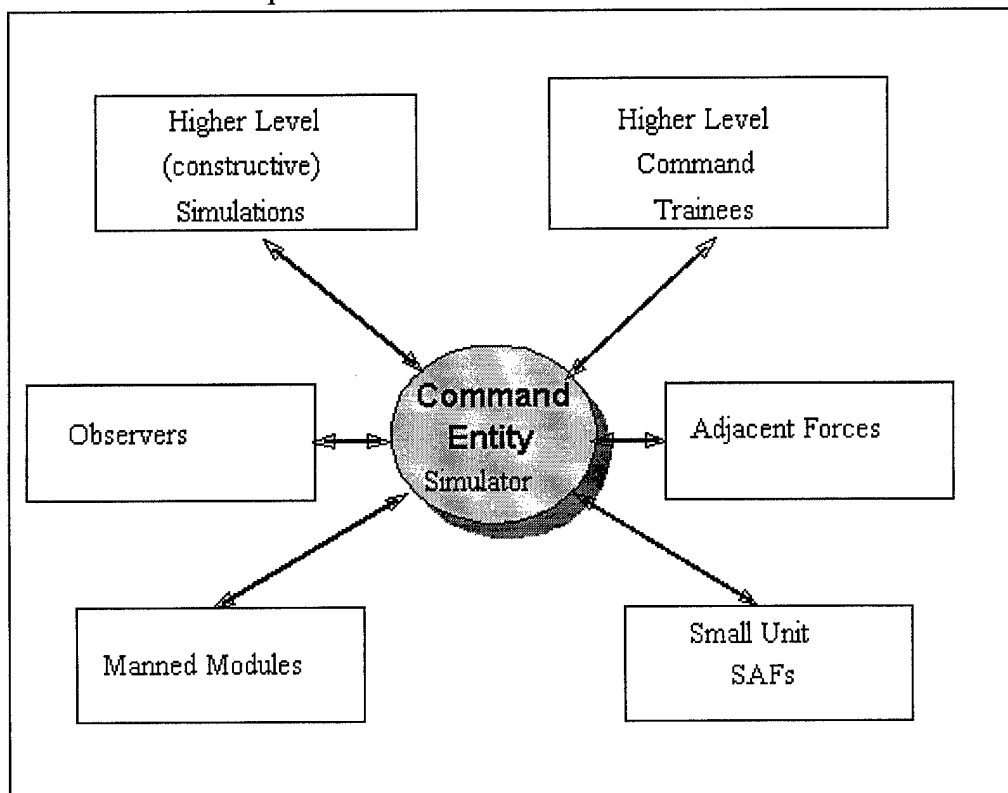


Figure 1: Command Agent Interfaces

Figure 1: Command Agent Interfaces

IMPLEMENTATION IMPLICATIONS

The general description of the command decision making domain described above and in [task1] captures the process appropriately. However, for military simulations, there are many different implementation issues that must be addressed. In a military simulation there are a variety of interfaces that a command agent must be capable of utilizing. Figure 1 shows a representative collection of systems that are involved in a modern Distributed Interactive Simulation (DIS) environment.

The growing trend in the military simulation community is for greater degrees of system interoperability, computer resource distribution, and entity and aggregate behavioral complexity. Simulations of today must find some optimal solution to balance these contentious goals. A command agent must also be designed and implemented for these varying degrees of integration.

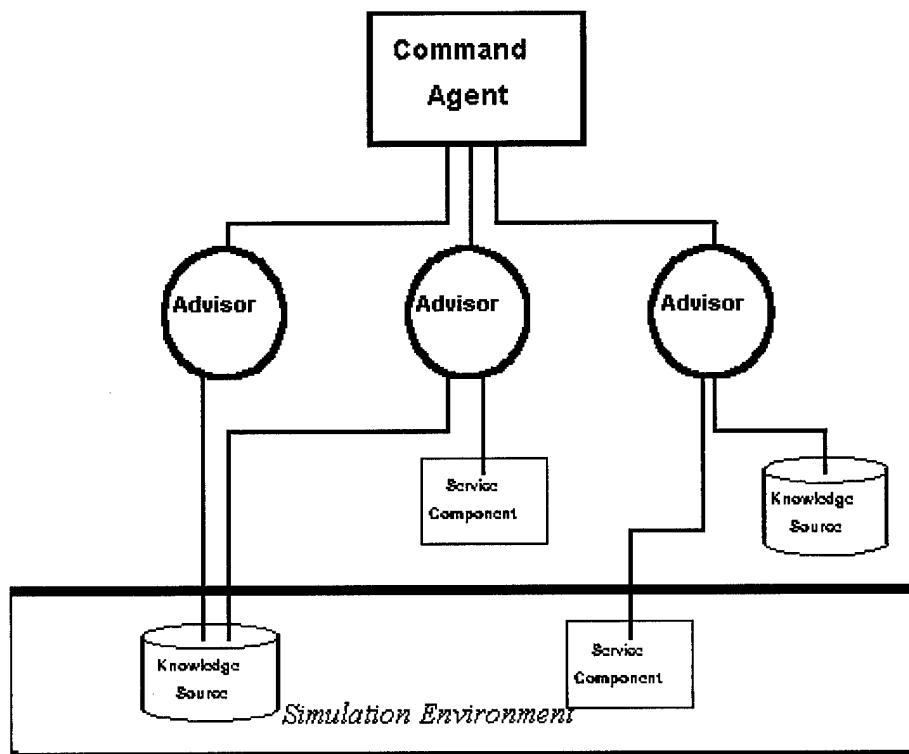


Figure 2: Command Agent Architecture

Complex decision making systems are typically a hybrid of artificial intelligence techniques. They deal with diverse types of information and knowledge that can be used in different contexts. A result of the work reported in [] was the need for a command agent to deal with environments rich with variegated information. The design of a command agent should reflect the need for heterogeneous integration both internally and externally. Figure 2 shows the generic layered approach suggested for constructing a command agent. A detailed application of this approach can be found in [] where "many different reasoning processes, societies of agents, are integrated in order to realize a software assistant capable of performing a broad range of tasks."

This design allows the implementation of different artificial intelligence techniques to be applied to their appropriate categories of problems. It also provides facilities to abstract the interfaces to external knowledge sources and simulations. The key to this approach is the implementation of advisors which are sub-agents constructed to monitor, interpret, and reason about information flowing into the command agent from various sources (i.e. terrain databases, sensor objects within the simulation, reports from subordinate units, etc...) The process modeled within the command agent can remain stable while new interfaces to various simulation components can be managed and translated by the advisors.

To correlate with the SHOR framework (Figure 3, BG1), stimulus is communicated from the simulation environment through the command agent's advisors. The advisors handle the API's to heterogeneous components within the simulation, which provide information and accept direction. Advisors receive information, either passively or through query, and synthesize the information into situation hypotheses. These hypotheses are translated into the command agent's internal representation. The command agent acts to arbitrate advisor input and to control the services and interfaces that the advisors provide. The command agent has knowledge of how to utilize the analyses of its advisors to make decisions.

This implementation corresponds to a model that is very close to the way modern command decision-making is done. The commander receives knowledge through his aides or advisors that analyze and interpret with specialized expert knowledge. The information is then presented to the commander as icons on a map or some other abstract representation that saves the commander from processing large amounts of irrelevant information. The commander receives only information appropriate to his decision-making process.

The command agent structure not only coincides with real-world models but it also provides many software engineering advantages as well. In order to add new services or other capabilities to the command agent, the developer need only extend an existing advisor or add a new one. This extension would not significantly impact the other components due to the layer of abstraction defined between the advisors and the command agent. This advantage applies, also, to advisors; whereby, they can supply the same services to the command agent but utilize new sources of information by switching APIs.

The command agent design provides a flexible architecture that allows modular functionality, extensibility, and interoperability. These requirements are necessary to perform within the diverse and plentiful simulation federations being fielded today. The implementation of a command agent can take many forms and depends on many factors derived from the environment in which command agent is intended to operate.

COMMAND AGENT CONSTRUCTION:

The command agent is composed of a set of sub-agent advisors that specialize in generating hypotheses about the situation environment and the decision making component that supervises the advisors and makes decisions based on the information they produce.

COMMAND AGENT:

The command agent's main role is to control the operation of its advisors and make decisions based on the results of these advisors. The command agent is made up of the following components:

- Internal Representations
- Inference Engine
- Command Behavior Model
- Agent Arbitration and Control Knowledge
- General Sub-Agent Interface

The command agent is an intelligent agent. "An intelligent agent is composed of the represented knowledge that drives its functions. This knowledge is described in the form of goal(s) and process(es) that lay out both the objectives for the agent and manner in which it will perform its tasks." []

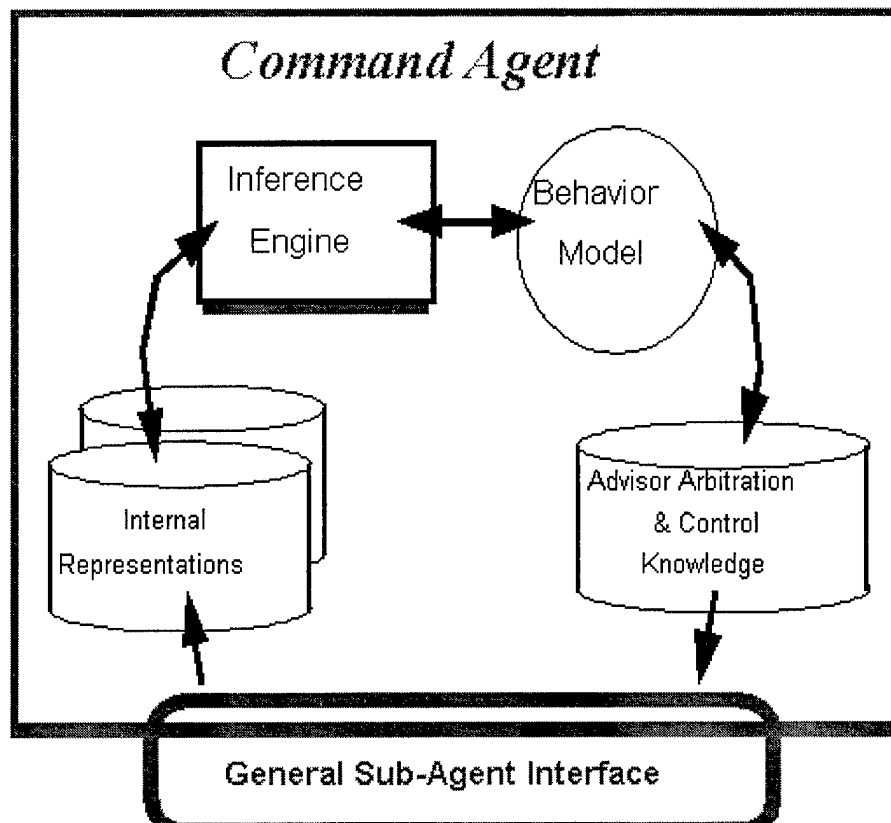


FIGURE 3: COMMAND AGENT INTERNALS

Internal Representations and Inference Engine

The Command Agent retains internal representations that represent its perceptions of the battlefield environment. These internal representations are updated and maintained by the flow of information from the Command Agent's Advisor sub-agents. Within [ii] the authors chose to use two representations to store battlefield knowledge: a semantic network and a tessellated terrain map. The choice of internal structures can depend on the level of abstraction required in order to accomplish the command decision-making process efficiently and effectively.

Frame-based systems appear to be the best representation for the internal knowledge of the command agent. This technique follows an object-oriented strategy to depict the concepts and relationships that the command agent must handle. It provides a degree of flexibility and resolution for organizing battlefield information, and it requires a simple inference engine processing control rules to utilize the information. This corresponds to the human commander who does not cognitively handle the vast amounts of data that flows around a battlefield. Instead, he relies on trusted advisors to fuse, filter, and extrapolate important information from this flow of data. Decisions are made based on these abstract views.

The frame-based system is recommended because it provides a general structured approach to modeling battlefield knowledge that applies at different echelon levels and different levels of complexity; it can be as simple or complex as the developer wants to make it. It is, as pointed out above, cognitively similar to the way human commanders conduct their decision-making process. Inference engines derived from expert system shells are also readily available, and there is prevalent legacy experience in the engineering of these kinds of systems for decision-making. Other features such as decision explanation, case-based reasoning, and machine learning can be easily value-added to a frame-based system.

Behavior Model

The Command Agent has a model of behavior which it uses to control its operation. These can be represented as constraints on its decision-making capability. These are usually defined by the higher-level controller of the command agent: a human operator or higher-echelon command agent.

These constraints can be represented in a variety of ways, but the easiest to implement would be as meta-rules within the inference engine. These rules would be differentiated from the static doctrinal rules by their dynamism. The implementation would be based on the capabilities of the expert system shell, but could include either newly written (marked) rules or existing rules that depend on defined data within the command agent's representation.

Advisor Arbitration and Control Knowledge

Arbitration and Control Knowledge defined for the command agent's advisors is of extreme importance to the design of the command agent. This component is where the command agent's knowledge of its advisors is stored. Utilizing the arbitration and control knowledge, the command agent is able to handle inconsistencies that may occur when two or more agent's produce conflicting hypotheses about the environment. In spite of the very careful compartmentalization of each agent's role in the simulation domain, agents can produce information that does not correlate either directly or by inference from that of other agents, especially as the system's components grow in complexity.

Arbitration and Control Knowledge also aid the command agent in directing the operation of its advisor sub-agents. This stores knowledge for the command agent to direct when and how the advisors should be utilized for efficiency. The behavior model greatly affects the utilization of this meta-knowledge for directing the decision making process. This knowledge could be stored as a procedural construct (e.g. a finite-state machine) in which advisor management strategies would be based on the context or node of the command agent. This provides a straight-forward and simple approach to handling advisor control and arbitration.

General Sub-Agent Interface

The General Sub-Agent Interface provides a layer of abstraction between the advisors and the command agent. The agent can call on the services encapsulated by the advisors through the sub-agent interface. The advisors also provide information to the command entity through this interface. This layer of abstraction allows the different pieces of the command agent to be functionally distinct but their combined behavior to act homogeneously.

A layered approach allows for independent and concurrent development of the interior processes of the advisors and the command agent. The different advisors and command agent can be tested without the other components being complete by simulating (stubbing) their behavior. The interface through which these pieces communicate must be maintained as well, but it provides flexibility required for the evolution of the design and implementation.

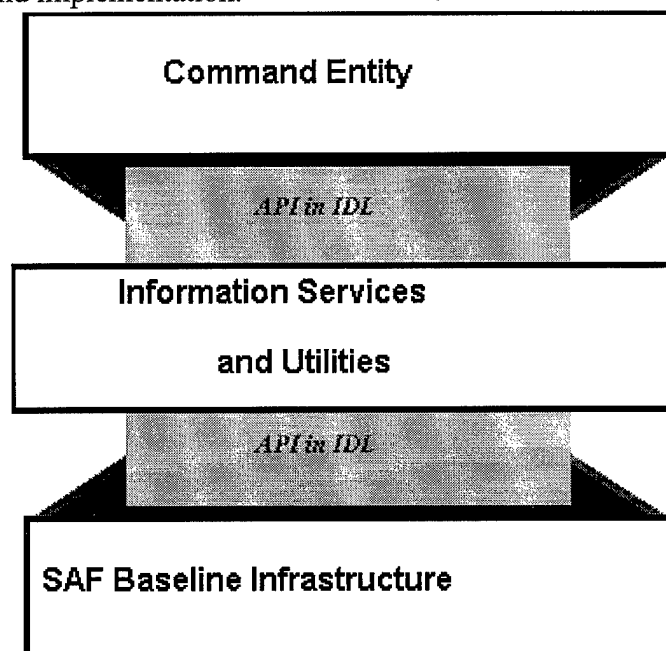


Figure 4: CFOR Reference Model

This layered approach can be seen in the technical reference model and implementation of the CFOR system []. Figure 4 shows the high-level layered model used as a basis for the CFOR design. There is a defined Application Programmers Interface (API), composed in an Interface Design Language (IDL), between the command entity application and decision processes, the information services and utilities, and the SAF baseline infrastructure (which in this implementation contains ModSAF and the CCSIL network protocols). This approach is extended in that advisor agents are the shepherds of a myriad number of interfaces and services and utilities that will support the command decision-making process.

ADVISORS

Advisors are sub-agents of the command agent that sift the extreme amount of battlefield data in a simulation and synthesize the data into information useful to the command agent's decision-making process. The agent's compartmentalized operation allows its internal process to be implemented using heterogeneous methodologies that are best-suited for that particular agent's specialized sub-domain of the battlefield environment. "In general, multiagent systems are computational systems in which several semi-autonomous agents interact or work together to perform some set of tasks or satisfy some set of goals." []

As can be seen from Figure 5 (next page), the advisor sub-agent consists of the following pieces:

- Interfaces
- Command Agent Directives
- Domain Knowledge
- The Reasoning Mechanism
- Provided Services.

The advisor controls the interfaces to components within the simulation that provide data and services. This can be terrain algorithms, C³ equipment, or vehicle primitives, etc. The advisor also receives and stores command agent directives that controls the advisor's operation. This can take the form of queries of data or services the command agent requests, constraints on the services of the advisor sub-agent, or tasks that the command agent has set for the advisor.

The domain knowledge element denotes the advisor's special capabilities regarding the context of its specialty. Domain knowledge can take many forms, some that encodes knowledge implicitly. Domain knowledge can be captured within rules, a training of a neural net, or within the cases of a case-base. "Agents could represent the same knowledge differently to optimize their particular use of it, or agents could obtain knowledge from different sources..." []

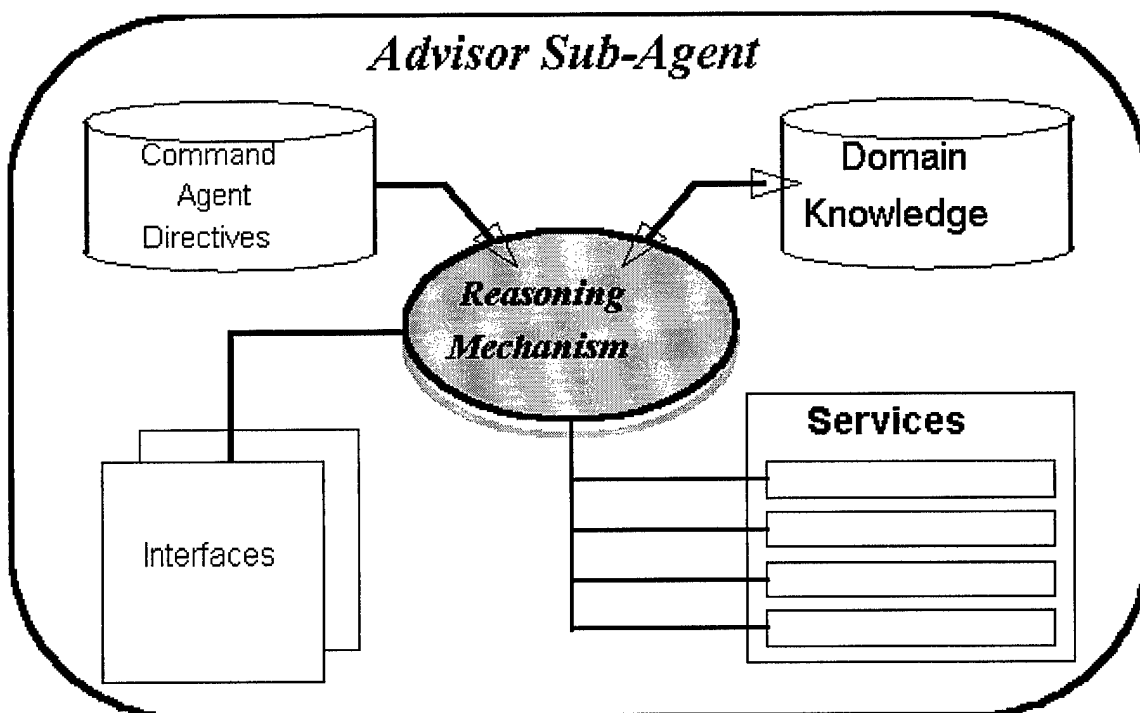


Figure 5: Advisor Sub-Agent Model

The reasoning mechanism of the advisor describes the operations that are performed utilizing the encoded domain knowledge. The reasoning mechanism could be a forward- or backward-chaining inference engine or neural nets in the form of the training and retrieval algorithms with which the net has been constructed. Case-based systems would employ key-matching and case adaptation as its reasoning mechanism.

Finally, the services that the advisor may be defined as part of its interface to the command agent. Data-driven techniques could be employed to allow the advisors to "register" its services with the command agent describing its capabilities and purpose. If the advisors conform to the protocol of the General Advisor Interface then this could facilitate an automated way of integrating altered or new advisors to the command agent's repertoire. This concept is a smaller-scale implementation of the ideas set down in the High-Level Architecture (HLA) specification.

COMMAND AGENT IMPLEMENTATION

Employing the layered concepts above allows the implementation of the command agent to take many forms while retaining the same decision-making process described in the SHOR framework. The flexibility of the design allows the implementation of command decision making agents to take any form which is appropriate to the problem domain. The developer is constructing an ontology of SHOR components. The following sections describe examples of command agents that may be constructed for different echelon levels and different strategies of conducting decision-making.

JUDGEMENTAL METT-T COMMAND AGENT

Command agents must be capable of rapid decision-making in constantly changing domains. Knowledge acquisition to apply to automated decision-making is of great difficulty because of the lack of canonical literature. To aid in this process the command agent can be modeled around doctrinal epistemology.

Situational awareness is paramount to the a battlefield commander. Army doctrine captured from Subject Matter Experts (SMEs) and training literature promotes the acronym METT-T to describe a process for an army commander to analyze the battlefield environment. METT-T stands for Mission, Enemy, Terrain, Troops, and Time, and it is utilized at all levels of the U.S. Army command structure.

As can be seen in Figure 6 (next page), the construction of the Command Agent's advisors corresponds with these subjective areas. As described above, the advisors act as the command agent's proxies and filters to the information sources that exist inside and outside of the simulation environment.

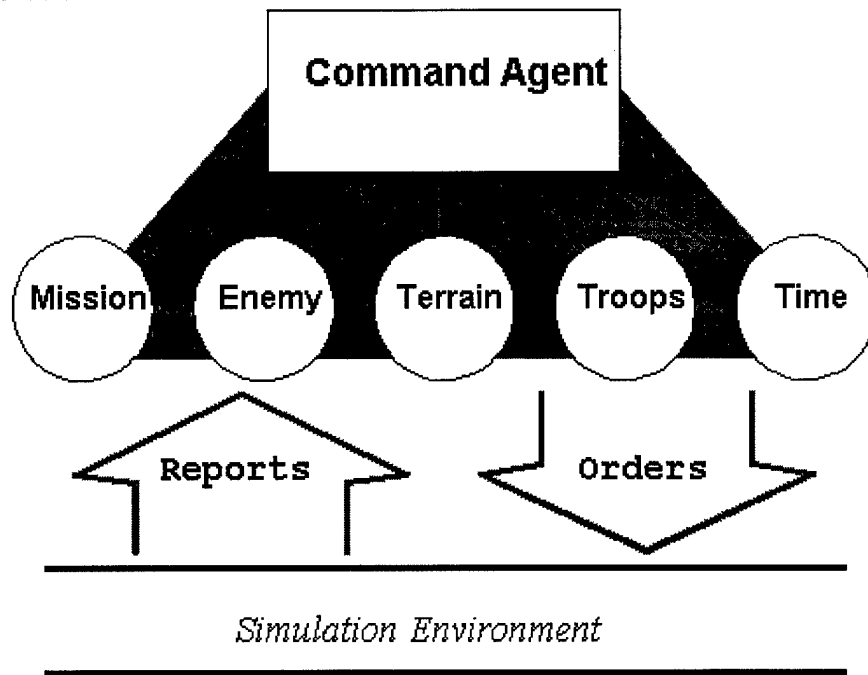


Figure 6: Command Agent with METT-T Advisors

For example, the Terrain Advisor controls the interface to the terrain database of the battlefield. The Terrain Advisor has rational methods of analyzing this data and deriving knowledge that is useful to the Company Team Command Agent. The other Advisors have similar interfaces to other information sources necessary to conduct their specialized purposes. The strength of this approach is that various AI techniques may be encapsulated within these Advisors providing a flexible, extensible framework in which to construct hybrid systems. On top of this design is the Command Agent that acts as arbiter and director of the outputs of each advisor's analyses.

A detailed description of this model as applied to a Company Team Command Agent utilizing METT-T advisor sub-agents can be found in [ii]. Table 1 shows some possible AI techniques that may be applied to the different areas of METT-T.

Table 1: Techniques applied to METT-T

Mission	<i>Objectives-based Planning</i>
Enemy	<i>Case-Based Reasoning (CBR)</i>
Terrain	<i>Neural Nets</i>
Troops	<i>Rule-based Systems</i>
Time	<i>Temporal Reasoning</i>

BOS COMMAND AGENT

The Battlefield Operating System (BOS) is a large connection of computers and intercommunication components that is used to disseminate and fuse large amounts of battlefield information. The BOS is divided into several different areas of command and control that is under a high-level echelon commander's purview. The example, shown in Figure 7, (next page) is meant to illustrate how a domain model of decision-making can be constructed for command agents that differs from the one presented above.

The same SHOR framework (see above) concept is employed, distributed through the different advisors. Data flows from the simulation environment and spawns one or a concurrent number of stimuli within the advisor environ. The advisors generate hypotheses that are incorporated into the BOS Command Agent's internal representations. The BOS Command Agent determines how to use these hypotheses. It can call on advisor services in order to make a decision or continue to collect information.

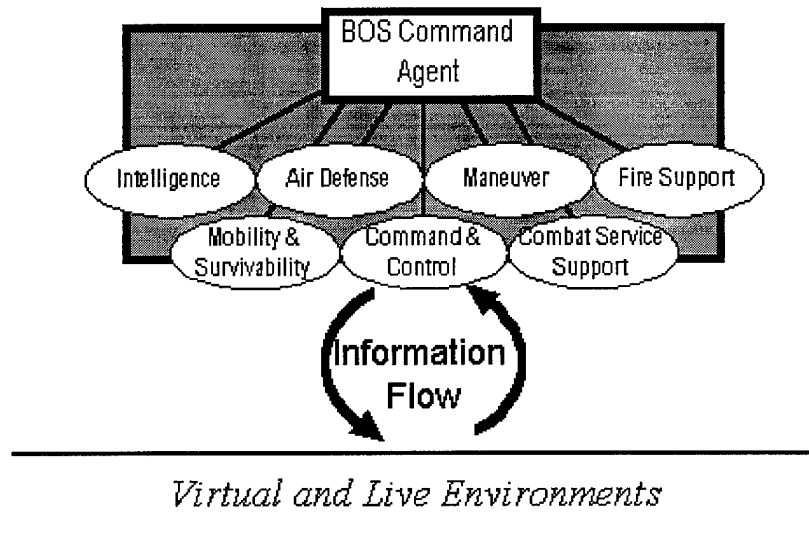


Figure 7: BOS Command Agent with Critical Combat Function Advisors

The advisors specialize in critical combat functions necessary to particular areas of a military operation. The significant aspect of this engineering structure is that each advisor controls the information flow from one channel implemented within the BOS. The interfaces to these different parts of the BOS are encapsulated within each specialized BOS advisor sub-agent. In this way, the construction of the command agent can be incremental and only concern specific aspects the designer wishes to implement. This design also shows how this approach provides an easy facility to integrating with existing military systems while employing hybrid AI techniques.

COMMANDING OFFICER AGENT

Commanders from Battalion and above have staff positions held by subordinate officers that are responsible for different areas of an Army organization. At the battalion and brigade levels these positions are called Staff (S) positions. At division to army level they are held by General (G) Officers. The positions and their designations are described in Table 2.

Table 2: Staff/General Positions

Designation	Position
S/G 1	Personnel
S/G 2	Intelligence
S/G 3	Operations
S/G 4	Logistics
S/G 5	Civil Liaison

The S/G1 (Personnel) handles discipline, awards, and assignments. This position assures that men are available and placed properly within the organization, whether it is a battalion or an army.

The S/G2 is the intelligence officer responsible for security, counter-intelligence, and the Intelligence Preparation of the Battlefield (IPB). The IPB is basically the proper presentation of mission materials to indicate where enemy emplacements are believed to be and how the enemy is expected to behave.

The S/G3 is the highly important operations officer. His responsibilities encompasses the planning of the movement and disposition of all forces in the organization. This includes training, detailed mission plans, and movement of all equipment, men, and their support.

The S/G4 deals with the logistics of his unit. He facilitates the supply and maintenance of all things the organization will require.

The S5 is a position that is generated based on mission requirements. This position manages the politics, public relations, and interface to civil authorities sometimes required by Operations Other Than War (OOTW), such as humanitarian aid. At higher echelons this position becomes a necessary part of the organization otherwise it is filled as needed.

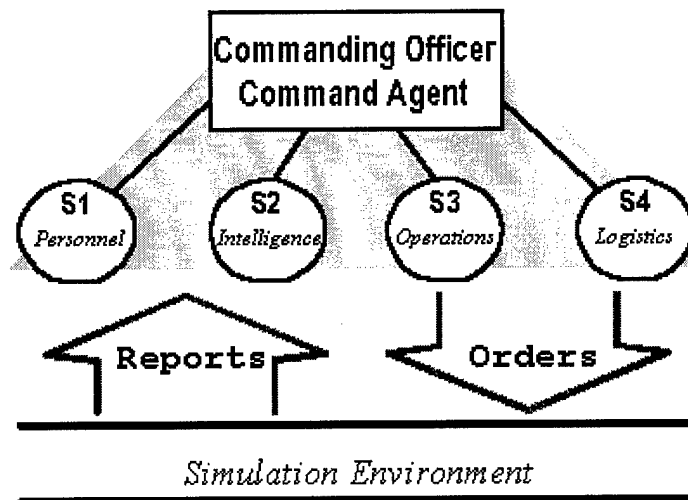


Figure 8: Commanding Officer Agent with Staff Advisors

Figure 8 shows the organization of the command agent architecture based on the commander and staff model. This shows the advisors being divided along the staff specializations: personnel, intelligence, operations, and logistics. The civil liaison duties would not commonly be required in a battle simulation. Figure 8 again shows the common theme of information flowing to and from the simulation environment which results in command behavior interaction with the simulation entities.

COMMAND AGENTS AT MULTIPLE ECHELONS

An interesting aspect of this problem is the creation of multiple levels of command agent interaction along echelon levels. The developer can create a hierarchy of command agents that act as both advisor and command agent. However, creating a stovepipe connection between command agents, as in Figure 9, can be detrimental to efficiency. This is true for real life military operations where informal, non-doctrinal communication relationships arise to assure the effective accomplishment of objectives, as in Figure 9.

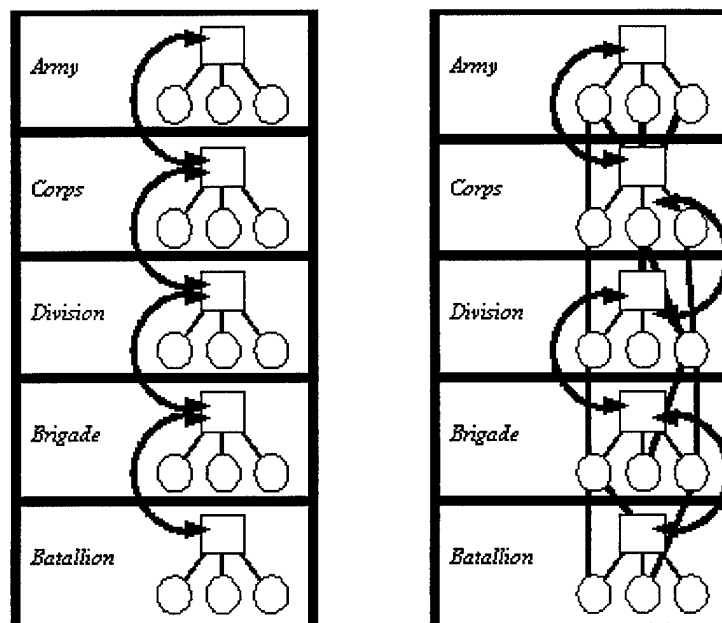


Figure 9: Multi-echelon Command Agents

CONCLUSION:

The implementation of a command agent must be designed for evolution in the changing simulation environment in which it will be employed. This requires a special architecture that can provide flexibility and scalability. The architecture described above can be used to design a command decision-maker that accounts for the highly dynamic domains of a simulated battlefield environment, but also allows for the implementation of heterogeneous strategies for reasoning about the environment. This architecture also allows for different models of command decision-making to be implemented. The proper implementation of this architecture can develop a command agent that functionally acts as a surrogate for a human commander in a military simulation.

[Return to the Top of this Section](#)

[Return to the Section 2 Index](#)

[Return to the Table of Contents](#)

[Return to the SCC Command Decision Modeling Page](#)

[Return to the WARSIM 2000 Page](#)

[Return to the NSC Home Page](#)

COMMAND DECISION MODELING
TECHNOLOGY ASSESSMENT

TECHNOLOGY AREA ASSESSMENTS:

1. RULE BASED SYSTEMS
2. FUZZY LOGIC
3. CASE-BASED REASONING
4. GENETIC ALGORITHMS AND
EVOLUTIONARY PROGRAMMING
5. NEURAL NETWORKS AND BOUNDED
NEURAL NETWORKS
6. LATTICE AUTOMATA
7. AI PLANNING SYSTEMS
8. PETRI NETS AND COLORED PETRI NETS

TAA1

TECHNOLOGY AREA ASSESSMENT:

RULE BASED SYSTEMS

ASSESSOR:

Christopher Dean, M.S.Cp.E.

Science Applications International Corporation
3045 Technology Parkway
Orlando, Florida 32826-3299
(407) 282-6700
deanc@orl.saic.com

OVERVIEW OF THE TECHNOLOGY AREA:

DESCRIPTION OF THE TECHNOLOGY:

Rule-Based Systems, originally known as Production Systems and later as a subclass of Expert Systems, are outgrowths of early problem solving research in Artificial Intelligence (AI). These systems are characterized by a general, underlying idea: problems in well-understood domains can be solved by structuring domain knowledge into IF-THEN rules by experts in the field. As the field of AI matured, techniques for encoding knowledge evolved into a second generation of expert systems which combines multiple knowledge representations and problem solving strategies within a single system. These systems are now known as Knowledge-Based Systems (KBSs). The underlying ideas for KBSs are: knowledge is central to problem solving and different models and problem-solving methods are needed for different aspects of the problem.

Encoding knowledge into rules is the most common form of knowledge representation for KBS [Gonzalez and Dankel, 1993]. One reason rule representation is popular is due to its simplicity: a rule is composed of an antecedent (IF condition) and a consequent (THEN clause). Rules can express a wide range of associations, such as situations and corresponding actions that are taken when these situations occur, and as premises and conclusions that result when those premises are true. Rules generalize relationships among objects of the domain. Facts about specific instances of objects are stored in a fact database and rules are evaluated against these facts. This representation of knowledge allows various types of knowledge to be encoded, for example:

1. facts: "A tank platoon is a platoon unit."
2. information rules: "A platoon unit is commanded by a platoon leader."
3. hard-and-fast or procedure rules: "Always stop the vehicle before dismounting DI units."
4. problem situation rules: "If the unit arrives at the destination early, then halt until time to depart."

A second reason for the popularity of the rule-based paradigm is that encoding domain knowledge into rules is similar to how human experts communicate their knowledge. This facilitates the acquisition of knowledge required to build a rule-base. Multiple experts can be interviewed and their expertise can be pooled and integrated into a repository of relevant knowledge. Analysis of legacy systems and written documents can also be used to acquire the necessary knowledge. In this way, a rule-base is built incrementally. However, the ease of knowledge acquisition is deceptive. Integrating the knowledge into a rule-base that can perform as well as a typical domain expert can be accomplished only with skillful knowledge engineering. That is, by focusing a domain so that the knowledge is specific, by writing and maintaining rules in tight chunks or sets of knowledge, and by editing and continually refining the knowledge base a successful KBS implementation can occur.

Essentially, a rule-base is invoked by presenting the KBS with a specific problem description or case, and by the KBS searching through its knowledge of rules and facts for an answer. The mechanism used to draw conclusions based on rules in the knowledge base and the data for the current case is the KBS's reasoning process or inference strategy. The inference strategy specifies the order in which the rules will be compared to the knowledge base and a way of resolving the conflicts that arise when several rules match at once.

The two strategies that control the sequence of rule firing are forward chaining and backward chaining. The forward chaining method of inferencing is a reasoning strategy that is also known as data driven, event driven, or bottom-up thinking. Starting from the known facts, rules are fired that derive new facts which in turn activate other rules thus forming an inference chain from the source state to the goal state. A rule is fired by matching the antecedent against the current fact-base to act out the consequent. The idea behind forward chaining is to find a sequence of rules that when given the data for a current case and known facts will lead to an answer. This approach is useful when initial facts are known about a problem situation and there are many possible goal conclusions. Problem domains, such as, design,

configuration, planning, scheduling, and classifying are good candidates for the forward chaining strategy.

The other fundamental reasoning strategy is backward chaining. This strategy is also known as goal driven, expectation driven, or top-down thinking. It requires looking at the consequent or action parts of rules to find ones that would conclude the current goal, then looking at the left-hand sides or condition parts of those rules to find out what conditions would make them fire, then finding other rules whose action parts conclude these conditions, and so on. Basically, this method starts with the desired conclusion or answer to a problem statement and decides if the existing facts support the derivation of a value present in the knowledge base. Backward chaining is useful in domains where there are many facts that pertain to a problem situation but not all are necessary for deriving the solution. Domains such as diagnosis are well-suited for backward reasoning. Tactical decision-making also seems well suited to a backward reasoning paradigm since military behavior is often expressed in terms of goals to be achieved [Kwak, 1995]. It can also use forward reasoning when opportunistically using a given situation.

With either inferencing strategy, new facts are derived as solutions are sought. Once a new fact is derived, it remains true and becomes a member of the knowledge base. Adding facts permanently to the knowledge base is called monotonic reasoning. However, some systems allow new facts to be created by making assumptions based on currently known facts. These derived facts have an amount of uncertainty associated with them. This is typical of real-world problems where assumptions are made or defaults are taken based on the context or pragmatics of the current problem situation. New facts can arise that contradict previously asserted facts. The ability of the system to retract conditionally asserted facts is called nonmonotonic reasoning. Retracting of a fact may necessitate the removal of other, dependent, facts. Truth Maintenance Systems have been developed to maintain the integrity of the knowledge base when nonmonotonic reasoning is used. Problem domains, such as classification or diagnosis, can be implemented with monotonic reasoning. For more complex problems, and hence real-world problems such as command reasoning, nonmonotonic reasoning becomes a requirement.

Choosing forward or backward chaining is just one component of the inferencing strategy. The other component is deciding how rule selection conflicts will be resolved. Backward chaining systems generally use depth-first backtracking to select individual rules. Forward chaining systems generally employ sophisticated conflict resolution strategies for selecting among the applicable rules. Examples of conflict resolution strategies are: the first rule that matches, the highest priority rule, the most specific rule, the rule that has not fired before, an arbitrary rule, and firing rules in parallel. The most straight forward and easiest to implement is the first rule that matches strategy. Other strategies raise the complexity of rule creation and reduce the rule comprehension. The choice is not a research issue for command decision modeling but purely a development tradeoff.

The fundamental disadvantage of rule-based systems that affects command decision modeling is that as the knowledge base grows in size the likelihood that the system is still performing as an expert decreases. Complex, real-world domains like command decision modeling are knowledge intensive and require lots of detailed knowledge. Decisions are multiply constrained by knowledge from various sources, such as mission, terrain, enemy, troops, and time requirements. These sources are in themselves complex and difficult to model. Modularizing the knowledge base into rule sets increases the ease with which the rule-base may be modified and validated but restricts the complexity of the problem being modeled. The small rule sets can be thought of as experts that can handle detailed, specific problems but cannot efficiently solve problems together since they cannot communicate. This leads into the research area of autonomous agents: local experts that need to learn to work together.

Interest in rule-based technology has reached an equilibrium point for the most part as the paradigm moves from a research area to more practical implementation areas. Rule-based systems often serve as the foundation of more complex and/or hybrid reasoning schemes [see for example, Mall et al., 1995; Chaib-draa et al., 1993; Gonzalez and Ahlers, 1995]. This has led to the development of knowledge based systems that are characterized by explicit modeling of different types of knowledge and by the use of a variety of problem solving methods that are geared toward particular subtasks.

NOTABLE IMPLEMENTATIONS AND VARIANTS:

RPD Architecture

An early implementation of rule-base reasoning for command-level tactical decision-making is the RPD architecture [Chaib-draa et al.; 1993]. It is a hybrid architecture design to facilitate the coordination of intelligent agents to promote beneficial interactions and avoid harmful ones. This becomes an important issue since local decisions many times do have global impacts. The architecture is composed of three primary components: a reactive component, a planning component, and a deliberative component, hence the name RPD. The rule-based knowledge representation paradigm is used to encode knowledge for the reactive component that handles the reactive agent behaviors and the planning component that handles planning in the presence familiar situations. The deliberative component handles deliberative decision making for completely unfamiliar environments and does not employ the rule-based paradigm and thus will not be discussed further.

Under the RPD architecture, information from the environment is perceived by an intelligent agent. A planning action occurs if the information is structured as a goal, otherwise a reactive action occurs. Rule patterns specify reactive actions to the environment and the specification of goals. Whenever a situation cannot be handled by these rules, identification and recognition agents are needed. Furthermore, if the information is ambiguous or a goal needs elaboration due to an unfamiliar environment then deliberative decision making is performed. Once unambiguous, specific goals have been identified, planning can begin.

Planning is closely linked to perception, identification, and recognition. Perception either leads directly into planning or is first followed by identification or recognition before it leads to planning. Planning is followed by an action. The planning component always deals with familiar situations. The recognition process uses rules to recognize a certain goal or task. The rules for recognition adhere to the following format:

IF < signs> AND <situation> THEN <accomplish-goal>

or

IF < signs> AND <situation> THEN <execute-action>

Signs and the situation clauses in the IF-AND portion of the rule are the facts that define the current state of an agent's world. When signs become active for a given situation, a goal becomes active or an action is executed. The identification process uses Case-Based Reasoning which is discussed in the CBR section of this paper. Once goals have been identified, the identification process tries to determine the actions necessary to achieve those goals. The planner instantiates templates for plans to achieve those goals based from a set of plans issued from familiar situations. These templates may also be modified to fit the specific goals.

This form of rule execution is known as rule-based planning. It assumes that the state of the world at a given time is represented by a set of facts. The planning is characterized by a set of preconditions which are logical formulas expressed in terms of the facts that must be true before an action can be successfully executed. The effects are the set of formulas that will be true after the action has been successfully executed. The body is the set of subactions to be performed or subgoals to be achieved [Chaib-draa et al.; 1993].

To reach a specific goal state, the agent will execute rules that build a plan to achieve the goals. Case based planning (CBP) is used for unfamiliar situations. Two reasoning approaches are used within the rule-based system to construct a plan. Forward chaining is used to decompose actions into more primitive actions. Primitive actions are easier to use for creating different plans. Backward chaining is used to ensure that the preconditions of each action are satisfied. If any of the preconditions is not satisfied and not generated by some previous action in the plan, then another action is found that generates the precondition and includes it in the plan.

The RPD architecture seems well suited to a hybrid command reasoning architecture. However, no implementation details are given suggesting that the architecture has never been developed beyond the initial stage. It does serve as a possible blueprint for command reasoning systems of the future.

Interactive Tactical Management System (ITEMS)

An early use of expert systems for Computer Generated Forces (CGF) is the Interactive Tactical Management System (ITEMS) [SikSik, 1993]. It provides an environment in which entities can interact with each other and the environment. The entities are modeled with respect to their underlying physical system and their intelligence. ITEMS supports, using rules, mission planning, opponent selection, and coordination within and between different echelons. For command and control, ITEMS supports company and battalion level commanders. The company level behavior is primarily characterized by platoons cooperating for a single purpose. The battalion level behavior is characterized by more complex coordination of forces with companies possibly cooperating on a task that does not directly benefit them.

Each echelon command level has an associated Command and Control Unit (CCU). The CCUs are templates that enable the assignment of tasks to each of the units and establish the chain of command. Tactical overlays can also be referenced by a CCU during scenario definition. This provides the unit with an initial position as well as a mapping appropriate to its command level. A set of doctrine is also associated with each CCU. The Company Coordination Doctrine provides the knowledge necessary to perform platoon coordination behavior. Typical coordination parameters include: messages and orders from the battalion commander, company survivability and kill indices, threat positions and status, company position, and mission information such as the current mission and the state of completion. Company behaviors encoded in rules are governed by doctrine according to the tactical situation [SikSik, 1993]. Example ITEMS company behaviors include: fire positioning, evasive maneuvering, withdrawing, halting, and finding of alternative routes.

The Battalion Coordination Doctrine provides the knowledge necessary to perform company coordination behavior. As an example of coordination behavior, a hasty attack behavior sequence is as follows [Siksik, 1993]:

the battalion commander assigns, according to its doctrine rule base, a fire-base company and maneuver company

directs the fire-base company to move to a fixed location relative to the enemy and provide suppressive fires

directs the maneuver company to outflank the enemy

Thus, planning is static in the sense that only predefined plans (via rules) are used and are not dynamically created from individual actions. The parameters considered are high level and include: company positions, company status and the battalion commander's orders to the company. The ITEMS battalion behaviors include: coordination of attack and maneuvers (e.g., hasty attack), artillery support, and resupply.

As mentioned previously, the knowledge base is written in the form of rules. The authors contend that as far as functional "how-to" knowledge is concerned, rules are very close to how experts organize their actions or ideas [SikSik, 1993]. The rule-base uses doctrine as its knowledge source and is organized into modules by echelon and internally by a tree structure hierarchy of rule sets. These rule sets partition the knowledge base into manageable units that describe a particular state or belief as appropriate to the behavior being represented. This limits the search space that a command agent has to use to decide upon an action.

The rules within ITEMS are based upon condition parameters calculated by the ITEMS control objects. Rule actions are divided into specific action categories, each of which is mutually exclusive. ITEMS uses forward chaining inferencing strategy since the state of the condition parameters is calculated and

known every real-time iteration and since the number of possible responses is large. The conflict resolution strategy used is to pick the first rule that matches. This strategy is utilized because it is the most straight forward.

ITEMS has shown that rules can be used for CGF behavior. However, ITEMS exhibits the shortcomings of the expert system approach. High level behaviors are brittle, that is, when they are faced with a situation slightly different from the ones the rule-base was designed to handle, the rule-based approach fails. Since ITEMS cannot make decisions for situations it was not specifically design to meet, decisions within ITEMS are static and hard coded. This is not flexible enough for real-life combat situations. If enough doctrine is encoded into the rule-base, then ITEMS may be adequate for military training. Adding to the knowledge base leads to the another expert system shortcoming: a large knowledge base is difficult, if not impossible, to maintain and validate. This is why rules are often used in conjunction with other knowledge representation paradigms and problem solving approaches.

An important aspect of command decision modeling is planning. In ITEMS the mission planning aspect is alluded to but not really discussed, suggesting that this type of planning is not really addressed. A system that cannot handle novel problem situations and lacks planning is not robust enough to handle the command reasoning problem on its own. ITEMS may be modified to serve as a component of a hybrid system or as a foundation for a more complex expert system.

Concurrent Control and Coordination

Research has also been performed on investigating the use of concurrent control and coordination techniques for higher echelon commanders [Harmon et al., 1994]. These techniques are typically used for low level reactive behaviors. The goal of the research is to determine if concurrent control techniques can effectively be applied to upper echelon commanders.

A concurrent control system is composed of several individual control loops each working towards its own goal. As much intelligence as needed can be placed in these control machines. Rules are used to arbitrate and select the behavior to execute. Thus, various aspects of nonlinear control can be realized. The rule-based component becomes more valuable as more control loops are added to the system. This approach enables modularization of interacting control laws and limits the complexity to a tractable level.

The concurrent control methodology is based on the premise that multiple simultaneous goals must be evaluated in the execution of a mission [Harmon et al., 1994]. Here, each goal represents a single behavior that attempts to optimize its own performance independent of the other goals. An arbitration scheme resolves the competing or conflicting goals and either chooses a goal or combines the goals into a single goal to execute. Since the system is responding to all the active control loops, it is always responding to the entire pertinent situation and, thus, does not become trapped in an inappropriate control state or local minima [Harmon et al., 1994].

For command reasoning, a concurrent control commander evaluates the status of its task according to its orders and generates the necessary low level goals for its subordinates. Each individual task is mapped onto a concurrent control module. These tasks correspond to mission tasking, reporting, and command transfer. Individual control modules are also needed for each of these tasks to represent multiple concurrent missions and reporting tasks. In addition, separate arbiters are designed to interface to the different subordinates to ensure consistent commands are sent to all the lower echelons.

Concurrent control does show promise for dealing with the enormous number of possible combinations of situations that can occur during training exercises. The results are positive for limited scenarios such as movement to contact, attack by fire, and defend from ambush. Coordinated activity is demonstrated as well as transferal of command when a commander is disabled or killed. However, cognitive abilities such as situation assessment and planning were not addressed. For command reasoning, situational assessment and planning are requirements. The rules were used only to arbitrate the control and thus were not the focus of the research. In addition, the research effort primarily demonstrated the applicability of concurrent control to the reactive side of a command agent. Rules were also

meaningfully applied to the plan monitoring and resource allocation problems associated with command entities.

Close Combat Tactical Trainer (CCTT)

The Close Combat Tactical Trainer (CCTT) simulates behavior from the lowest level vehicle behaviors through battalion. A rule based system was used for these behaviors in the CCTT SAF prototype [Ourston and Bimson, 1994; Bimson et al., 1994]. The decision to use a rule-based system for tactical decision making was based upon an analysis of the reasoning requirements (rules are suited to this kind of reasoning), extensibility, modification flexibility, and the need to isolate the reasoning component from the more algorithmic processes of platform behaviors and terrain analysis [Ourston and Bimson, 1994]. The SAF prototype focused on tactical decision making at the platoon level. While this is not useful for autonomous command agents, the approach can be scaled up to higher echelon behaviors. The prototype's purpose was to only illustrate the use of a knowledge base in making tactical decisions and arbitrating among competing alternatives of behavior from different sources.

It is a generally accepted conclusion that expert systems are too slow in reaching conclusions and, therefore, are not viable approaches for modeling real-time behavior. One reason for this conclusion is that real-time expert systems, such as RTworks, are agenda based systems. A system with an agenda is one where rules continue to fire until all potentially satisfiable rules have been tried and no new rules have been activated as a consequence of another rule firing. Rule processing time can vary from a few milliseconds to a few minutes depending upon the number of rules on the agenda. Another reason for slow execution speed is the complex pattern matching that expert systems such as RTworks perform. The SAF Behaviors prototype models entities and tasks as objects with slots for data values. Rules match on these slots as a precondition for rule firing. It is not known what rules will fire or in what order. With the addition of wildcards in the rule patterns, a great amount of rule processing can be performed, making the execution time difficult to predict. Algorithmic languages such as C or Ada could be used to execute the rules increasing the execution speed but at the cost of flexibility, maintenance, and increased development complexity.

The Behaviors prototype deals with mission, enemy, terrain, and troops over time (METT-T) concerns. Decisions concerning the who, when, where, why, and what of behavior are assigned to the rule-based behavior component while the how is contained in the algorithmic vehicle behaviors module. The execution problems of expert systems mentioned earlier is not as significant for tactical decision making since the response time requirements are less stringent, often in the range of seconds or even minutes. Tactical behaviors are characterized as more knowledge intensive but occurring less frequently than lower level platform behaviors such as moving a vehicle which is algorithmically modeled and real-time.

Most of the tactical decision making in the Behaviors prototype is done in response to situational interrupts. It is here that METT-T is addressed. The prototype only addressed Enemy and Time aspects of METT-T. Behaviors implemented include detection of enemy forces, direct small arms fire, direct antitank fire, indirect fire, fixed wing air attack, rotary wing air attack, defiles, and open terrain movement. The rules that respond to the situational interrupts are prioritized by threat and can be nested based upon this priority. The rulesets also detect termination conditions that allow an interrupted task to be continued. It should be noted that the prototype allows the human SAF operator to handle the more complex behavior via command overrides to the modeled entities.

The rules are organized into sets of modules for different echelons. Each module includes rules that represent and execute: orders, reports, command overrides, situational interrupts, and arbitration of behavior alternatives. Rules are further organized by category (movement, formations, reactive behaviors, defensive, offensive) and by Combat Instruction Set (CIS). Ultimately, rules were not used for the development system, not because they were not efficient enough or a close fit, but because knowledge engineering is still a research area and the project was under a tight development schedule. For behaviors guided strictly by doctrine, such as required to represent adequate training in CCTT, rule-based systems are appropriate.

In the CCTT SAF Prototype, rules were shown to be effective for platoon level tactical decision making, especially for multi-contingency situational awareness in which competing alternatives must be resolved. The rule structure closely resembles the kind of reasoning that is typical of tactical decision-making [Bimson et al., 1994]. The modularity of the rules was shown to be important for real-time execution. Most importantly, it was shown that the rules were not a bottleneck for system execution. The rate at which rule sets must execute and the rate at which situational interrupts occur during an exercise are relatively low and thus do not impact the overall execution time [Bimson et al., 1994].

One of the results of the SAF prototype effort was that it shed light on the representation and reasoning of METT-T. There is a difference between the behavior of command agents, typically judgmental METT-T, and event-driven METT-T [Bimson et al., 1994]. Judgmental METT-T data and behavior representations are more complex and performance intensive. Event-driven METT-T is the behavior typically characterized by the situational interrupt and thus was the concern of the prototype. Judgmental METT-T behaviors were not addressed by the rule-based system because of their complexity. Whether or not the rule-based paradigm was appropriate was not addressed.

Rational Behavior Model (RBM)

The Rational Behavior Model (RBM) has recently been proposed as the representation for CGF behavior [Kwak, 1995]. RBM is a backward reasoning representation that can scale up to the higher echelon behaviors necessary for command agents. It is composed of three levels: strategic, tactical, and execution levels. Behavior control logic is located in the strategic level and controls the operation of the tactical level. The tactical level maintains the behavioral attributes for the system and represents the internal behaviors of the strategic level. The behavior interface is located in the execution level.

The strategic level is composed of backward chaining rules that form an AND/OR goal tree. The rules are written in Prolog without asserting any facts, only backtracking memory is used. The priority between valid goals is defined implicitly by the rule order. There is also a node priority for sibling sub-goals. Both priorities are statistically assigned a priori and together are used as the conflict resolution strategy when equally possible sub-goals are encountered. AND sub-goals have the same priorities but are of less significance since only OR sub-goals can potentially be conflicting. That is, more than one sub-goal might be eligible for a given condition. AND sub-goals are a sequence to achieve. If any AND sub-goals fail, the parent goal fails. Thus, multiple sub-goal activations cannot be allowed to occur.

The author contends that RBM is a natural representation scheme for military behaviors since it is well developed and organized. Military doctrine effectively expresses the global objective and sub-objectives and their associated actions and priorities. The author also states that the Army FM has effectively written the backward reasoning representation even though it gives the impression of a time-line based description [Kwak, 1995].

For command reasoning, the RBM provides the better domain match to target behavior description (FM or SME) [Kwak, 1995]. Application of the RBM to higher echelon behaviors is not mentioned in any detail. The goal-directed viewpoint is promising for higher echelon behaviors but the backward reasoning approach presented here seems to require all the knowledge a priori. There is no explicit support in the RBM for uncertainty, intentions, and partial fulfillment of goals that are necessary for autonomous command agents.

Context-Based Reasoning (CxBR)

A recent use of the rule-based paradigm for autonomous agents is using rules to represent contexts. The use of contexts overcomes some of the deficiencies associated with case-based reasoning, namely the requirement for a large case base containing specific cases. Context-Based Reasoning (CxBR) is an attempt to overcome these deficiencies [Gonzalez and Ahlers, 1995]. CxBR uses the concept of scripts to provide tactical knowledge to intelligent agents on the battlefield. These scripts provide the 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 CxBR 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. This work does not explicitly address the upper echelon behaviors that command reasoners need. However, the work's general architecture is applicable.

Because the contexts are 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. CxBR can be used to define behavior at the mission level (mission contexts) and at the task level. Since the domain is narrow and specific, CxBR 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.

The CxBR approach works under some of the following assumptions [Gonzalez and Ahlers, 1995]:

- Decision making is continuous and is heavily influenced by the continuous sequence of contexts. Each active context regulates the behavior of the agent and provides an expectation for the future. Contexts change as a result of external events and actions made by the agent. A context can be described as a well Understood situation that has a predetermined set of procedures that must be carried out, either sequentially, methodically, or arbitrarily.
- Only one context is active at one time.
- Contexts are represented as intervals of time rather than time points. Contexts may be considered to be transitions to reach a goal or a goal themselves.
- Goals can be time points, but only to serve as transitions to other contexts.
- Only a limited number of things can take place in any single context.
- Certain cues exist which will indicate that a transition to another context is desirable.
- The presence of a new context will alter the present course of action and/or the applicable expectations to some degree. Using rules, the potential contexts and corresponding actions are associated with specific situations, simplifying the identification of a situation because only a subset of all possible situations is applicable. Similarly, the course of action to choose is easier under the recognized situation.

The CxBR approach is composed of a mission context, major-contexts, and sub-contexts. The mission context defines the objectives, constraints, and problems to avoid in the mission. It is usually defined in terms of lower level contexts and describes the political environment under which the mission is to be carried out. Major-contexts contain behavioral knowledge and context transition knowledge in the form of rules. Each major context is an individual task that is necessary to meet the overall goals of the mission. A context is activated by retracting from the fact base the identifier of the current context and asserting the identifier of the new context. The initialization procedure of the new context is also called which in turn activates new context monitoring rules. Any agent attributes (parameters) may also be modified if necessary. The context is set up as a class that can be used for pattern matching in a rule-based system. The context contains initializer functions, an objective slot, a compatible-next-major-context slot, and a compatible-sub-context slot. The objective slot is stated in general terms and references the frame that has attributes that are the goal of this context. The Compatible-next-major-context attribute contains the allowable context transitions. The Compatible-sub-context lists all the sub-contexts that are compatible with the current context.

As mentioned previously, the recognition of the situation is done through pattern matching rules. The use of active context pattern in the premise modularizes the active rules to make the inferencing more efficient by reducing the possible action space. Monitoring rules handle the transition from one context to another. They fire continuously as long as the context is active. They monitor parameters which are

relevant to the continued execution of the current context. When the actions of a monitoring rule are executed, the current context identifier is changed and the new context identifier is asserted along with any initialization.

CxBR is an effective and efficient mechanism for imparting sufficient intelligence to intelligent agents to achieve training objectives [Gonzalez and Ahlers, 1995]. Sufficiency is defined by the ability to behave as a human enemy would in a very specific and narrow domain. CxBR can be used to accurately represent tactical behavior from a qualitative standpoint [Gonzalez and Ahlers, 1995]. An important conclusion is that through the prototype, CxBR was shown to be compatible with a distributed simulation environment. This is an important consideration for military command reasoning.

CxBR is done in the rule-based paradigm but is not a pure rule-based mechanism since it contains objects and context identifiers. Through experimentation it was shown to be slightly faster than a pure rule-based implementation. As the size and scope of the domain expand this will have more of an effect. For a limited prototype, the pure rule-based approach is more concise because of the CxBR overhead of the context objects. As the situation becomes more complex however, the overhead will become a smaller part of the knowledge base. For the rule-based approach the rules required become more numerous, complex, and inter-related.

CxBR has the advantage of encapsulating all the facets of tactical knowledge for a small slice of the domain. By modularizing the knowledge in this way a more efficient execution can be achieved. Also limiting the number of possible transitions between contexts is important for execution

The prototypes used in this research are primarily reactionary in nature and thus the planning capabilities were rather limited. Nevertheless, from a conceptual standpoint, planning is quite consistent with the general CxBR approach. Thus, the CxBR approach shows promise for scaling up to higher echelon behaviors.

APPLICABILITY TO COMMAND DECISION MODELING.

Depending upon training objectives, a rule-based paradigm may not be the best choice for command reasoning due its brittleness in domain coverage. Situations in the domain not covered by the rules cause the command reasoning to fail. This is definitely a limitation of the paradigm since tactical decision-making involves dealing with an infinite combination of situations, such as number, types, and positions of entities, weather, and terrain. A more promising approach that has recently shown promise is the use of *intelligent agents* in support of command decision making. These consist of relatively small rule-bases that are used to solve well-defined and constrained subproblem areas. These intelligent agents contribute to the problem solving process as the conditions that apply to their domain expertise are met. In this way, the problem solving process departs from the monolithic architecture of early systems and incrementally makes use of domain knowledge as it applies to the problem at hand.

Rule-based systems are not good candidates for embedding into general real-time applications. Their response time is slow relative to the needs of the real-time application. However, command reasoning models high level command decision making

which is slow relative to the overall execution time of the simulation [Bimson et al., 1994]. Therefore, rule-based systems are a possibility for this domain.

Rules are a natural method for representation of doctrine which is expressed in terms of conditions and sequences of events to perform. In addition, the ability to modularize rules makes them a good choice for expressing tactical knowledge. Rule-based reasoning is adequate for command reasoning in familiar situations and for representing the reactive behaviors component of a command agent. However, when situational awareness and planning are required in addition to doctrine knowledge, the rule-based paradigm alone is not sufficient. For real-life command reasoning a hybrid architecture using other reasoning mechanisms, such as case-based reasoning, is necessary.

REFERENCES.

Bimson, Kent; Marsden, Craig; McKenzie, Frederic; Paz, Noemi, "Knowledge-Based Tactical Decision Making in the CCTT SAF Prototype", *Proceedings of the Fourth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando, FL, May 4-6 1994, pp. 293-303.

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.

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

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.

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.

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.

Ourston, Dirk; Bimson, Kent, "Integrating Heterogeneous Knowledge Processes in the SAF Behaviors Implementation", *Proceedings of the Fourth Conference on Computer Generated Forces and Behavioral Representation*, Institute for Simulation and Training, Orlando, FL, May 4-6 1994, pp. 333-341.

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. 3-9.

[Return to the Top of this Section](#)

TAA2

TECHNOLOGY AREA ASSESSMENT:

Fuzzy Logic

ASSESSOR:

LTC Robert J. Hammell II, Ph.D.

Army Research Laboratory

ATTN: AMSRL-SC-SA
Aberdeen Proving Ground, MD 21005-5067

Phone:(410)278-4659 (DSN298)
Fax :(410)278-2405
email: rhammell@arl.mil

OVERVIEW OF THE TECHNOLOGY AREA:

Fuzzy set theory provides a formal system for representing and reasoning with uncertain or imprecise information. In classical modeling, relationships are expressed as mathematical functions. As systems become more complex, it is increasingly difficult to develop mathematical models directly from knowledge of the system. Fuzzy models have been used to model complex systems for which only imprecise or approximate specifications are available.

The earliest and perhaps most prevalent applications of fuzzy logic involve the area of control theory. Fuzzy logic controllers have been developed to control cement kilns (Larsen,80;Umbers,80), aircraft (Larkin,85), elevators (Fujitec,88), and automobile transmissions (Kasai,88) just to name a few. Fuzzy models have also been successfully applied in many other areas, including pattern recognition, decision support, approximate reasoning, robotics, natural language and image understanding, machine learning, database systems, expert systems, information processing, data analysis, inventory control, knowledge mining, and others (Zimmerman,85; Bezdek,93; Schwartz,92; Kosko,93; Cox,95).

DESCRIPTION OF THE TECHNOLOGY:

Summary. Fuzzy set theory provides the basis for fuzzy modeling and fuzzy inference. Fundamental to fuzzy set theory is the use of fuzzy sets; in fuzzy sets, objects have a grade of membership ranging from 0 to 1. Fuzzy inference provides the means by which to perform approximate, or fuzzy, reasoning. Key to a fuzzy inference system is the fuzzy rule base, which contains information relating the input conditions to the output conditions. A fuzzy model is created when the fuzzy rule base defines the relationship between the input domain(s) and the output domain(s) of some target system. Most approaches to fuzzy inference can be categorized as generalizations of either logical deduction or approximation theory.

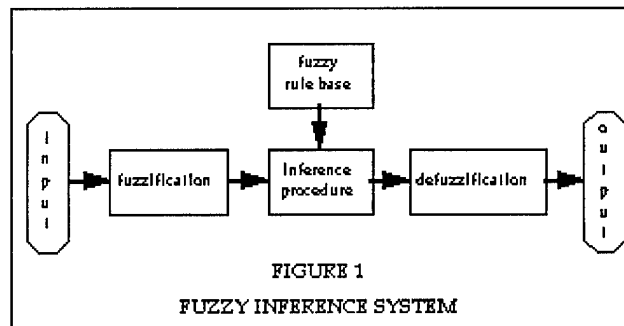
Additional Detail. Fuzzy set theory provides the formal framework for representing (fuzzy modeling) and reasoning (fuzzy inference) with uncertain or imprecise information. There are many books that provide a thorough introduction to fuzzy set theory. Examples include (Dubois,80; Zimmerman,85; Klir,88; McNeill,93).

Fundamental to fuzzy set theory is the use of *fuzzy sets*. The basic concept of fuzzy sets is relatively easy to grasp. We all gather, interpret, and use vague instructions and imprecise data every day. Suppose you are teaching your teenager to drive and are discussing how to approach an upcoming intersection. Would you offer the instruction "If the light has been green 27 *seconds*, release the accelerator 100 *feet* from the intersection"? Or, perhaps "If the light has been green for a *long time*, release the accelerator as you *get near* the intersection"? The precision in the first instruction prohibits its implementation, while the more vague or fuzzy

second instruction can be easily interpreted and applied.

Conventional set theory uses *crisp* sets in which the degree of membership of an object is either 0 or 1. A value of 1 indicates that the object completely belongs to the set; otherwise, it does not belong at all and the object is assigned a membership value of 0. Within fuzzy set theory, the membership function for fuzzy sets provides a *grade* of membership ranging from 0 to 1. That is, objects may partially belong to a fuzzy set ($0 < \text{membership grade} < 1$), completely belong (grade = 1), or not belong at all (grade = 0). For example, 27 seconds might belong to the fuzzy set *long time* with a membership grade of .60, 35 seconds with a grade of .75, and 50 seconds (and anything longer) with a grade of 1. Thus, fuzzy set theory involves classes of objects that do not have crisp boundaries and in which membership is a matter of degree.

Fuzzy inference systems provide a means by which to perform approximate, or fuzzy, reasoning. Zadeh (Zadeh,79) defines approximate reasoning as "the process or processes by which a possibly imprecise conclusion is deduced from a collection of imprecise statements." The general structure of a fuzzy inference system, also called a fuzzy rule-based system, is shown in figure 1 (next page).



A critical component of the system is the fuzzy rule base. The rule base contains the information that relates the input conditions to the output responses. Fuzzy rules have the form "if speed is *fast* and distance is *close* then brake_pressure is *high*," where *fast* and *close* are fuzzy sets in the input domains speed and distance, respectively, and *high* is a fuzzy set in the output domain brake_pressure.

The input to the fuzzifier may be a precise value or a fuzzy set. When the input is from a human observer or in the form of a database query, it is often given as a fuzzy set. If the input is from a sensor device it is usually used as a precise value, with the possibility of inaccuracy due to noise. Fuzzification - the process of transforming the input based on the noise in the data source or the degree of precision required for the inference - produces an interpretation of the input. For example, a database query may ask for the names of all persons six feet tall. The fuzzification of this query may produce a fuzzy set consisting of all heights between 5'10" and 6'2". The fuzzified request may be considered an interpretation of "six feet" by the inference system. Fuzzification is a standard part of fuzzy database systems but is usually omitted in fuzzy control systems.

The input domain is divided into a group of classifications that usually overlap. Each of these classifications is a fuzzy set; together they represent a decomposition of the input domain. The

output domain is decomposed in a similar manner. The input and output domain decompositions define the *topology* of the fuzzy system. An element in the domain has some grade of membership, from 0 to 1 inclusive, in each fuzzy set within its domain. The grade of membership is determined by the membership function, as mentioned above.

The fuzzy sets from the input domain compose the antecedents of the rules. The (perhaps fuzzified) input to the system is compared with the antecedents of the fuzzy rules in the rule base and a degree of matching is obtained. This degree of matching is used in the fuzzy inference process to produce a fuzzy set over the output domain. For situations which require a precise response, the output fuzzy set is transformed to a single value by the defuzzification module.

Linguistic variables are an important concept in fuzzy inference. Basically, a linguistic variable is used to approximately characterize the values of variables as well as their relationships. For example, numbers can be used to characterize a person's height, but using words instead might provide categories such as tall, quite tall, more or less tall, not very tall, more or less small, and so on. In the preceding examples of fuzzy rules, linguistic variables were used to identify the fuzzy sets (categories). The imprecision introduced by using words may or may not be by choice. That is, the imprecision may be intentional based on not needing to be more precise. More often, however, the imprecision is dictated by the lack of a means to quantitatively specify the attributes of an object.

A fuzzy model is simply a collection of rules that define a relationship between the input domain(s) and the output domain(s) of the system being modeled. The language of the model is comprised of the terms used in the antecedents and consequents of the rules. These terms are determined by the input and output domain decompositions mentioned above. The antecedents of the rules consist of the fuzzy sets that partition the input domain. Likewise, the consequents of the rules come from fuzzy sets that partition the output domain.

Numerous approaches for inference using fuzzy rules appear in the literature, but most can be categorized as either generalizations of logical deduction or approximation theory. The first approach, an extension of logical implication to fuzzy predicates, is usually used for inference in fuzzy expert systems and databases. One method is to represent the fuzzy rules with an implication relation between the fuzzy sets, and then use a compositional rule of inference to produce the output fuzzy set (Zadeh, 83). Using the compositional rule of inference is not the only approach to fuzzy inference that falls within the category of generalized logical deduction. Another basic method, among the many that have been proposed, is the compatibility modification method. An excellent treatment of the relationships between these two methods can be found in (Cross,93).

The second approach views fuzzy rule-based inference as functional approximation. An example is in the controls environment, where the control surface of the system under consideration is approximated using fuzzy control rules. Based on this view, a fuzzy rule "if X is A then Z is C " represents a functional relationship between input A and output C . The elements of X that have non-zero membership in A (called the *support* of A) and the elements of Z that have non-zero membership in C (support of C) define a fuzzy relation patch (Kosko,93) which represents the rule and bounds the area covered by the rule (see figure 2,

below). The patch is a fuzzy relation defined by the Cartesian product of the two fuzzy sets A and C , and determines the region in which the rule provides information to the inference process.

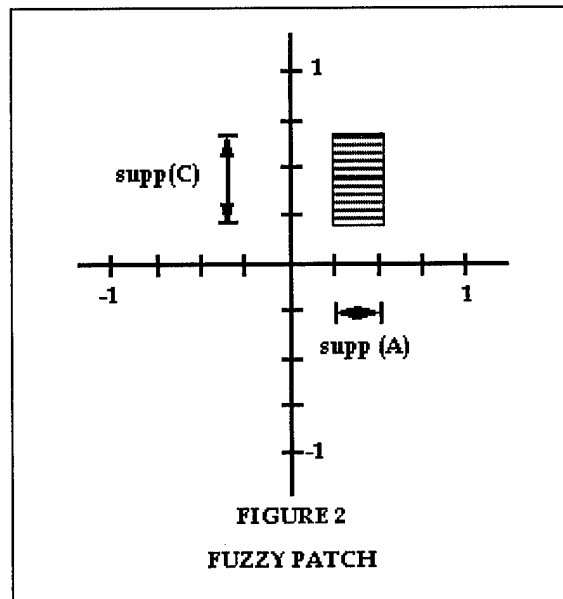
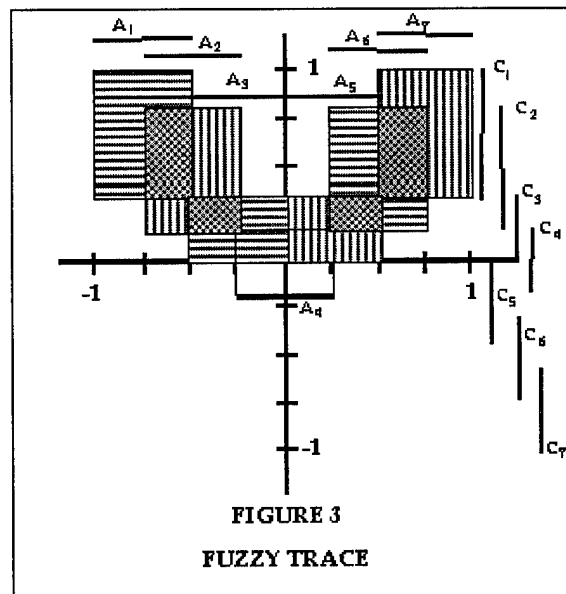


Figure 3 (below) shows an example of a fuzzy rule base "fuzzily" tracing the function $f(x)=x^2$, where the input domain of $[-1,1]$ is decomposed into seven fuzzy sets $A_1, \dots, A_7$ and the output domain of $[-1,1]$ is decomposed into seven fuzzy sets $C_1, \dots, C_7$. With such a rule base, the role of fuzzy inference is to determine an output when the input is any fuzzy set A .



BACKGROUND:

Multivalued logic was developed in the 1920s and 1930s, initially allowing a third "truth" value based on the Heisenberg uncertainty principle in quantum mechanics (Birkhoff,36). The

first formal three-valued logical system was developed in the early 1930s by the Polish logician Jan Lukasiewicz (Rescher,69). He extended the $\{0, 1/2, 1\}$ truth value range to a continuous-valued truth function. Also in the 1930s, continuous logic was applied to sets of elements by Max Black (Black,37). The first fuzzy-set membership functions were developed by Black.

In 1965, Lotfi Zadeh wrote his famous paper formally defining multivalued, or "fuzzy", set theory (Zadeh,65). He extended traditional set theory by changing the two-valued indicator function to a multivalued membership function. The membership function assigns a "grade of membership," ranging from 0 to 1, to each object in the fuzzy set. Zadeh formally defined fuzzy sets, their properties, and various algebraic operations on fuzzy sets. In a later paper (Zadeh,73), he introduced the concept of linguistic variables which have values that are linguistic in nature (ie., height = {tall,medium,short}). Zadeh is widely accepted as the "father of fuzzy logic."

Since the mid-1970s, fuzzy logic has been applied to a variety of applications. The earliest commercial application of fuzzy logic was in the area of controls. In 1978, a fuzzy logic controller for a cement kiln was tested for six days; in 1980 the fuzzy controller was installed in the kiln permanently (Holmblad,82). Although this application was highly successful, and other early tests of fuzzy logic were impressive, researchers and engineers in Europe and the U.S. failed to embrace the technology. In fact, fuzzy logic encountered significant resistance from many mainstream scientists in the Western world. Numerous factors contributed to the lack of interest in fuzzy logic. These included the negative connotations in the name itself and the fact that fuzzy concepts did not parallel conventional control techniques. More complete philosophical and technical discussions regarding the resistance to fuzzy logic can be found in (Kosko,93; McNeill,93).

The Japanese, however, quickly adopted fuzzy logic techniques and began successfully marketing many innovative commercial applications. By September 1990, there were approximately 389 fuzzy patents in Japan. In addition, according to the U.S. Patent and Trademark Office, the Japanese held 30 of the 38 U.S. fuzzy patents issued by December 1990. Products now offered by Japanese companies that utilize fuzzy logic technology include washing machines, camcorders, televisions, air conditioners, microwave ovens, rice cookers, fuzzy focus cameras, copiers, elevators, dishwashers, chemical mixers, language translators, toasters, and many systems for automobiles such as controls for antilock brakes, suspension systems, automatic transmissions, and engine air-fuel mixing (Kosko,93; McNeill,93).

Not surprisingly, fuzzy logic is now enjoying a resurgence in Europe and the United States. Corporate researchers and commercial applications still lag the Japanese but the gap appears to be narrowing. Research interest has grown immensely over the last 10 years and it appears that research devoted to fuzzy logic, and fuzzy logic technological capabilities, will continue to grow at a rapid pace well into the foreseeable future.

NOTABLE IMPLEMENTATIONS AND VARIANTS:

Summary. One main component of fuzzy systems is the fuzzy rules. Historically, fuzzy rules have been obtained by knowledge acquisition from domain experts. Research has been done recently to allow the learning of fuzzy rules from data, much in the same manner as neural nets. The completeness of the rule base is a primary concern regardless of the technique used to develop the rules. Rule base completion techniques exist to allow fuzzy systems to overcome some of the problems of an incomplete rule base.

The other main component of fuzzy inference systems is the membership functions. The choice of membership functions in most systems is not driven by any systematic procedure. Membership functions can be subjectively determined as part of the system design process or, like rules, can be learned from data. Most instances of learning membership functions for fuzzy systems utilize neural network implementations of fuzzy learning algorithms.

A conventional fuzzy system is static. That is, the design of the system is done off-line and the components are not changed once the system becomes operational. An adaptive system, however, has the capability to adjust one or more of its design components during operation. As performance moves outside of some acceptable range, system feedback causes on-line tuning to occur.

Additional Detail. This section will discuss variations in determining the two main components of fuzzy systems - the rules and the membership functions. In addition, adaptive versus non-adaptive fuzzy systems will be examined. Adaptivity refers to the ability of a fuzzy system to adapt to changes in its environment during operation.

Fuzzy Rules. As reported in (Takagi,83) and summarized in (Lee,90), there are four basic methods of deriving fuzzy control rules: 1) based on elicitation of knowledge from a domain expert, 2) based on the observation of a human operator, 3) based on a fuzzy model, and 4) based on learning. While these methods were presented in the context of producing fuzzy *control* rules, they apply equally as well to the generation of fuzzy rules for other applications.

The first three methods listed above use qualitative knowledge of the domain and are heuristic in nature. They are the traditional methods by which fuzzy rules have been obtained, especially the technique of knowledge acquisition from domain experts. These methods can be extremely time consuming and may involve a trial-and-error phase. As the systems being modeled become more complex, these

methods become increasingly unwieldy and less efficient. A review of results achieved by using heuristic methods to develop rules is provided in (Lee,90).

Historically, fuzzy rules have been developed by the heuristic methods while the weights on neural nets have been learned from data. Research has recently been done to allow the learning of fuzzy rules from data (the fourth method mentioned above). Many of the approaches utilize neural nets to derive the fuzzy rules for subsequent use in fuzzy systems design, such as in (Sin,93; Berenji,93). Kosko (Kosko,92) proposed an approach that views rule base construction as a search through the space of all fuzzy rule bases, and uses a relative frequency technique to select rules from the total rules possible. Wang and Mendel (Wang,91) presented a learning algorithm that derives fuzzy rules by determining the entries in a fuzzy

associative memory (FAM). This method was experimentally analyzed in (Sudkamp,94) and extended in (Hammell,95) by incorporating error analysis into the approximations. The extended FAM algorithm demonstrated significant improvement in accuracy over the original algorithm without requiring larger training sets. An advantage of the Wang & Mendel algorithm, and its extension, is that rules from domain experts can be easily combined with the fuzzy rules derived from training data if desired.

In any rule-based system, completeness of the rule base is a primary concern. When domain experts are used to develop rules, conditions may exist that have not been experienced nor anticipated. Likewise, when rules are learned from data the training set may not represent all possible system configurations.

Two approaches can be used to produce an output for an input that "has no rule": interpolation in the underlying Cartesian space or rule base completion. Both of these techniques cause a rule to be influential outside the support of the antecedent thus changing the locality of the fuzzy rule. Interpolation uses the *training data* to interpolate on the known data points in the underlying domains (Koczy,93; Mukaidono,90). Rule base completion can be viewed as interpolating on the *rule base* as opposed to the training data. Rule base completion techniques are presented in (Sudkamp,95) along with results from applying these techniques to the extended FAM learning algorithm mentioned above.

Fuzzy Membership Functions. Membership functions are the other major component of fuzzy inference systems. Figure 4 (below) illustrates some, but by no means all, example continuous membership functions.

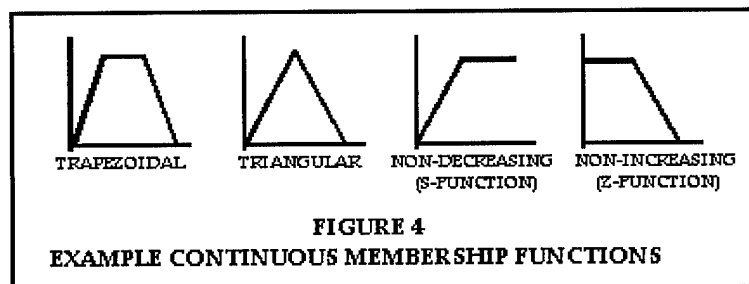
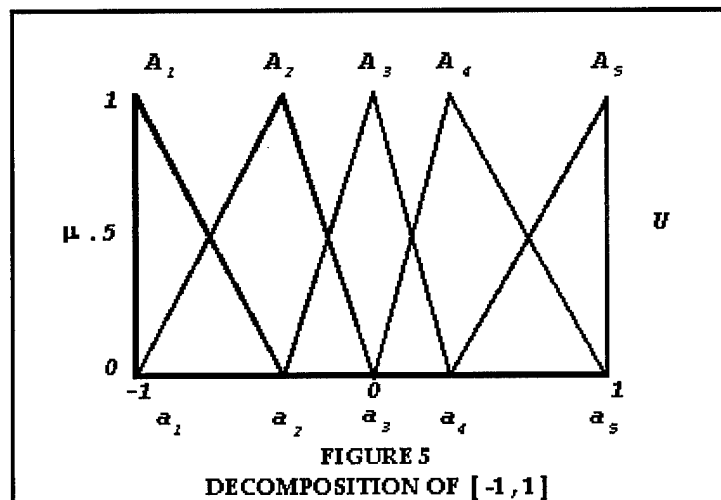


Figure 5 (below) shows a decomposition of the domain $[-1,1]$ into five fuzzy sets, $A_1, \dots, A_5$ using triangular membership functions.



Even though membership functions are an essential part of any fuzzy system, the choice of membership functions in most systems is not driven by any systematic procedure. The method for determining membership functions is usually a subjective process. Several methods for practical estimating of "reasonable" membership functions are presented in (Dubois,80) but suffer from lack of generality. A technique for establishing constraints on membership functions relative to a particular algorithm is presented in (Turksen,93), but is still relatively narrow in applicability.

Like rules, membership functions can be learned from data. Most implementations of learning membership functions for fuzzy systems utilize neural network implementations of fuzzy learning algorithms. The literature is filled with "fuzzy neural network" and "neural fuzzy" references. There are basically two approaches for integrating fuzzy inference and neural nets (Bersini,93). The first approach respects the basics of neural networks and tries to simplify the network elements; for instance, the activation signal could be changed from a sigmoid type to a fuzzy relation type. The other approach is to respect the basics of fuzzy inference by maintaining the concept of reasoning from a rule base of fuzzy rules. This can be done by using the standard neural network learning process to determine the rules and membership functions separate from the fuzzy system, or by simply building a neural net that implements the fuzzy system. A sampling, in no way exhaustive, of neural network implementations of fuzzy algorithms that learn membership functions follows.

The Fuzzy Membership Model (FMM) of Archer and Wang (Archer,91) was proposed as a tool to develop fuzzy membership functions in two-class monotonic pattern recognition problems. The FMM consists of three individual neural nets, with each representing a fuzzy membership function. The "extra" two membership functions are based on two levels of misclassifications of the original training data.

A method of fuzzy reasoning called neural-network driven fuzzy reasoning is described in (Takagi,91). The concept corresponds to incorporating knowledge into the neural network in the form of fuzzy inference rules. The authors explain that their model not only produces the "best design of membership function" during training, but also allows for adaptive tuning of the membership function.

Combining the Restricted Column Energy (RCE) neural network with fuzzy recognition concepts to produce a fuzzy RCE neural network was proposed in (Roan,93). The model uses triangular membership functions and learns them by adjusting the center and width of the triangles during training.

In (Kahn,93), a method for neural network based fuzzy logic design for embedded processor implementation was presented. A multilayered feedforward neural network is used as a fuzzy rule generator (FRG). The FRG learns the rules and membership functions from the sample data via supervised training; after training, the rules and membership functions are represented by the final weights in the network. The fuzzy rules and membership functions are then extracted and used to complete the design of a fuzzy system.

Adaptive Fuzzy Systems. Conventional fuzzy systems are static. That is, design of the system, including rule generation and domain decomposition, is done off-line and is not changed once the system is put into operation. This produces the obvious disadvantage of prohibiting a fuzzy system from adapting to changes in its environment.

Consider a fuzzy controller as part of a braking system. Suppose the fuzzy rules have been developed off-line based on the performance of new parts and the system begins operation. Initially, a brake pressure of *somewhat hard* may have been sufficient to stop within a certain distance at a particular speed. But as the braking components wear, the same amount of pressure in the same situation may be inadequate to meet the required stopping parameters. A non-adaptive system will be unable to adjust its performance to rectify this problem. An adaptive system, however, has the capability to adjust one or more of its design components based on system performance feedback. As performance moves outside of some acceptable range, the feedback causes on-line "tuning" to occur. While this adaptive capability might be nice in some systems, it may be absolutely vital in others.

According to (Isomursu,94) there are basically three sets of fuzzy system parameters that can be manipulated to allow adaptivity: 1) the input and output scaling factors, 2) the membership functions, and 3) the fuzzy rules. Each of these approaches will be briefly reviewed.

The first approach for enabling fuzzy system adaptivity is that of changing the input and output scaling factors. Scaling factors are used to transform the actual range of values for the input and output variables into the normalized range, usually [0,1] or [-1,1]. Although not plentiful in the literature, this method has been investigated for use in adjusting fuzzy system performance. Examples of this technique can be found in (Maeda,92), (Braae,79), (Bare,90) and (Isomursu,94).

Chen and Perng (Chen,94) show that changing scaling factors for input variables amounts to changing the universe of discourse for these variables. They assert that, through this tuning process, any linguistic meaning present in the rule base is destroyed. While this may be acceptable if the only goal of tuning is to improve system performance, the resulting loss of linguistic meaning may not be acceptable otherwise. This is especially true if the rules came from domain experts (versus numerical training patterns) or if the incorporation of new knowledge from domain experts is desired in the future.

The second method mentioned for adding adaptivity to fuzzy systems is that of dynamically changing the membership functions. There are numerous instances of systems that use this technique throughout the literature. Some examples are (Berenji,92), (Jang,92), (Takagi,85), (Karr,93), (Freeman,90), (Lui,94) and (Lotfi,94).

Modifying the fuzzy rule base is the last method of adding adaptivity to fuzzy systems. As with modifying membership functions, the literature provides many examples of adaptive systems that modify the fuzzy rule base during operation. These include (Wu,92), (Gau,94), (Jou,93), (Esogbue,93), (Maeda,92), (Kosko,92), and (Hammell,96).

Tong (Tong,76) asserts that modifying the rule base is the most effective way of improving the performance of a fuzzy controller. Alternately, it is argued in (Lotfi,94) that tuning the membership functions will allow the realization of any input-output mapping. The literature seems to indicate that both methods are capable of effectively allowing adaptivity in a fuzzy system. There are two considerations, however, that make the rule base modification strategy more attractive in some cases. First of all, this technique seems more akin to what humans might do when adapting; at least it seems more intuitive. Secondly, rule base modification allows changes in the system's behavior to be directly reflected and understood by changes that occur in the linguistically interpretable rules. If the capability for tracing, explaining, or understanding performance changes in relation to the rule base is desired, then the rule base modification approach is obviously the best.

TECHNICAL DETAILS:

Most of the performance characteristics of fuzzy systems are fully dependent on the complexity of the model. The complexity of the model, in turn, is usually driven by the complexity of the system being modeled. As such, the computational requirements and the performance potential of fuzzy systems is typically problem-dependent. Specifically, the implementation, the number of rules, the number of inputs and outputs, the degree of precision required, and other related system characteristics will influence the resulting performance of the model.

Some fuzzy system implementations are more efficient than others. Design tradeoffs may be made, especially in the area of number of rules versus accuracy of the model. It is possible, however, to minimize the computational penalty that usually results from an increased number of rules. The two-level fuzzy associative memory (FAM) model (Hammell,95) mentioned earlier can be designed to make the amount of computation required for processing input independent of the number of rules. The two-level FAM architecture is not the only fuzzy system with this property; any TPE system (triangular-partition rule base using evenly spaced midpoints in the fuzzy sets) will allow the time from receiving an input to producing an output to be independent of the number of rules in the fuzzy inference system (Sudkamp,94).

In general, fuzzy systems should not inherently impose additional requirements over more conventional systems in the areas of bandwidth, hardware, or network considerations. In fact, a fuzzy system might very well be less complex than a comparable conventional expert system or decision support system.

The use of fuzzy systems can often reduce development time, improve the knowledge acquisition process, and reduce the number of required rules. The following sections on strengths and weaknesses of fuzzy systems are mostly summarizations of the material found in (Cox,95).

Strengths. One of the main advantages of fuzzy systems is that they are model-free estimators. That is, no mathematical model of the underlying system needs to be developed or even known. The fuzzy model is basically a black-box that represents a system within its rules, membership functions, and fuzzification and defuzzification procedures. If a learning algorithm is used to build the rules from training data, even less about the underlying system needs to be known to construct the fuzzy model.

Fuzzy systems have been shown to be universal approximators. They can easily model complex, non-linear systems, as well as "noisy" systems. According to Cox, fuzzy systems "combine high level flexibility and knowledge representation of conventional decision support and expert systems with the power and analytical depth of natural computing paradigms." Natural computing techniques, such as neural nets and genetic algorithms, must address complex design and interconnection parameters whose interrelation can significantly affect performance. This problem is greatly reduced with fuzzy systems.

The use of fuzzy systems can minimize cognitive dissonance - the difference between the way a human expert views or articulates a problem and its solution versus the way it is coded. A reduction in cognitive dissonance has several implications, including improved knowledge acquisition, reduced development time, and greater understanding of the model. Fuzzy systems also tend to be more robust than conventional systems, thereby providing more predictable and stable behavior.

Weaknesses. If a well-established mathematical model exists, it would probably be cheaper and easier to use than to develop a fuzzy system. This is not really a weakness but simply a common-sense design decision. It may be more expensive, and produce a less accurate solution, to build a fuzzy system for a process that is well behaved, well understood, and mathematically describable.

Despite common perception, fuzzy systems do not easily handle missing or incomplete information. As pointed out by Cox, "fuzzy logic is not a clairvoyant technology." Fuzzy models by themselves cannot make accurate judgments about missing data. The expression and use of uncertainty and imprecision of a parameter is fundamental to fuzzy logic; the absence of information about a parameter is not. Given some sort of known reference point or historical information, however, fuzzy logic can assist in "discovering" probable values for missing information in certain cases.

Fuzzy systems are generally sensitive to large numbers of inputs and outputs. This is mainly due to the number of rules required to "cover" the underlying system being modeled. The number of rules is increased with each additional input or output variable; for any given variable, the number of rules is increased as the number of fuzzy sets in the variable's domain

are increased. The exact number of rules needed may depend on the fuzzy rule base representation as well as on the behavior of the system being modeled. Generally, the number of rules will at least grow exponentially with the number of inputs.

The selection of membership functions and domain decompositions for fuzzy systems is usually a subjective process. While there are some methods to assist in these design decisions, no optimal solution is readily available. The performance of the system is directly influenced by the "appropriateness" of both these parameters. Although not as complicated as the design decisions for most neural networks and genetic algorithms (which include objective functions, training algorithm, number of layers, network connectivity, neuron type, data transformation method, etc.), the selection of membership functions and domain decompositions must be given careful attention.

APPLICABILITY TO COMMAND DECISION MODELING:

The fundamental nature of fuzzy logic makes it an ideal technology for use in command decision modeling. All humans use approximate reasoning based on fuzzy information and fuzzy rules daily. The ability to use imprecision is what allows us to easily park a car, call a particular color green, or decide if a product is too expensive.

Fuzzy logic has already begun to be applied to military simulations. Sonalysts, Inc. has developed a fuzzy logic decision support system to determine mission effectiveness for special operations forces. The focus of the project dealt with the launch phase of seaborne SEAL missions and was concerned with analyzing the effort of various employment options on mission effectiveness. The Simulator Systems Research Unit at the Army Research Institute has developed a prototype fuzzy logic command entity to deploy and control MSE communications equipment. These are only two examples; others no doubt exist.

Not all decisions faced by a commander, or an individual soldier, are fuzzy. Much of the work done for current battlefield simulations can still be used. Fuzzy logic, however, provides the potential for more autonomous intelligent behavior by allowing an agent to reason like a human - fuzzily. What is the best course of action? Is the enemy strength too great for a particular mission to succeed? Where should the communications resources be placed? Does the available sensor and imagery data indicate enemy forces in the area? All these questions and decisions, and many more, are faced by commanders during battle and by agents during simulation. The successful application of fuzzy logic to the areas of decision support, data analysis, risk analysis, scheduling, approximate reasoning, signal and information processing, and expert systems indicates its significant potential as a technology for command decision modeling.

CONCLUSION:

Fuzzy logic is a powerful tool for modeling and inference. It has been successfully applied in the commercial world to manufacturing processes as well as being incorporated in currently marketed consumer products. Additionally, research in fuzzy logic is extremely active both within industry and academia.

Fuzzy systems can process vague, imprecise information to produce approximate solutions for complex, ill-defined problems. In addition, fuzzy logic has the potential to simplify the knowledge acquisition process, reduce development time, and help provide a better understanding of the system model. The fundamental principles of fuzzy logic make it an excellent technology for assisting in the command decision modeling effort.

REFERENCES:

- Archer, N. and S. Wang, "Fuzzy set representation of neural network classification boundaries," *IEEE Trans. Sys., Man and Cybernetics*, Vol. 21, No. 4, pp 735-742, July/ August 1991.
- Bare, W., R. Mulholland, and S. Sofer, "Design of a self-tuning rule based controller for a gasoline refinery catalytic reformer," *IEEE Trans. of Automatic Control*, Vol. 35, pp 156-164, 1990.
- Berenji, H. and P. Khedkar, "Learning and tuning fuzzy logic controller through reinforcements," *IEEE Trans. on Neural Networks*, Vol. 3, No. 5, pp 724-740, Sep 1992.
- Berenji, H. and P. Khedkar, "Clustering in product space for fuzzy inference," *Proc. of the Second IEEE Int'l Conference on Fuzzy Systems*, San Francisco, CA, pp 1402-1407, April 1993.
- Bersini, H., J. Nordvik, and A. Bonarini, "A simple direct adaptive fuzzy controller derived from its neural equivalent," *Proc. of the Second IEEE Int'l Conference on Fuzzy Systems*, San Francisco, CA, pp 345-350, April 1993.
- Bezdek, J.C., "Fuzzy Models - What are they, and why?" *IEEE Trans. on Fuzzy Systems*, Vol. 1, No. 1, pp 1-6, Feb 1993.
- Birkhoff, F. and J. von Neumann, "The logic of quantum mechanics," *Annals of Mathematics*, Vol. 37, No. 4, pp 823-843, October 1936.
- Black, M., "Vagueness: An exercise in logical analysis," *Philosophy of Science*, Vol. 4, pp 427-455, 1937.
- Braae, M. and D.A. Rutherford, "Selection of parameters for a fuzzy logic controller," *Fuzzy Sets and Systems*, Vol. 2, pp 185-199, 1979.
- Chen, Y. and C. Perng, "Input scaling factors in fuzzy control systems," *Proc. of the Third IEEE Int'l Conference on Fuzzy Systems*, Orlando, FL, pp 1666-1670, June 1994.
- Cox, E., *Fuzzy Logic for Business and Industry*. Rockland, MA: Charles River Media, Inc., 1995.
- Cross, V., "An analysis of fuzzy-set aggregators and compatibility measures," Ph.D. Dissertation, Wright State University, Dayton, OH, 1993.
- Dubois, D. and H. Prade, *Fuzzy Sets and Systems*. Orlando, FL: Academic Press, 1980.

- Esogbue, A. and J. Murrell, "A fuzzy adaptive controller using reinforcement learning neural networks," *Proc. of the Second IEEE Int'l Conference on Fuzzy Systems*, San Francisco, CA, pp 178-183, April 1993.
- Freeman, L.M., K. Krishnakumar, C. Karr, and D. Meredith, "Tuning of fuzzy logic controllers using genetic algorithms - aerospace applications," presented at AAAIC'90 Conference, Dayton, OH, Oct 1990.
- Fujitec, F., "FLEX-8000 series elevator group control systems," Fujitec Co., Ltd., Osada, Japan, 1988.
- Gau, J. and W. Lu, "Application of self-organizing fuzzy logic controller in 2-D motion tracking control," *Proc. of the Third IEEE Int'l Conference on Fuzzy Systems*, Orlando, FL, pp 1660-1665, June 1994.
- Hammell II, R.J. and T. Sudkamp, "A two-level architecture for fuzzy learning," *Journal of Intelligent and Fuzzy Systems*, Vol. 3, No. 4, pp 273-286, 1995.
- Hammell II, R.J. and T. Sudkamp, "An adaptive hierarchical fuzzy model," *Expert Systems with Applications*, Vol. 10, No. 4, to appear, July 1996.
- Holmblad, P. and J. Ostergaard, "Control of a cement kiln by fuzzy logic," in *Fuzzy Information and Decision Processes*. M.M. Gupta and Elie Sanchez, Eds., Amsterdam: North-Holland, 1982.
- Isomursu, P. and T. Tauma, "A self-tuning fuzzy logic controller for temperature control of superheated steam," *Proc. of the Third IEEE Int'l Conference on Fuzzy Systems*, Orlando, FL, pp 1560-1563, June 1994.
- Jang, J.R., "Self-learning fuzzy controllers based on temporal back-propagation," *IEEE Trans. on Neural Networks*, Vol. 3, No. 5, pp 714-723, Sep 1992.
- Jou, C., "Supervised learning in fuzzy systems: Algorithms and computational capabilities," *Proc. of the Second IEEE Int'l Conference on Fuzzy Systems*, San Francisco, CA, pp 1-6, April 1993.
- Kahn, E. and P. Venkatapuram, "Neufuz: Neural network based fuzzy logic design algorithms," *Proc. of the Second IEEE Int'l Conference on Fuzzy Systems*, San Francisco, CA, pp 647-654, April 1993.
- Karr, C.L. and E.J. Gentry, "Fuzzy control of pH using genetic algorithms," *IEEE Trans. on Fuzzy Systems*, Vol. 1, No. 1, pp 46-53, Feb 1993.
- Kasai, Y. and Y. Morimoto, "Electronically controlled continuously variable transmission," *Proc. of the Int'l Congress on Transportation Electronics*, Dearborn, MI, 1988.
- Klir, G.J. and T. Folger, *Fuzzy Sets, Uncertainty and Information*. Englewood Cliffs, NJ: Prentice Hall, 1988.
- Koczy, L. and K. Hirota, "Approximate reasoning by linear rule interpolation and general approximation," *Int'l Journal of Approximate Reasoning*, Vol. 9, No. 3, pp 197-225, 1993.

- Kosko, B., *Neural Networks and Fuzzy Systems: A dynamical systems approach to machine intelligence*. Englewood Cliffs, NJ: Prentice Hall, 1992.
- Kosko, B., *Fuzzy Thinking*. New York, NY: Hyperion, 1993.
- Larkin, L.I., "A fuzzy logic controller for aircraft flight control," *Industrial Applications of Fuzzy Control*. M. Sugeno, Ed., Amsterdam: North Holland, pp 87-104, 1985.
- Larsen, P.M., "Industrial application of fuzzy logic control," *Int'l Journal of Man-Machine Studies*, Vol. 12, No. 1, pp 3-10, 1980.
- Lee, C.C., "Fuzzy logic in control systems: fuzzy logic controller - Part I," *IEEE Trans. on Systems, Man and Cybernetics*, Vol. 20, No. 2, pp 404-435, March/April 1990.
- Lotfi, A. and A.C. Tsoi, "Importance of membership functions - A comparative study on different learning methods for fuzzy inference systems," *Proc. of the Third IEEE Int'l Conference on Fuzzy Systems*, Orlando, FL, pp 1791-1796, June 1994.
- Lui, H., M. Gu, T. Goh, and P. Wang, "A self-tuning adaptive resolution (STAR) fuzzy control algorithm," *Proc. of the Third IEEE Int'l Conference on Fuzzy Systems*, Orlando, FL, pp 1508-1512, June 1994.
- Maeda, M. and S. Murakami, "A self-tuning fuzzy controller," *Fuzzy Sets and Systems*, Vol. 51, No. 1, pp 29-40, Oct 1992.
- McNeill, D. and P. Freiburger, *Fuzzy Logic*. New York, NY: Simon and Schuster, 1993.
- Mukaidono, L.D. and Z. Shen, "Approximate reasoning based on the revision principle," *Proceedings of NAFIPs'90*, Toronto, Canada, pp 94-97, 1990.
- Rescher, N., *Many Valued Logic*. New York: McGraw-Hill, 1969.
- Roan, S., C. Chiang, and H. Fu, "Fuzzy RCE neural network," *Proc. of the Second IEEE Int'l Conference on Fuzzy Systems*, San Francisco, CA, pp 629-634, April 1993.
- Schwartz, D.G. and G.J. Klir, "Fuzzy Logic: Concepts to constructs," *AI Expert*, pp 26-33, Nov 1993.
- Sin, S. and R. deFigueiredo, "Fuzzy system design through fuzzy clustering and optimal predefuzzification," *Proc. of the Second IEEE Int'l Conference on Fuzzy Systems*, San Francisco, CA, pp 190-195, April 1993.
- Sudkamp, T. and R.J. Hammell II, "Interpolation, completion, and learning fuzzy rules," *IEEE Trans. on Systems, Man, and Cybernetics*, Vol. 24, No. 2, pp 332-342, Feb 1994.
- Sudkamp, T. and R.J. Hammell II, "Rule base completion in fuzzy models," in *Fuzzy Modeling: Paradigms and Practice*. W. Pedrycz, Ed., Kulwer Academic Publishers, to appear, 1995.

Takagi, T. and M. Sugeno, "Derivation of fuzzy control rules from human operator's control actions," *Proc. of IFAC Symposium on Fuzzy Information, Knowledge Representation and Decision Analysis*, Marseilles, France, pp 55-60, July 1983.

Takagi, T. and M. Sugeno, "Fuzzy identification of systems and its application to modeling and control," *IEEE Trans. on Systems, Man, and Cybernetics*, Vol. SMC-15, No. 1, pp 116-132, 1985.

Takagi, H. and I. Hayashi, "NN-Driven fuzzy reasoning," *Int'l Journal of Approximate Reasoning*, Vol. 5, No. 3, pp 191-212, May 1991.

Tong, R.M., "Analysis of fuzzy control algorithms using the relation matrix," *Int'l Journal of Man-Machine Studies*, Vol. 8, pp 679-686, 1976.

Turksen, I.B. and Y. Tran, "Constraints on membership functions of rules in fuzzy expert systems," *Proc. of the Second IEEE Int'l Conference on Fuzzy Systems*, San Francisco, CA, pp 845-850, April 1993.

Umbers, I.G. and P.J. King, "An analysis of human decision-making in cement kiln control and the implications for automation," *Int'l Journal of Man-Machine Studies*, Vol. 12, No. 1, pp 11-23, 1980.

Wang, L. and J.M. Mendel, "Generating fuzzy rules from numerical data, with applications," Tech. Rep. USC-SIPI-169, Signal and Image Processing Institute, University of Southern California, Los Angeles, 1991.

Zadeh, L.A., "Fuzzy sets," *Information and Control*, Vol. 8, pp 338-353, 1965.

Zadeh, L.A., "Outline of a new approach to the analysis of complex systems and decision processes," *IEEE Trans. on Systems, Man, and Cybernetics*, Vol. SMC-3, pp 28-44, 1973.

Zadeh, L.A., "A theory of approximate reasoning," in *Fuzzy Sets and Applications*. R.R. Yager, S. Orchinnikov, R.M. Tong, H.T. Nguyen, Eds., New York: John Wiley & Sons, pp 367-412, 1987, originally written in 1979.

Zadeh, L.A., "The role of fuzzy logic in the management of uncertainty in expert systems," *Fuzzy Sets and Systems*, Vol. 11, pp 199-227, 1983.

Zimmerman, H.J., *Fuzzy Set Theory and Its Applications*. Boston, MA: Kluwer-Nijhoff, 1985.

[Return to the Top of this Section](#)

TAA3 **TECHNOLOGY AREA ASSESSMENT:**

CASE-BASED REASONING

ASSESSOR:**Christopher Dean, M.S.Cp.E.**

Science Applications International Corporation
3045 Technology Parkway
Orlando, Florida 32826-3299
(407) 282-6700
deanc@orl.saic.com

OVERVIEW OF THE TECHNOLOGY AREA:*DESCRIPTION OF THE TECHNOLOGY:*

Many human problem solving techniques are based on reviewing previous problem experiences and applying their solutions to similar new problems [Gonzalez and Dankel, 1993]. This is the basis for Case-Based Reasoning (CBR) which uses a knowledge base of previous explicit experiences known as cases to solve problems. Similar cases are retrieved from a case-base and modified to fit the current situation. The new case is added to the case-base and each retrieved case is updated with the knowledge of whether it succeeded or failed in providing support for the solution of the new case. Unlike other reasoning paradigms such as rules, CBR is a machine learning technique that adapts to new situations and learns from them. How cases are adapted vary among implementations of CBR and are dependent on the domain of application.

One of the critical CBR tasks is the construction and organization of the case-base. Searching and reviewing of cases for applicability can be computationally expensive, so a proper set and organization of case indices must be developed that can be used to determine the similarity of cases. Indices usually consist of attributes that define the case, such as, inputs (the situation) and goals (solutions) it achieved. Proper refinement and weighting of these indices are of extreme importance not only to ensure that just relevant cases are retrieved, but also to reduce the total number of cases retrieved for review.

CBR is often compared against traditional rule-based reasoning. Both reasoning methodologies are patterned after human reasoning, but their knowledge representations are quite different. To represent knowledge as rules an expert must recollect and interpret the domain. CBR details the knowledge directly from historical cases and hence is a more formal and explicit representation of the domain by an expert [Gonzalez and Dankel, 1993]. Having a set of historical cases eases the knowledge acquisition process and allows the base of experience to come from more than one individual. If a library of historical cases does not exist, then creating the case-base can be more difficult than representing the knowledge as rules.

Since tactical decision-making has a great deal of historical examples, it is well-suited for CBR paradigm. An essential aspect of command reasoning is the need to plan courses of action. Case-Based Planning (CBP) is a form of CBR that is applicable to command reasoning systems. Cases consist of situational contexts and a solution plan to be executed. The situational context is composed of events that signify the situation (state of the world), any constraints, and the goals and sub-goals of the mission. The plan contains the actions necessary to achieve the goals in the case. CBP can be compared to both the state space representation and action ordering representation of generative planning [Blau et al., 1991]. The state space representation of previous plans is adapted using domain knowledge, current goals, and the context to obtain new plans.

The use of analogical reasoning is CBR's greatest advantage. New problems are solved by analogy with old ones and explanations are stated in terms of prior experience. The power of CBR is drawn from a large case library and not from a set of first principles. To use these cases effectively, there must be a rich indexing mechanism. What the indices are and their order of importance depends on the domain. Thus, a general CBR system not only learns new cases as it proceeds, but also learns from experience a proper set of indices.

The primary disadvantage of the CBR involves issues associated with the computational cost of obtaining a solution. Searching through a case-base of poorly selected and indexed cases is expensive. A successful implementation of CBR depends on issues such as the quality of case indexing, case library architecture, case quality, and the proper number of cases. These issues are not inherent to the CBR technique but mere implementation problems to be overcome [Gonzalez and Dankel, 1993].

Case-based reasoning is applicable to a wide range of real-world situations. These situations can be knowledge rich, making construction of solutions complex. Cases can be built with specific knowledge about a problem, thereby reducing the effort to find the fewest relevant cases. Other real-world situations are weak in specific domain knowledge and only have access to domain-independent generally applicable knowledge for solving problems. Knowledge can be incomplete and evidence can be sparse. For these situations, CBR makes assumptions to fill in incomplete or missing knowledge based on cases in its case-base. The answers arrived at in this manner may not be optimal or not even completely correct, but CBR does provide a methodology to generate answers easily.

Cases provide a concrete starting point for deriving a solution. In these ways, CBR reduces the cognitive load involved in solving problems from the complex real-world [Kolodner, 1993].

NOTABLE IMPLEMENTATIONS AND VARIANTS:

CBR and Computer Generated Forces

A recent use of CBR for modeling reactive behavior for Computer Generated Forces (CGF) focuses on the knowledge acquisition aspect of CBR research [Keirsey et al., 1995]. While the research does not explicitly use CBR for command and control or for command reasoning, the work does address the use of CBR for lower level behaviors. This work has the applicability of being scaled up into more complex upper echelon behaviors for tactical decision-making. The domain modeled is air combat. The choice of the air domain greatly simplifies the case structure since terrain does not need to be included. The technique does illustrate the capability of multiple autonomous agents to interact intelligently and effectively in both cooperative or competitive modes of operation.

Cases are defined simply using state variables that describe the motion and geometry of intelligent forces (IFOR) at a particular instant (e.g., angle off and range for air-combat domain). This describes the simulation space. Behavior responses are assigned to each case. The state variables form orthogonal axes of a multi-dimensional decision space. Thus, a case is defined as point in the decision space in terms of these axis state variables. Any specific configuration of vehicles in the simulation space can be mapped to a specific point in the decision space. A point in the decision space may map to a variety of configurations in the simulation space. Through careful selection of the variables that define the axes of the decision space, the simulation space may be designed to be invariant to rotation and translation of configurations in it.

Thus, a single point in the decision space maps to all possible rotations and translations of a particular configuration of vehicles. This helps reduce the number of applicable cases and aids in the case retrieval. The retrieved case corresponds to the point closest to the current situation point in the decision space. Actually, the decision space is divided into regions that represent different actions to be performed. These regions are defined by a Voronoi Diagram. A Voronoi Diagram for N cases in the decision space is a partitioning of the plane into N polygonal regions, one region associated with each case. Each point within a Voronoi region is closer to the case point for that region than it is to any other case point in the decision space [Keirsey et al., 1995].

For lower level behaviors, the retrieved case controls actions by turning on or off the appropriate behaviors (e.g., Pursue Threat and Avoid Threat) and selecting the relative priorities of component control laws. In many cases, control laws are set to an intermediate state giving them only partial influence over the ultimate control decision. Each control law is assigned a weight between zero and

one. It is possible that this mechanism can be applied to command and control behavior, but more research in this area is required.

One of the problems with using the CBR approach here is that for complex multi-agent scenarios, the number of possible configurations of vehicles becomes too complex for the meaningful retrieval of cases. This necessitates the organization of an intelligent agent's perception against a more consistent and stable set of features [Kiersey et al., 1995]. In other words, a focus of selection is needed. For example, a focusing technique akin to human decision-making is focusing on a single target, real or hypothesized.

Another problem with this approach is that agents are unable to capture deep knowledge. This makes it difficult to indicate reasons why certain actions are performed and to incorporate this reasoning into the decision making process. Complex tactics are also difficult to capture with cases. Many times slightly different situations require drastically different actions. This leads to many cases that are similar making the retrieved case set large or making it difficult to provide enough cases to fully characterize the decision space. To reduce these difficulties a rich and effective case index mechanism must be devised.

Military History as a basis for CBR

An early association between CBR and tactical decision-making can be found in Dupuy [1988]. In this work, the deficiencies in traditional expert systems for command reasoning are highlighted while the appropriateness of CBR is suggested. Dupuy states traditional expert systems have not been very successful in the military domain due to several intractable problems [Dupuy, 1988]:

- the complexity of the interrelated and interdependent problems commander must handle.
- the majority of military situations are dependent upon the individual, idiosyncratic behavior of large numbers of unpredictable human beings.
- the flow of engagements are never exactly the same as previous ones even if they look superficially similar.
- there are no living experts personally familiar with all the variety of combat situations that have occurred in the past.
- the continued addition of new weapons and other hardware make the current situations very different from past ones.
- the doctrine continually changes as weapons and equipment change.

Using military history as the basis for a case base can overcome these problems. Since there are patterns and similarities in the conduct and results of battles going back to 500 BC, a rich case-base can be constructed for military decision-making [Dupuy, 1988]. However, Dupuy only suggests an overall methodology without addressing the major issues or implementation. The problem of case matching to the current situation is mentioned but no substantial solutions are presented.

The emphasis of the work is on advising the commander more than autonomous command and control. This is quite common of the early CBR work which placed CBR in an advisory role [Dupuy, 1988; Wall et al., 1988; Goodman, 1989]. CBR advises the commander of possible course of actions (COAs), cautions, and alternatives with their associated risk. The approach is based upon the classical US Army approach to the estimate of a situation involving these steps [Dupuy, 1988]:

- 1) Assess the current situation based upon METT-T concerns. The results relate to indices in the historical case base.
- 2) Analysis of cases from historical combat. Indices are created for each case to facilitate categorization and comparison with the current situation. The details of a historical situation are elaborated, as is the extent to which action contributed to the success or failure of the case.
- 3) Review of expert's wisdom. A database containing the knowledge of past theorists and successful commanders who have written extensively on the art of war is used. The knowledge will be in the form of rules, hypotheses, laws, etc. that are related to the case indices and displayed to the commander.

- 4) Doctrine Review. The case indices are related to the current situation.
- 5) Analysis of Results. Identify what the winners did, losers did, the exceptions to do's and don'ts, and any special considerations or cautions.
- 6) Determination of possible courses of action, including the commander's and the enemy's.
- 7) Evaluation of courses of action. The feasibility of each COA is determined and success probabilities are assigned. The best COA is recommended along with errors to be avoided.
- 8) Decision. In this case the commander makes the decision but a command agent could easily choose the recommended COA.

This methodology serves only as a foundation for future CBR processes for tactical decision-making.

Case-Based Planning Since tactical decision-making is based upon experience and involves uncertainty, CBR seems to be a natural fit. The primary application of CBR to command reasoning is in the area of case-based planning. In implementing case-based planning, three questions need to be addressed [Wall et al., 1988]:

- 1) How should case library be organized?
- 2) How should the retrieval be performed so that all relevant cases are retrieved?
- 3) How should the cases be presented so that a correct decision can be made?

The third question is a user interface issue requiring explanation and argumentation facilities that are of little use to an automated command reasoner. The first two questions, however, are very relevant to command decision modeling. Early work by Wall et al. [1988] suggests the use of

dimensions as indices to organize cases. Dimensions have conditions that act as an initial filter and contain focus slots that identify what can be changed in a case to make the case stronger or weaker along a given dimension. It is the intersection of the situated cases with these dimensions that guides retrieval much like the CBR for CGF mentioned previously [Kiersey et al., 1995]. Dimensions allow the similarity of cases to be determined while still emphasizing the differences between the cases [Wall et al., 1988].

Scenario Generation.

Case retrieval is guided by goal-directed reasoning. The focus is placed on the cases that are the most relevant to the success or failure of the given plan. This is especially important for military tactical planning in which many times the plans are predetermined (orders) but the actions to implement the plan are not specified. However, CBP can be taken one step further and determine the plan itself as well as the actions [Chaib-draa et al., 1993].

The CBP approach in this work centers around the generation of scenarios [Wall et al., 1988]. A scenario is simply defined as a collection of examples of possible future events. The information contained in the scenario is used to select a COA. The scenario depicts the events that might happen if a given COA is selected, the opponent follows an assumed response, and random processes, such as weather, act in assumed ways. This case base is divided up into categories depending upon these characteristics. The steps of scenario generation include [Wall et al., 1988]:

- 1) Describe current engagement and identify key situational factors including unit goals.
- 2) Given the possible COAs, potential future cases are retrieved.
- 3) Examples are used to explain the advantages or disadvantages of a given COA. Since this work considers CBP in an advisory role this is necessary.
- 4) The commander (or command agent in future systems) chooses COA based upon features of interest such as surprise or indirection.

- 5) The unit carries out the selected plan.
- 6) The events, decisions, and results are analyzed for the situational features that need to be modified or added to the case. The organization of the cases may also need to be modified for more efficient retrieval in the future.

The cases themselves are organized in a taxonomy based upon the type of engagement (1-on-1, 1-on-many, or overwhelmed), the force (friendly or enemy force), the components of the force (air support, strength of reserves, logistics, force-mobility, disposition, and command and control), the goal, and terrain factors (fields-of-fire, cover and concealment, mobility, key terrain, visibility, obstacles, and avenues-of-approach). These form the dimensions of the case. The information available to the case determines its granularity facilitating a multiple fidelity level approach. The network proximity based upon these dimensions determines the similarity and is used to generate hypothetical situations. Domain relationships are specified in the definition of the dimensions, not in the cases, and thus provides a general way to retrieve cases without depending upon explicit links in the cases themselves. Modifications are only made to the domain relationships. Cases themselves are not modified other than to modifying the case's dimensions. Using domain dimensions, cases become graph structure that can be matched based upon the taxonomy, implying a semantic similarity [Wall et al., 1988].

Each case contains:

- the start conditions of the situation.
- the sequence of events performed by the units and the corresponding results.
- the goals of the planner at the time.
- the intended actions, including the COA selected by the planner and the anticipated enemy actions.
- the disposition of opponent, i.e. the perceptions of the opponent's goals.
- any assumptions about domain unknowns or random processes (e.g., weather) and their effect, if any, on achieving the goals.
- the key aspects that led to success or failure of the engagement.

As a practical example of the CBP process, consider seven T-80 tank platoons advancing in echelon with heavily protected flanks and a headquarters (HQ) in the rear. A BLUFOR M-1 tank platoon had established contact with the OPFOR's left flank. Four BLUFOR tank platoons and a company HQ are located to the south of the engagement area. Due to the presence of roads, both forces have MODERATE to HIGH mobility. Command and control attribute for the OPFOR is BELOW-AVERAGE since the HQ is out of visual range and because of the strict command hierarchy imposed on the OPFOR. Command and control attribute is GOOD for BLUFOR since they have an HQ present. The friendly force factors are classified as either MOBILE-FORCE, MODERATE-FORCE, or HIGHLY-MOBILE-FORCE. The enemy force factors are classified as either MODERATE-OFFENSIVE-FORCE or LARGE-OFFENSIVE-FORCE. The goal of the BLUFOR is DEFEND as ordered by the upper echelon commander. The cases retrieved were classified as MOBILE-DEFENSE, SCREENING-DEFENSE, MOBILE-OR-SCREENING-DEFENSE, COUNTER-ATTACK, and ASSUME-OBJECTIVE [Wall et al., 1988].

As an example of case, consider the MOBILE-DEFENSE case. It has the following representation [Wall et al., 1988]:

Case-ID: MEETING-ENGAGEMENT

Background: OPFOR units of unknown strength are exploiting a breach in the front lines. BLUFOR forces consist of an armored battalion with M-1's and two M-2 infantry companies. The objective is to delay and seal the breach if possible.

Course of Action: MOBILE-DEFENSE

Results: Heavy attrition occurred on both sides. OPFOR overextended itself and was exploited by

a BLUFOR counterattack which was accomplished by splitting the force into wings.

Mobility factor: Enemy force was reconnoitered quickly.

Deployment factor: The BLUFOR forces were split opportunistically.

Plan Advisor.

Another early work in CBP was done by Goodman [1989] for a CBR expert system that served as an advisor to a planner. As is with the work in Wall et al. [1988] there is no reason that with present technology it could not be adapted for an automated command reasoner. It focuses on the problem of military situational assessment and its role in planning; there is no complete military causal domain model. Much like the work of Dupuy [1988], a database of historical battles and prior planning information is needed. Also, plan modification needs to be supported [Goodman, 1989].

Cases are represented in the form of frames using the Land Warfare Database. Fifty-seven fields of data are divided into three groups: battle characteristics useful for case retrieval and prediction, predicted outcome information, and documentation/designation information. The frames use symbolic values to represent semantic information and contain links to related cases (same combatants, same attacker, same defender, etc.). Also included is a tactical map and historical commentary. The cases were developed at the constructive level of granularity. For example, the original Land Warfare Database contains twenty-seven distinct values for information on whether each side had an advantage or disadvantage concerning logistics [Goodman, 1989]. This was reduced to three values: attacker has the advantage, defender has the advantage, and neither has the advantage. Similar reductions were made for terrain, weather, etc. The symbols form hierarchies that are used for case retrieval. For example, cases with low trafficability can be retrieved whether they are heavily wooded or consist of marshy terrain. For case base construction, the symbolic indices have the problem of fidelity when deciding to subclass them. For this work, classes were created only when a clear generalization between subclasses could be identified. Cases also consisted of formulas to derive new features, and like the symbolics, the granularity of the formulas is an issue.

The initial case is retrieved using nearest neighbor similarity function. Weights are assigned to the fields to be used in the matching. For symbolic fields, the inferential distance based upon the hierarchy is used for the weight. There are several shortcomings with this approach. From a knowledge engineering standpoint there can be problems determining the weighting factors. In this case, subject matter experts could come up with an initial weighting and could then refine the weight for different case retrievals. Over time, the experts could lose the context under which modifications were made causing previous cases to be retrieved incorrectly [Goodman, 1989]. In addition, it is difficult for a planner to determine what modifications need to be made to the plan in order to bring about a more favorable outcome.

This same problem also applies to autonomous planners. The problem can be reduced by presenting a list of similarities and differences in the input and by ordering retrieved cases by field weight. However, it is still difficult to interpret. Furthermore, the retrieval algorithm is very sensitive to small changes in the situation description which causes new cases to be retrieved. This is not easily understood by the planner and thus causes confusion. For an automated command reasoner, these presentation problems is not a problem since there is no symbolic to human thinking translation necessary. However, a methodology still needs to be developed to guide plan modification.

To counteract the previous problems, the nearest neighbor retrieval algorithm is replaced with an inductive discrimination analysis for knowledge acquisition. An interactive interface aids the subject matter experts in screening superfluous cases made during various consultations. [Goodman, 1989]. Traces of significant indices are displayed during a consultation to provide easier recognition of alternatives for plan modification.

When the planner is serving as an advisor the ease in which retrieved cases can be related back to the original situation is an important issue. An example the authors use is a commander defending a battle position in a prepared posture with the system returning similar cases where the defender won but used a fortified defense [Goodman, 1989]. This option may not be available to the commander in the field. The present solution is to introduce case prototypes which are conjunctions of indices which reflect conceptual categories of cases. For example, separate prototypes can be created for prepared and fortified defensive postures. Discrimination analysis only proceeds on cases indexed under each prototype. When a commander describes the current situation, the planning system returns only the cases where the defender is in a prepared defense. This situation demonstrates a possible problem with indices. The importance and usefulness of fields (indices) are context dependent and thus vary with the situation.

The research also noted that learning new failures automatically is difficult because subject matter experts must screen for causally meaningful indices. Indices must be automatically generated to explain these failures which is difficult to do. At the time this article was written the authors were investigating the addition of causal information to the system to guide automatic index generation [Goodman, 1989]. For an autonomous command reasoner this would be a necessity. Without it, the reasoner could never learn from its mistakes or even its successes.

Military Transportation Planning.

A more recent work in CBP was performed in the domain of military transportation planning and scheduling [Blau et al., 1991]. While this area is not the same as that of autonomous command agents, it is related to command and control and does illustrate important aspects of CBP. For example, the research focuses on domains where multiple agents, with different goals and viewpoints, attempt to plan strategies using incomplete, or uncertain information. The domain of tactical decision-making fits under this definition. In addition, cases in these domains develop over time, requiring reasoning about sequences of events. Military Transportation Planning and Scheduling is somewhat simpler in that all agents involved are under at least partial control of a central agent, and are all working cooperatively. However, the actions in this domain fail frequently due to equipment failures, changing environment conditions, etc. This form of logistics is especially concerned with trade-offs to maximize goal attainment which has direct applicability to tactical decision-making.

Most CBP systems deal with cases consisting of a plan and information about its execution. Blau et al. [1991] focus their attention on the development of a case representation language (CRL) for these domains. The CRL provides for the dynamic aspects of these cases. It is designed to represent cases consisting of the top-level goal(s) and information about states and events [Blau et al., 1991]. The CRL provides for two types of knowledge: general domain knowledge and cases. The general domain knowledge consists of objects, actions, goals, and plans in the domain. The cases consist of situation/solution pairs where the situation consists of the top-level goal(s) and a starting state. The solution consists of the representation of the observable portion of the agent's execution of the plan necessary to satisfy the goals (a network of events and linear sequence of states). The cases are built from instances of the classes in the domain knowledge hierarchies.

Domain knowledge is organized into classification hierarchies, linked to one another via frame slots. Example domain hierarchies include [Blau et al., 1991]:

Object Hierarchy:

C-130 IS A cargo plane
capacity: 42673 lbs
range: 2356 miles

Action Hierarchy:

TAKEOFF IS A DEPARTURE
vehicle: :type AIR-VEHICLE
from-location: :type AIRSTRIP
destination: :type AIRSTRIP
actor:
procedure: (takeoff vehicle from-location)

The goal/plan hierarchy contains an And/Or tree for goal/plan decomposition. A Goal is defined as an OR Node whose inputs are links to alternative plans [Blau et al., 1991].

A plan is defined as an AND Node with links to all required sub-goals. The hierarchy is incomplete because it represents the known partial knowledge of the domain which is why CBP is necessary in the first place. Added cases can also add to this domain knowledge.

Instances of domain objects fill in the slots of frames in these hierarchies. These objects (frames) also contain Part-Of links to connect actions to interpretations and expectation links that provide some partial causal model of the domain. Interpretations are explanations for the occurrence of set of events with a degree of certainty. They may contain some local knowledge such as a goal, objects affected by the interpretation, etc.

Interpretation links can be implicit or explicit. Implicit links are those in which other actions are expected as part of the same interpretation. Explicit links are concerned with different interpretations or actions that are not in the same interpretation, e.g. defensive actions expected in response to an offensive action.

Cases are formed by creating links between the various hierarchies. Event and states are related by causal, temporal, and membership links. Also included are behaviors that can cause a state change and events that the state enables. Goals specify values that a state should hold, i.e. a goal state. For goals that contain sub-goals, fuzzy values are used to represent goal attainment. Events associate action and state information by joining an action with the goal achieved and with the information about the execution of the action. Events contain goals, actions, costs (resources), contexts, and time intervals. They are linked causally to both state changes and other events with a measure of certainty (fuzzy). Events can also be linked to a state which enabled the event. For example, the following would be the representation of a cargo transfer event [Blau et al., 1991]:

```
TRANSFER-CARGO-001 IS A TRANSFER-CARGO
from: WAREHOUSE-001
to: WESTOVER
cargo: (contents WAREHOUSE-001)
goal: '(((location cargo) to))
parts: '(LOAD-001 TAKEOFF-001 LAND-234 UNLOAD-23)
```

Cases are retrieved based upon the description of the situation and the case's similarity to the situation based upon relevant and salient features. The most salient features for initial retrieval are its top level goals and available resources. The similarity is determined based upon the inferential distance of salient features from abstractions in hierarchies. Currently only a strict metric is being used that prefers similarity to parents and grandparents instead of siblings.

Once a case is retrieved, plan adaptation is performed on the parts of the solution which are not applicable or repeatable, because of a lack of resources. These parts are individually adapted by using substitution (replacing a sub-plan for a sibling node in a goal-plan taxonomy), compression's (e.g., eliminating the step from the plan and substituting dependent sub-plans), extensions (e.g., generalizing another sub-plan to cover and replace current one), and recursively performing CBR on sub-plans (which may have been achieved or modified previously) [Blau et al., 1991].

In many cases the plan can be simulated or assessed by other analytical means in order to predict the success or failure of the plan (see Lee [1995]), to find any tradeoffs, to identify possible sources of failures, and to determine the cost associated with the plan. For the military transportation domain, scheduling algorithms and analytical techniques are used [Blau et al., 1991]. Finally, the tradeoffs are accepted or plan is re-modified under a different adaptation rule, or another plan must be selected for adaptation if the retrieved plan cannot be modified.

This work has promise for command reasoning because of its sound implementation foundation. It also

deals with many of the same problems faced in tactical decision-making, specifically goal attainment and tradeoffs. No future work as yet to emerge.

RAND Integrated Simulation Environment (RISE)

CBP was actually implemented in the extension of the RAND Integrated Simulation Environment (RISE) [Catsimpoalas and Marti, 1992]. For RISE, higher echelon behaviors were implemented using scripts describing goal-directed behavior in terms of lower level scripts and so on down to the primitive level. Lower level scripts are created by first analyzing the top level plan and passing the appropriate portions to the lower levels where more planning is initiated. This is done recursively down to the platform level where the behaviors become primitive.

The basis of the RISE work is situational assessment to support planning. The state information and world view of each agent is used to select and implement plans. The scripts are the instantiation of a plan which, in the dynamic military domain, are expected to fail frequently [Catsimpoalas and Marti, 1992]. Thus, the scripts contain contingencies for retrying steps or trying alternate paths instead of costly replanning. However, replanning is performed if necessary. The scripts encapsulate military knowledge and are matched against the current situation. One a similar match is found, they may be modified and are either executed or distributed to subordinate units.

A plan consists of a series of scripts that define the events to be executed. Each step during the plan is composed of goals that may be successful, may fail, or need to be retried. If the successful step was the last of the subplans then the higher level plan also completes. If the subplan failed then the remainder of the subplan is discarded and the higher level reasoners either replan the step or signal the failure up the chain of command. The goal graph built during recursive decomposition can be traversed differently depending upon which goals are failures and which are successful. If necessary the current script is considered a failure and replanning occurs. Replanning occurs when a lower level goal fails but a higher level goal remains valid. When necessary, the lower level goals are recursively replanned up to the highest level if necessary. The script itself is similar to programming statements that are events that mark the passage of simulation time. For example, they may invoke other scripts, perform conditionals, and perform loops.

The scripts are labeled so that the case-based planner can modify the script to conform to the current situation. Portions of the script can be removed or added as needed. The selected and modified scripts become the orders given to the units. They may invoke further planning all the way down to the lowest levels where the scripts contain only primitive actions and planning activities. A small initial case pool is used initially and thus many generic scripts are modified during execution to perform activities. This is a tradeoff between the space and time required to support the enormous number of cases necessary to produce exact matches versus the time required to generate scripts from scratch with no case-base at all.

The situational assessment aspect of the planner concerns the situational geometry of the situation. The situation is defined in terms of a partial ordering among objects of interest in the local environment. The actions include [Catsimpoalas and Marti, 1992]:

- locating the objects of interest.
- rotating all the objects to a coordinate system based upon the line from the object to its objective.
- computing the partial orderings to satisfy the left of, right of, in front of, and behind relations.
- computing the near, far, between, and line-of-sight relations, and a situation description data structure.

The planning involves subordinate objects, their assigned tasks, and the situation geometry. The current situation is not only defined by the situation geometry but also any cooperating units, the object state, future objectives, etc. This description is used to match against the case base. The script associated with the selected case is then modified by the planner and assigned to the object in question.

The RISE CBR method was tested on simple, aggregate level BLUFOR versus OPFOR exercises in which a BLUFOR is trying to reach a destination that is guarded by an OPFOR. The case base initially

only contains case scripts that instruct the units how to fight the enemy, fight for a location objective, and fight a nearby enemy who is not blocking the unit. The results look promising but the experiments are very simple when compared to the tactical decision-making required of upper echelon units in a virtual simulation. Their future work includes expanding the granularity of the case base and more efficient case searching and matching [Catsimpoalas and Marti, 1992].

RPD Architecture

An early implementation of case-base reasoning for command-level tactical decision-making is the RPD architecture [Chaib-draa et al.; 1993]. It is a hybrid architecture design to facilitate the coordination of intelligent agents to promote beneficial interactions and avoid harmful ones. Since local decisions may have global impacts, this becomes an important issue. The architecture is composed of three primary components: a reactive component, a planning component, and a deliberative component, hence the name RPD. The deliberative component handles decision making in completely unfamiliar environments, e.g. when planning fails, and takes into account intentions of others.

Under the RPD architecture, information from the environment is perceived and either a reactive action occurs or a planning action occurs if the information is in the form of a goal. If the information is not perceived as an action or a goal then identification and recognition are needed. If the information is ambiguous or a goal needs elaboration due an unfamiliar environment then deliberative decision making is performed in order to commit to achieving a goal that spawns the planning.

The identification system uses case-based reasoning for situations that the rule based planner cannot identify. It works like an indexing system that uses the perceived information and similarity metrics to discriminate the goal and action information in the memory. The hierarchy between situations is then used to decide which goal or action must be executed. For ambiguous goals or actions, the deliberative component is used.

Using CBP, a command agent must anticipate problems in order to find in memory a plan that avoids those problems. It must then extract this plan and adapt the plan if necessary. If the plan has a flaw then it must repair it by looking for causal explanations of the flaw in order to determine the failure(s). Next, it must determine the strategy necessary to arrange the plan when the cause of the flaw is found. Finally, any successful plans are saved to memory indexed by the goals they satisfy and the problems they avoid.

The identification and recognition module and the planning module for the RPD architecture both use CBR. Both have a memory hierarchy where general cases are represented at the top and concrete cases at the bottom. The identification and recognition module use this memory to identify the current situation and generate the goal that best suits the situation. This can be done in two ways: If the situation is familiar, the recognition sub-module identifies the proper goal using rules that can be dynamically modified. For unfamiliar situations the identification sub-module uses case-based identification. In order to retrieve goals, the memory hierarchy is organized with situations and their corresponding goals. Of course similarity metrics must be determined to compare the situations, but actually, this problem is very specific to each application domain and the authors have not considered any general solution.

The CBP uses the mental state of the agent (beliefs, intentions, and commitments), which are continually modified, to modify its plans. First a plan that was used to accomplish a similar goal under a similar mental state is found. The memory hierarchy links previous mental states of the agent to the plans used in those circumstances. Similarity metrics are also needed to compare the mental states. Once a plan is found, it needs to be modified if it does not exactly meet the requirements of the current mental state.

The repair process is an essential mechanism because the plan may have flaws or be the "wrong" case (i.e., goals not do not apply may be identified for the given situation). This aspect was not investigated by the research. For an intelligent command agent that can continually adapt and learn, case repair is an important process. CBR alone will not solve all the command reasoning problems. CBR and CBP will play a crucial role in the hybrid approaches that will be necessary for autonomous command agents. Unfortunately, the RPD architecture does not provide any implementational evidence to support the

CBR role in agent planning.

Intelligent Agents

CBR has often been used under the context of intelligent agents. Simulating intelligent agents with CBR addresses several command modeling issues, namely anticipation, experiential learning, failure avoidance, goal attainment, and adaptive behavior [Castillo, 1991]. Intelligent agents possess the ability to anticipate, learn, and perceive situations about the real world. In the research by Castillo [1991], an intelligent agent is an object containing rule sets, plans, constraints (resources and requirements of the plan), methods, and a cognitive system. The CBR planner is one aspect of the cognitive system. The cognitive system also contains a sensory filter, problem anticipator, and a rule base containing prediction rules, operating rules, utilization rules, modification rules, and similarity metric rules. Rules in this context are used to support the CBR process and not to provide intelligent behavior on their own.

This work emphasizes specially designed cases to support goal directed reasoning, anticipation, dynamic memory, plan organization, and retrieval [Castillo, 1991]. The intelligent agents contain models of their decision making components and use them when faced with alternatives. They formulate plans and alter plans according to the demands of their environment. As shown in previous work, planning is the most important aspect of an intelligent agent's cognitive activity and provides a mechanism for dealing with a dynamic environment [Wall, 1988; Goodman, 1989; Blau et al., 1991; Chaib-draa et al, 1993; Catsimpooulas, 1992; Castillo, 1991].

Cases are used in conjunction with scripts and rules to provide intelligent behavior. Unlike traditional case-based reasoning, a heterogeneous case base is used containing different formal representations of plans, requests, goals, etc. Each case is represented as an aggregation of these objects such as the goals they satisfy and the problems they avoid. Thus, the case-base is not made up of cases per se, but it is a discrimination network with multiple entry and exit points. The network has with three types of interconnecting semantic relations: logical (specializations/subclasses), structural (aggregation/decomposition of objects), and causal (plans and goals that satisfy or do not satisfy the plan). Any or all of these relations can be used to derive cases. These relations connect the different homogeneous layers of the case base and traditional CBR techniques can be performed for intra-level organization.

The scripts themselves are similar to frames except they represent dynamic behavior in a declarative fashion. They are dynamically altered and hierarchically composed. These are the solution plans that are represented in the case base. They are abstractions of atomic activities that define the actual events comprising the agent's plan. Thus, scripts are the realizations of plans. Plans specify the goals or intentions of the agent. This sequence includes temporal considerations, resource constraints, and trade-off opportunities. The goals are also prioritized so partial achievement of the goals is not necessarily considered a failure. The CBR's role is to match actions that satisfy similar goals. The goals themselves can also have varying fidelity (general to specific) and degrees of decomposition (subgoals). Goals are frames containing semantic links, procedural attachments, and production rules stored in the form of an And/Or graph.

The intentions of the intelligent agent are the driving force behind case selection and are provided in the form of a request. The plans and corresponding scripts chosen are those most likely to achieve the objectives of the request. Case retrieval involves the use of similarity metrics to compare goals, plans, etc. Three metrics are used: a taxonomic similarity metric that measures the closeness in the specialization hierarchy, a feature similarity metric that compares attributes and their values, and an importance metric. Special heuristics are used to compute these metrics for each kind of knowledge unit in the case base. However, before these metrics are applied, an anticipation step is first performed in which failure avoidance is the focus. Failure cases are indexed by the goals they failed to accomplish and contain all the plans, goals, and scripts that were involved in the failure. Any previous attempts to satisfy the same or similar goals are examined to see if those attempts were a success or not. The anticipator attempts to predict future event behavior using domain independent heuristics designed to predict the state of a location or resource at some future point in time. The retrieval step follows. If no exact matches are found the case base is searched for scripts that address the goals specified in the

request and a new case is built. Goals can also be incrementally removed until a match is found and then merged with scripts that satisfy the removed goals. If this is not successful, then the system searches for partial matches. In this situation, different requests, goals, and scripts are substituted in order to find a similar plan in the case-base. When CBR cannot determine a plan, it resorts to learning by inquiry from other agents in the system [Castillo, 1991]. This is an area of further research and suggests, as shown previously, a hybrid architecture.

Once the plan is retrieved it may need to be modified. This can take on several forms: modifying activity durations, removing activities, adding activities, replacing activities, modifying temporal relations, or formulating new behavior. Next, the case is executed and assessed. During plan assessment, new cases are added to the case base as failures or successes. The repair of failures is delegated back to the plan modifier. Performance criteria are used to determine success. They include: total script duration, total resource time, number of unique locations, number of unique resources, number of preemptions, number of precondition violations, number of run-time failures, and percentage of goals satisfied [Castillo, 1991].

A common problem in CBR is transforming the data in the environment into a form that can be used to select cases. The environment, especially the military environment, contains infinite variations in, for example, equipment, positions, and weather, for the same situation. These variations that must be coalesced into a single cohesive case representation. This work 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. This is so the research could better focus on the command and control aspect of the problem rather than the environment translation.

This work was not applied to a real time system so its performance in a command and control environment is unclear. The partial matches were time consuming. This is an aspect of case-based reasoning in general, but it may affect realistic command agent performance. The approach does set the foundation of the architecture necessary for command and control. It should be noted that no one paradigm will be sufficient for command reasoning. A hybrid scheme combining the best of the paradigms mentioned in this report that address the specific segments of cognitive processing will be necessary. Finally, the number of knowledge units needed to perform basic level agent behavior was too large and, therefore, impractical [Castillo, 1991]. The similarity metrics alone required an incredible amount of knowledge. However, these learning agents are a step in the right direction for realistic command agents.

The Judgmental METT-T work mentioned previously used to concept of multiple intelligent agents working together to provide the tactical decision-making of a command agent [Mall et al., 1995]. Agents handle command reasoning aspects such as the mission and enemy concerns. The CBR aspect of this command reasoner was concerned with using CBR to select course of actions when problem situations arose.

Problem situations are generated by other intelligent agents in the system in response to situations such as a bridge out or enemy presence. The alternative generator intelligent agent uses a reduced form of CBR to retrieve solutions from a case base of problem solutions. Problem situations are characterized by their class, type, and origin. Using rules, the solutions are matched first by the problem class, returning general solutions and then by the type that specify specific solutions of the general solution. The agent further discriminates alternate possibilities since some solutions require a specific echelon level in order to be implemented. For example, the problem situation of a tank platoon facing a bridge out on its route would return general solutions such as reroute or breach-obstacle from the case base (since its class is an obstacle). Since the problem is of type bridge-out, more specific alternatives such as find-river-crossing or build-temporary-bridge are retrieved. Since the unit involved is a tank platoon and a bridge platoon is necessary for a bridge, this alternative cannot be implemented. The resources and time required to implement the alternative are also used to eliminate possible alternatives. This work requires exact matches of problem to alternative and will use general alternatives when specific ones do not exist. No attempt to modify cases and store them for future use, i.e. learn, was done. This serves as simply as a portion of a more complex CBR command agent.

APPLICABILITY TO COMMAND DECISION MODELING

Tactical decision-making is based upon experience and involves uncertainty. Since there are patterns and similarities in the conduct and results of battles going back to 500 BC, a case-based approach seems applicable. However, the lack of robust implementations does not support (or discredit for that matter) its applicability. Like the work in Wall et al. [1988] there is no reason that with present technology it could not be adapted for an automated command reasoner.

Some problems to be faced by CBR for command agents include:

- transforming the environment into a situation description that facilitates the meaningful retrieval of cases. The infinite varieties of the environment make this a difficult task.
- selecting proper indices and indices for failures to facilitate learning from mistakes.
- case base organization. How the case base is decomposed, indices applied, and case links affect the case retrieval efficiency and correctness.
- being unable to capture deep knowledge. This makes it impossible to indicate reasons why certain actions are performed and to incorporate this reasoning into the decision making process.
- difficulty in capturing complex tactics with cases. In some instances, when slightly different situations required drastically different actions, there could be difficulty in providing enough cases to fully characterize the domain. Difficulties with case mismatching must be resolved through the careful selection of the features used in matching.
- As mentioned previously, a hybrid architecture is probably required to model all the aspects of a command agent. In whatever architecture is selected, CBR will most certainly play a role. The ability of CBR to match against previous similar situations to provide actions or plans and modify these results and store them for later use is a requirement for command decision modeling.

REFERENCES.

Blau, Lauren; Bonissone, Piero P.; Ayub, Saad, "Planning with Dynamic Cases", *Proceedings of a Workshop on Case-Based Reasoning*, Washington D. C., May 8-10 1991, pp. 295-306.

Castillo, David, *Toward a Paradigm for Simulation Intelligent Agents*, Doctoral Dissertation, University of Central Florida, 1991.

Catsimpoolas, Niels; Marti, Jed, "Scripting Highly Autonomous Simulation Behavior Using Case-Based Reasoning", *Proceedings of the 25th Annual Simulation Symposium*, University of Florida, DARPA, Army Research Office, Gainesville, FL, December 7-9 1994, pp. 13-19.

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.

Dupuy, T. N., "Military History and Case-Based Reasoning", *Proceedings of a Workshop on Case-Based Reasoning*, Clearwater Beach, Florida, May 10-13, 1988, pp. 125-135

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

Goodman, Marc, "CBR in Battle Planning", *Proceedings of a Workshop on Case-Based Reasoning*, Morgan Kaufmann: San Mateo, CA, Pensacola Beach, Florida, May 31- June 2 1989, pp. 264-269.

Kiersey, 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. 307-315.

Kolodner, Janet, *Case-Base Reasoning*, Morgan Kaufmann Publishers, Inc. San Mateo, CA, 1993.

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. 220-235.

Mall, Howard; Bimson, Kent; 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.

Wall, Rajendra S.; Donahue, Dan; Hill, Stan, "The Use of Domain Semantics for Retrieval and Explanation in Case-Based Reasoning", *Proceedings of the 1988 DARPA CBR Workshop*, Morgan Kaufmann: San Mateo, CA, Clearwater Beach, FL, May 10-13 1988, pp. 447-462.

[Return to the Top of this Section](#)

TAA4 ***TECHNOLOGY AREA ASSESSMENT:***

GENETIC ALGORITHMS

AND

EVOLUTIONARY PROGRAMMING

ASSESSOR:

Rick McKenzie, Ph.D.

Science Applications International Corporation
3045 Technology Parkway
Orlando, Florida 32826-3299
(407) 282-6700
Rick_McKenzie@cpqm.saic.com

OVERVIEW OF TECHNOLOGY AREA:

BACKGROUND:

Genetic algorithms (GAs) have been around for a number of years. Interest in genetic algorithms began in the 1970s when John Holland introduced what is referred to as his Simple Genetic Algorithm (SGA) and gave a complete architecture and implementation technique [Srinivas & Patnaik 1994]. In the last few years, they have begun to gain popularity as effective optimization and search methods for large, difficult search problems where the search space is non-convex. In general, non-convex problems are known to be intractable with only those problems with few variables being solvable by traditional

optimization methods such as gradient descent [Vavasis 1995].

Evolutionary programming (EP) is closely related to genetic algorithms. Both are based on the survival-of-the-fittest principle accredited to Darwinian theories of evolution and natural selection. These techniques start with an initial set of data and then iteratively change the data in order to approach solutions to a particular problem. Therefore, both GAs and EP have proven to be robust optimization methods. The difference being that GAs rely heavily on *crossover* while evolutionary programming emphasize *mutation*. Crossover and mutation are functional operators that are performed on a *population* of data sets. These operators get their names from science of genetics. In reproduction, chromosomes routinely exchange genetic code as part of a natural process. This genetic code comes from the male and female strings of chromosomes. The exchanging of genetic code is called crossover. The resulting child is a new data set which enters the population. In GAs, the selection of chromosomes for crossover is carefully controlled so that promising strings are selected to have more offspring than other strings. Mutation is also a somewhat frequent occurrence in nature. Mutation is used in GAs because crossover techniques using better strings to influence a population will eventually create a population where a majority of strings contain much the same genetic make-up. Therefore, new radical looking strings become impossible and the population could converge around a local optima. Mutation makes it possible for radical strings to be introduced and a global optima to be reachable.

A population is simply a finite set of solutions in the search space where an infinite number of solutions may be possible. These techniques are intended to find the optimal solution in this search space by iteratively modifying the population by increasing the number of promising solutions in the population and removing or decreasing the number of less promising solutions. A promising solution is decided according to a fitness function which is used to rank the solutions. A fitness function is simply the normalized version of the objective function which is the representation of the problem to be solved. In GAs, each solution may be likened to a string of chromosome in that it is made up genetic code which is shared or changed to make new solutions that enter the new population. In essence, GAs search for the sequence of genetic code that provides the globally optimal solution to the objective function. Evolutionary programming also searches for a globally optimal solution. However, it does so based on the behavioral traits passed down from parent to child, *phenotype*, rather than the genetic code, *genotype* [Fogel 1994]. Genotypes represent the blueprint of an individual. Phenotypes represent how the items of the blueprint are manifested in the individual.

A notable technique that also mimics physical processes to perform heuristic search is called simulated annealing. This technique models the methods used to strengthen metals by leveraging the thermodynamic properties of materials. The metal is heated and cooled in a particular way to achieve the optimal alignment of atoms in the material. In contrast to GAs, simulated annealing is performed on one string which is replaced by a new string if the new string is better. Since, there is only one string, it is not possible to apply the crossover operator. It is possible for a worse string to replace a better string. The probability of this is based on the current temperature of the process. Higher probabilities for higher temperatures. The temperature starts

high and then the process undergoes a cooling schedule. The temperature is lowered when the best string has been modified a number of times without finding an improved string. The process is terminated when the temperature is reduced a number of times without finding a better string.

DESCRIPTION OF THE TECHNOLOGY:

ANATOMY OF GENETIC ALGORITHMS

The basic algorithm for genetic algorithms is illustrated below.

```

Initialize population (arbitrarily or otherwise)
Calculate fitness values of population members
LOOP While (Termination Criteria is Not Reached)
    Apply selection criteria
    Generate offspring
  
```

```

    Apply crossover operator
    Apply mutation operator
    Generation = Generation + 1
    Calculate fitness values of population members
END LOOP

```

Encoding Mechanism

Before this algorithm can be applied, a proper encoding mechanism for the strings must be selected. Traditionally, according to Holland's SGA, the GA is best optimized by a binary encoding scheme which imparts a characteristic known as implicit parallelism [Fogel 1994]. The idea is that based on the fitness value for a particular schema of binary code, one can derive information about the fitness value of strings matching that schema. A binary encoding maximizes the number of strings in the search space that would match the schema. However, this has been found not to be the case for all optimization problems and is certainly a candidate for further study.

If one were to use the binary encoding method, a simple mapping from variable values to manageable integers would suffice to create the needed concatenated string of binary numbers.

Selection and Offspring

During selection, the survival-of-the-fittest principle is applied. Each string is assessed a fitness value according to the fitness function. Strings with better fitness values are given a higher probability to produce offspring than those with lower fitness values. Therefore, the strings with stronger gene sequences will pass them on to serve as building blocks towards an optimal solution. The proportionate selection scheme is a popular method for generating offspring. Using this method, the number of offspring for a particular string is determined by its fitness value divided by the average of all the fitness values of the population. The resulting number may be rounded as needed.

It has been shown that Holland's SGA without modification will not find a global optima [Gunter 1994]. Therefore, minor modifications to its mutation and selection strategies are needed to make convergence possible. The adoption of an elitist strategy is a sufficient change. An elitist strategy simply retains the best solution in the population from generation to generation. In this way, optimal solutions are not destroyed by crossover.

Crossover

Crossover is performed on two randomly selected strings based on a predetermined probability referred to as the crossover rate. The three main techniques for crossover are single-point, two-point, and uniform. In single-point crossover, the strings are broken at a randomly determined location and then the remainder of each string is switched with the other to create two new strings. This introduces biases since bits at the ends of strings tend to change more frequently than bits towards the middle of the strings. Two-point crossover is used to remove this bias. Strings are segmented using two randomly chosen locations and the segments are then switched between the two strings. Uniform crossover dictates that each bit location in the pair of strings has an equal probability of being switched. The drawback of this method is that it tends to break up the building blocks of genetic sequences that strive toward optimal solutions. Since whole segments are moved in the one-point and two-point methods, these building blocks are preserved.

Mutation

Each bit of every string is subjected to a probability of mutation. This probability is called the mutation rate. In GAs, the mutation is usually very small in comparison to crossover rates. In the case of binary in coded strings, the mutation of a particular bit means to simply change it from a 0 to a 1 or vice-versa.

CONTROL PARAMETERS & TERMINATION CONDITION

The control parameters of GAs are the population size, crossover rate, and mutation rate. Other operators besides crossover and mutation may also be applied to the strings and, therefore, would also be a source of control. Typical population sizes range up to 200 while crossover and mutation rates range

from .5 to .95 and .001 to .05 respectively. Termination conditions are based on reaching a specified number of generations, on attaining a string with a desired high fitness value, or on achieving a considerable amount of homogeneity among the strings in the population.

NOTABLE IMPLEMENTATIONS AND VARIANTS

The popularity of evolutionary programming and especially genetic algorithms have increased in recent years. Much research has been done in many areas including operations research, robotics, and Very Large Scale Integrated (VLSI) circuit layout.

Because GAs provide good optimization techniques, they have been widely applied in operations research endeavors. NP-hard problems such as the traveling salesman problem (find the minimum distance for traveling between a number of cities) are favorably solved using GAs. These operations research problems provide practical use for resource constrained situations common to military logistical elements.

In the robotics industry, GAs are typically used for path planning and pattern recognition. Where efficiency and precision is a premium, GAs select the optimum balance of manipulator movement and torque reduction. Likewise, efficient routing of chip circuitry in VLSI designs are important in reducing cost.

TECHNICAL DETAILS:

Genetic algorithms and evolutionary programming are iterative techniques for optimization and search. As such, these techniques may be very slow to converge to global optima and may not even attain an optimal solution. However, they are good at solving non-convex optimization problems that are generally intractable. The good news is that these techniques may be implemented in a parallel fashion.

APPLICABILITY TO COMMAND DECISION MODELING:

POTENTIAL AS A PRIMARY OR SOLE TECHNICAL APPROACH:

Genetic Algorithms and Evolutionary Programming rely on optimization of objective functions. Complex objective functions with many variables may result in a non-convex search space and non-convergence. The potential complexity of command decision models and the near real-time, reactive nature of command agents make the use of GAs and EPs very unlikely integrators of command agent architectures.

SUBPROBLEMS ESPECIALLY SUITED TO THIS TECHNOLOGY:

As shown in current work with robotic and VLSI systems, GAs can be used to perform path planning activities that would be of use to a command agent. Pellazar (1994) uses GAs for vehicle route planning. Other planning activities such as operational and tactical planning should also be considered. As an off-line strategy for developing operational and tactical techniques, evolutionary algorithms could be useful in maximizing effectiveness while minimizing time. An example of this would be to consider a mission to be a solution to an objective function that characterizes the effectiveness of winning a battle or resupplying a unit. The possible tasks in the mission may be mapped to a binary encoded string and evolved using a GA. New solutions may be evaluated based on the sequence of tasks executed via a

simulated unit. In this manner, new doctrine may be developed for the unit of the 21 century.

Recent work by Adamson and Joshi (1996) used the Close Action Environment (CAEN) simulation to study behavior changes in posture, speed, sensors, target acquisition, and types of fire. The fitness function used for a string of activities carried out in this simulation is based on the number of friendly forces killed and the time taken to reach the objective. Iteration of the GA continues until a stable sequence of activities is achieved. The researchers used a fairly restricted number of variables and tasks but illustrated interesting and successful use of GAs in the development of tactical behavior. Another example of using GAs to develop combat behavior can be found in [Smith 1995].

Fogel et. al. (1996) also presents a similar application using evolutionary programming. This innovative approach dynamically modifies mission plans in ModSAF to adapt to a changing environment. This is accomplished by mutating a population of mission plans for units and evaluating the population against an objective function that is based on a number of probabilities such as the probability of a kill between two vehicles. Once an optimized population is found, these mission plans are reassigned to the units. This sequence is followed throughout one exercise. One can certainly argue about how well these probabilities approximate the battlefield situation and whether effective optimization can be achieved in this manner but the authors state that results show successful mission adaptation on the simulated battlefield. This technique could be modified evaluate exercise results and therefore evolve optimized behaviors as in [Adamson 1996]. However, the execution of that method is sure to be time consuming.

WEAKNESSES AND CONCERNS:

As mentioned, the solutions to optimization problems provided by these evolutionary techniques may not converge to a global optimum. Can the command agent make use of sub-optimal solutions?

CONCLUSION.

Genetic algorithms and evolutionary programs are not viable options for the representation of command agents. However, these techniques may be used in a near-real-time supporting role for the development and analysis of new and existing command and control procedures, the optimization of logistical functions, and the planning of unit routes.

REFERENCES.

- Adamson J. and Joshi, K.G. "Genetic Algorithms and Force Simulation." In *Proceedings of the Sixth Conference on Computer Generated Forces and Behavioral Representation*. Orlando, FL. July 23-25, 1996. pp. 237-242.
- Andriole, Stephen J. "Information Technology for Command and Control." *IEEE Transactions on Systems, Man, and Cybernetics*. Vol. 16, No. 6. November/December, 1986. pp. 762-764.
- Fogel, David B. "A Comparison of Evolutionary Programming and Genetic Algorithms on Selected Constrained Optimization Problems." *Simulation*. Vol. 62:4. June, 1995. pp. 397-403.
- Fogel, David B. "An Introduction to Simulated Evolutionary Optimization." *IEEE Transactions on Neural Networks*. Vol. 5, No. 1. January, 1994. pp. 3-11.
- Fogel, Lawrence J.; Porto, V. William; and Owen, Mark. "An Intelligently Interactive Non-Rule-Based Computer Generated Force." In *Proceedings of the Sixth Conference on Computer Generated Forces and Behavioral Representation*. Orlando, FL. July 23-25, 1996. pp. 265-271.
- Goldberg D.E. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley. Reading, MA. 1989.

Gunter, Rudolph. iConvergence Analysis of Canonical Genetic Algorithms.i *IEEE Transactions on Neural Networks*. Vol. 5, No. 1. January, 1994. pp. 96-101.

Holland, John H. *Adaptation in Natural and Artificial Systems*. The University of Michigan Press. Ann Arbor. 1975.

Pellazar, Miles B. iVehicle Route Planning with Constraints Using Genetic Algorithms.i In *Proceedings of the IEEE National Aerospace and Electronics Conference (NAECON)*. Vol. 1, 1994. pp. 111-118.

Smith, Robert E. and Dike, Bruce A. iLearning Novel Fighter Combat Maneuver Rules via Genetic Algorithms.i *International Journal of Expert Systems*. Vol. 8, No. 3. 1995. pp. 247-276.

Srinivas, M. and Patnaik, Lalit M. iGenetic Algorithms: A Survey.i *IEEE Computer*. June, 1994. pp. 17-26.

Vavasis, Steven A. iComplexity Issues in Global Optimization: A survey.i *Handbook of Global Optimization*. R. Horst and P.M. Pardalos, Eds. Kluwer Academic Publishers. Netherlands. 1995. pp. 27-41.

[Return to the Top of this Section](#)

TAA5 TECHNOLOGY AREA ASSESSMENT:

NEURAL NETWORKS

AND

BOUNDED NEURAL NETWORKS

ASSESSORS:

Howard Mall, M.S.C.S. and Dirk Ourston, Ph.D.

**Science Applications International Corporation
3045 Technology Parkway
Orlando, Florida 32826-3299
(407) 282-6700
Howard_Mall@cpqm.saic.com
Dirk_Ourston@cpqm.saic.com**

OVERVIEW OF THE TECHNOLOGY AREA:

In the field of Artificial Intelligence the neural networks implemented in hardware or simulated with software are artificial constructs which are based on theories of how biological neural nets behave. For the sake of precision and the conventions of the domain, these computational constructs shall be referred to as Artificial Neural Networks (ANNs) to distinguish them from other neural networks. ANNs encompass a very rich and controversial field of computation that is now finding application with the engineering of robust tools.

DESCRIPTION OF THE TECHNOLOGY:

The DARPA Neural Net study states, "[a] Neural Network is a system composed of many simple processing elements operating in parallel whose function is determined by network structure, connection strengths, and the processing performed at computing elements or nodes." [] This accurately describes some of the aspects of Neural Networks that enable their two main features. Namely, that ANNs are adaptive and inherently massively parallel.

ANNs take the approach of combining the very simple behavior of its simple nodes to create more complex and generalized capabilities. Many data are input to ANNs which processes these inputs to produce desired outputs. The outputs are modified based on the flow of this data through the weighted pathways between each neural node. The computation of this data is implicitly captured within the combination of simple calculations that are carried out within the node of the ANN.

These structures are developed by "training" the ANN with a representative set of data. The strategy by which the net is trained is one of the differentiating factors between implementations of neural nets. The adaptive details of the ANNs are expressed by "some sort of 'training' rule whereby weights of connections are adjusted on the basis of data." [] ANNs learn from examples and have an ability to generalize from these examples. The ANNs extrapolate to formulate correct outputs when presented with data outside of that with which it was trained.

ANNs perform best on problems that are data-rich and knowledge-poor. This makes them well suited to problems involving classification or approximation of solutions. ANNs strategies are analogous to statistical techniques and thus introduce some imprecision. They are especially useful when explicit methodology (such as rules in an expert system) cannot be easily derived from the domain. This feature is readily apparent in trying to capture cognitive processes like command decision modeling.

BACKGROUND:

Neural network research originated from the biological structures observed in animal brains. ANNs are loosely based on these structures. *Axons* grow out from the neuron's *soma*, or cell body, and make a special connection, called a *synapse*, with the *dendrites* of other neurons. Axons send electrical signals which are received by dendrites.

The synapse creates a chemical gap between two neurons. When a synapse receives a signal from the axon it releases chemicals called neurotransmitters that can change the electrochemical potential of the synapse. Repeated impulses through the synapse improves its effectiveness. This pathway from one neuron to another gets reinforced making it a more probable impulse than others. The synapse "learns" to always allow the transport of an impulse to occur in this manner. Patterns of combinations of reinforced synapses are believed to form memory in biological systems.

ANNs work in very much the same way. They are composed of a very large number of processing elements that have massively parallel interconnections. Part of the neural network community "regards massive parallelism with high connectivity to be defining characteristics of neural networks." [ii] They are known as connectionists. The computation "engine", if you will, of an ANN is very general, simple, and distributed throughout an interconnected structure. Their power lies in the ability of the ANN to do statistical inference. In fact, many ANN approaches are analogous to statistical methods of data analysis. This means that they are capable of "learning to generalize from noisy data" [ii].

The history of ANN research has been a striving to find ways of solving problems in which the approach to finding the solution could not be easily gleaned. This led to the concept of applying a very simple processing technique which could be distributed into a network. The inherent properties of the system could capture the solution method through a "training" process which exploited the interaction of the distributed processors. ANNs proved that they could learn and generalize from sets of training data and expected results, but the critics argued that there was no way to determine **how** the ANN was solving its given problems. No rules or computational processes could be extracted from the ANN. It could learn how to solve a particular class of problems, but it couldn't teach it to the human beings who had made it.

ANN research went through three phases of flourishing advancement followed by lulls of nearly two decades. Many different approaches to constructing ANNs have been tried over their long history, but it is not until the past few years that an understanding of what ANNs can and can't do has lead to their application in commercial, civil, and military projects. Table 1 shows a brief matrix indicating the eras of Neural Network excitement and the developments that occurred in those times.

EARLY WORK <i>1940-1960</i> <i>Fundamental</i> <i>Concepts</i>	McCulloch & Pitts	- Boolean Logic
	Hebb	- Synaptic Learning Rule
	Farley & Clark	- First Simulation
	Rosenblatt	- Perception
	Steinbuch, Taylor	- Associative Memories
TRANSITION <i>1960-1980</i> <i>Theoretical</i> <i>Foundations</i>	Kohonen	- LMS Algorithm
	Albus	- Cerebellum Model (CMAC)
	Anderson	- Linear Associative Memory
	Von Der Malsberg	- Competitive Learning
	Fukushima	- Neocognition
RESURGENCE <i>1980-...</i> <i>Theory</i> <i>Biology</i> <i>Computers</i>	Grossberg	- ART, BCS
	Kohonen	- Feature Map
	Feldman & Ballard	- Connectionist Models
	Hopfield	- Associative Memory Theory
	Reilly, Cooper, et al.	- RCE
	Hinton & Sejnowski	- Boltzmann Machine
	Rumelhart et al.	- Back Propagation
	Rumelhart & McClelland	- PDP Books
	Edelman, Reeke	- DARWIN III

Table 1: **Neural Network History (from [i])**

The seminal thinking in neural networks began in the 1940's with the work of McCulloch and Pitts [1]. Rosenblatt carried this work into the 60's with the development of the single-layer perceptron and its attendant convergence theorem. Minsky and Papert cast the field into its 20 year "dark age" by pointing out the limits of the single-layer perceptron (most significantly that it could not solve the 'exclusive OR' class of problems). The interest of computer scientists in ANNs atrophied.

ANNs did not reemerge until the 1980's when breakthroughs with multi-layer neural networks, computing resources, and the understanding of neuro-biological systems were made. The renewed interest has continued today spurred by the development of the multi-layer perceptron with back-propagation and Hopfield energy reduction learning algorithms.

ANNs are suitable for a variety of data processing tasks. These have been categorized as: []

- **Pattern Classification:** Input patterns are assigned to pre-specified classes for which it has been trained. Neural networks have been applied in this manner to do character and speech recognition.
- **Clustering or Categorization:** Input patterns are compared with other patterns. Similar patterns are clustered into groups. This type of network has not been trained with a set of data as with the classification task. It is applied to advanced data processing applications such as mining, compression, and exploratory analysis.
- **Function Approximation:** This task takes a set of training input-output pairs and estimates a function that will approximately generate the expected outputs with the inputs with which it was trained. This has application to all sorts of scientific and engineering modeling problems.
- **Optimization:** ANNs can be trained to find outputs for given inputs which satisfy a set of constraints that are assessed by the maximum or minimum of value of a given objective function. There are a wide variety of problems in the world in which optimization is applied to examine the trade-offs between different variables. Economics, statistics, engineering design, and computer science (the Traveling Salesman Problem) are fields that employ this strategy.
- **Prediction or Forecasting:** ANNs can be trained to extrapolate the trend in a given set of input data, determining what sample data is expected at a future time. This can be used for tasks from stock market projection to weather forecasting.
- **Content-Based Data Retrieval:** This is a significant task that is of extreme importance in visual or aural identification tasks. The ability to implement an associative memory using a neural network is a very powerful application. It does not have to maintain an explicit address to a memory location, as in the von Neumann architecture, but can retrieve data event with partial or noisy input query.
- **Control:** ANNs can carry out process control tasks by producing control directions based on some input feedback from system it is controlling. This is especially useful when a control function is non-linear and not trivially derived from the system's design or it needs to adapt to unexpected changes in the system's performance.

The capabilities of ANNs suggests many applications where they might be useful. Some areas to which ANNs have been applied are: []

- Robotic Control,
- Electrical Load Forecasting,
- Speech Recognition,
- Face Recognition,
- Classification of Medical Imaging,
- Signal Processing,
- Playing Competitive Backgammon, and
- Modeling of Human Decision Making.

Neural nets appear to have many wide applications that have suitability to the problem of human command decision modeling. Their strength is their ability to acquire knowledge by being trained in which the knowledge engineering task is inherent to the construction of ANNs. However, describing this knowledge in tractable ways is extremely difficult if not impossible. The ANN can produce appropriate output to selected inputs but the how it accomplishes this task is hidden within the complexity of its widely distributed network of processors. But, some new techniques are now being experimented with in which rules or processes can be derived from the 'learning' that neural networks. A capacity to 'teach' is being incorporated into emergent ANN systems.

NOTABLE IMPLEMENTATIONS AND VARIANTS:

Implementations of ANNs have been discussed previously for general classes of problems. This section will discuss specific simulation systems and studies that have utilized neural network technology. Primarily military applications have been examined, but work that has definite bearing on the problem of Computer Generated Forces (CGF) and indicates a possible area of technology transfer will be discussed as well.

Military Applications of ANNs have run the gamut of automatic target recognition to sensor fusion activities. However, all of them suggest solutions or partial aspects of modeling command decision making. The following implementations are discussed as they apply to Command and Control or planning activities that are the realm of military commanders.

The National Test Facility (NTF) was established to support Strategic Defense Initiative Organization simulations and experiments. At this facility an application of ANNs was designed to rapidly assign weapons to targets in a large-scale simulation. The system simulated a variety of interceptors designed to remove both Inter-Continental Ballistic Missile and strategic ballistic missile targets from the sky. This kind of planning requires both the use of resource optimization and physics routines. The physics routines are highly complex and computationally expensive, especially if the type of interceptor that was being evaluated was in earth orbit. To assign the optimal number of interceptors to unique targets over several time periods "was posed as a 50,000 variable problem." [] The neural network and supporting hardware designed for this simulation out performed many algorithmic statistical optimization techniques.

The NTF study shows the ability of neural networks to rapidly arrive at solutions. This is due to the fact that the distributed computational properties of ANNs are simple and inexpensive, however neural net optimization is approximately precise. However, for the modeling of command decisions they can be trained to be "good enough", especially in the "fuzzy" information space of the battlefield. The NTF work represents one aspect or component of simulating a human commander, just one part of making decisions within the constant flux of the battlefield.

Research conducted at the TRADOC Analysis Command (TRAC) implemented a system that applied ANN technology as a way to aid the building, maintenance, and testing of large knowledge-bases that are prevalent in the system family of Distributed Interactive Simulation (DIS). Knowledge acquisition is labor intensive and often difficult to elicit from Subject Matter Experts (SMEs). The TRAC work has attempted to demonstrate an ability to utilize closed-loop, reinforcement learning neural nets to "conduct automated development of knowledge bases." [] "The unique characteristics of that model include the representation of complex goal functions, an extensible architecture, truly autonomous computing units, and the ability to learn temporal relationships." [vii]

The basic approach of the TRAC system was that during simulation it would use an expert system when the situations were familiar. The neural network would be used to improve the rule-base when situations were not familiar. This work indicates another way that ANNs can be applied to the modeling of command decision-making. It allows the system to adapt to changing situations and to derive knowledge from unusual situations and incorporate them using traditional methods of reasoning.

The work at BDM Federal, Inc., relating to the DIS tool study for the Army Research Institute, sought to tackle the exact problem of "simulating the human characteristics of decision making." [] Modeling the capabilities of human commanders to deal with incomplete, ambiguous, and contradictory information in a creative manner was the goal of their project. They observe that humans make decisions based largely on experience through association of past decisions with current ones.

Their approach was to divide the structure and data of decisions into four major components. The explicit data of both the intrinsic, context-sensitive facets of the decision and the extrinsic, environmental characteristics of a decision are fed into a model of decision-making behavior (in this case an ANN). The processing of this information produces a decision and meta-information for applying the results of this unique decision within the simulation. The ANN is trained with a representative set of cases that span all class of decision the system is expected to make. The neural network retains the experiential knowledge and classifies them based on a small number of variables.

"The neural network provides the decision-making memory, non-linear responses, association and classification skills, and the ability to operate in ambiguous and contradictory environments that normally distinguish human behavior from rule-based computer simulations." [viii] The approach of this work uses the associative memory capabilities of neural networks to provide a high-performance case-based retrieval technique. A case-base whose structure is 'learned' by the ANN and provides original decisions by generalizing from its training data. This is a good initial effort and suggests future work in this area.

Pathfinder Systems developed a rich collection of models for applying ANNs to simulating battlefield command and control.[] The people there created the Linear Interactive Activation and Competition (LINIAC) model to construct ANNs. They implement a collection of decision objects, each a modular ANN designed to make a specific decision, into a structure called the ROLEPLAYER model. Using this model simple ANNs are combined to form more complex decision-making behaviors. Pathfinder developed a graphical user interface for interactively creating and training LINIAC ANNs called the Course of Action Planner (COAP). COAP facilitates the development of ANNs by personnel with little software development experience.

The work at Pathfinder Systems simulated commanders from platoon leader to brigade staff. Their model could be utilized as a standalone autonomous system or as a decision component within a larger, hybrid system. They "implemented neural networks to perform intelligence evaluation, threat evaluation, combat evaluation, mission evaluation, self-intention decisions, operations decisions, and next objective decisions"[] within the EAGLE simulation for the WARSIM 2000 Testbed at the MITRE corporation. The power of their work was the use of modular neural net objects that could be organized within a hierarchical decision structure making testing and development much easier. Another useful concept was to supply a tool to non-computer experts so they could create and train the ANNs to form behaviors through "battle rehearsal"[x].

One of the weaknesses of ANNs is their lack of semantic representation or symbolic reasoning. ANNs deal with numerical values and weights using simple functions to determine the processing and propagation of data throughout its network connections. Knowledge-based approaches are rich in information about the problem domain to which they are being applied but they lack the adaptability of ANNs. In [] the researchers attempt to ameliorate these problems by combining the capabilities of neural networks and expert systems.

Their approach is to create a Neural Logic Network (Neulonet), an ANN with special structure and activation function that allows symbolic rules to be explicitly encoded within a neural net. The developer builds sets of Neulonet rules which are equivalent to expert system rules created by a domain expert. The Neulonet rules are chained to form the larger network. The resulting structure is closer to the expected solution and makes logical sense to the user. The generalization, adaptive, and opportunistic qualities of neural networks can then be applied to create a system that closely matches the human decision-making behavior that includes bias and intuition. The work in [xi] is a representative example of many different projects which are attempting to bridge the gap between neural networks and knowledge-based systems.

TECHNICAL DETAILS:

ARCHITECTURES:

ANNs have a variety of ways in which they can be structured. The manner in which data is propagated through the structure has two possible structure: **feed-forward** where data pass from one layer of nodes to another until the end of the path is reached or **feedback (or recurrent)** where data can be returned to the node through some kind of loop.

The interconnections of ANNs can also define its structure:

- **Fully connected** ANNs have every processing element connected all other processing elements.
- **Sparsely connected** have processing elements which are only connected to a few other processing

elements.

The propagation of data through the neural network will define its ability to learn. Feedback loops can increase an ANN's capacity for adaptability. The interconnections of the processing elements within an ANN define the degree of parallelism of the system. This has implications to the level of sophisticated behavior the ANN may be capable of emulating.

LEARNING:

ANNs have an ability to be trained with representative data of the decisions they will be expected to make. For this reason ANNs require no large knowledge base and the concomitant knowledge engineering which is required to construct one. This reason makes ANN technology an important area to explore for the domain of command decision-making.

The learning techniques which can be applied to instruct ANNs are the following:

- **Supervised training** is a technique in which the neural networks responses to data are corrected by an external teacher, typically a human being.
- **Unsupervised training** allows the classification capabilities of neural networks are exploited. The ANN learns by comparing all of the representative examples it is given.
- **Reinforcement training** is where the ANN teaches itself by internally monitoring its performance through several iterations of input data.

TYPES OF ARTIFICIAL NEURAL NETWORKS:

Single-layer Perceptron

The work of Rosenblatt on the single-layer perceptron created the first examples of ANNs. This network was arranged as simple set of interconnected processors arranged in one layer. The network was applied to the problem of classification and pattern recognition [], "labelling data items in classes according to their attributes" [Error! Bookmark not defined.]. This classical approach attempts to discover a linear decision boundary separating two classes of data. [] When there is no linear boundary the single-layer perceptron fails. It fell into disrepute when Minsky and Papert showed that the single-layer perceptron could not solve the 'exclusive OR' problem. However, later research returned to this approach and incorporated multiple layers.

Multi-layer Perceptron

The multi-layer perceptron was an improvement on the work the Rosenblatt had begun. This ANN organizes perceptrons into connected layers. The multi-layer perceptron is a feed forward network that employs supervised learning. The training of this type of ANN is in two phases [xiv], the first in which the network node output is computed and in the second "weights are adjusted with supervision to provide the desired output during training" [i]. One important feature of the multi-layer perceptron that was added was its ability to generalize. This means it can produce correct output for input data for which it was not trained.

Radial Basis Function

The radial basis function network is very similar to a multi-layer perceptron. A multi-layer perceptron network's function "may be viewed as reducing the dimensionality of the data- projecting a large input space on to a smaller number of hidden units and forcing the data through a bottleneck. The radial basis function does precisely the opposite. [Error! Bookmark not defined.]" The functions employed by nodes within the layers of the network are linearly combined with that of the output nodes. [xiv] In this design the nodes can respond locally to input data. Within a multi-layer perceptron there is not connection between internal nodes of its different layers. Another basic difference is the functions that are used by the nodes in the radial basis function network are Gaussian versus sigmoidal as in the multi-layer

perceptron.

Kohonen's Self-Organizing Feature Map

Kohonen's Self-Organizing Feature Map employs a competitive learning algorithm. The network becomes divided into neighborhoods of computation output. The nodes that provide the closest match with the intended output (i.e. the winner) and their neighbors are reinforced iteratively. The network begins to converge toward a correct solution that not only produces the correct output but can also be mapped to topological areas of the neural network. The factors that impact the performance of an ANN of this type "include the dimensionality of the neuron array, the number of neurons in each dimension, the shape of the neighborhood, the shrinking schedule of the neighborhood, and the learning rate." [Error! Bookmark not defined.]

The Adaptive Resonance Theory Model

Adaptive Resonance Theory (ART) networks are "non-linear recurrent networks with feedback connections" [i]. This design was developed to overcome the problem of the *stability-plasticity* dilemma ([Error! Bookmark not defined.]). The architecture tries to ensure that existing knowledge is not erased or corrupted as a competitive learning ANN acquires new knowledge. The learning algorithm will only update those clusters of the network that *resonates* with (is sufficiently similar to) the input data.

The extent of similarity between the input data and any part of the neural network is measured by a *vigilance parameter*. If there is no sufficient match then the ANN will create a new category, a new cluster within the network from a set of unused nodes that were set aside. This type of ANN design could be unbounded, instantiating new output nodes as it gains more and more data, except for the limits of computer hardware.

The Hopfield Model

The Hopfield network is a completely connected, recurrent neural network that was applied to the associative memory problem [Error! Bookmark not defined.]. This kind of ANN utilizes an energy function which is applied throughout the network to form areas of minimum energy. These minima become stable as network energy decreases. They are called attractors that can act as content-based indexes from which to retrieve larger patterns from the ANN. Combinatorial optimization problems can also be formulated as a minimization of network energy. Variations of this ANN are the Boltzmann machine, which uses simulated annealing from statistical thermodynamics as its energy minimization function, and the Mean-Field-Theorem ANN [xiv] which dynamically alters the interaction of energy fields within the network.

APPLICABILITY TO COMMAND DECISION MODELING.

The initial work (see above) that has been done with ANNs suggests that they can provide a high-speed generation of decisions based on significant amounts of data. They can generalize about the data they receive, capable of non-linear reasoning. They are capable of adapting to changing situations- altering their behavior in regard to new events. They can classify, categorize, optimize, and project. All are qualities that a human commander must exhibit. Most significantly, ANNs are trained. There is no significant amount of knowledge engineering that accompanies their construction. Due to the significant amount of manpower required by knowledge engineering neural nets could provide a cost effective solution to CGF simulations. In regards to modeling human behavior by training an ANN, it acquires the biases of the training data which it is fed.

However, there are some detractors to be considered. ANNs cannot transfer their knowledge. Because the knowledge is distributed through the connections of its processing units it is impossible for a human to comprehend how an ANN provides its solutions. ANNs operate on numerical data and a translation must be done for it to use symbolic information, something that other forms of rational computing (and humans) do very well. ANNs cannot utilize legacy knowledge engineering acquired for other systems (i.e. expert systems). There are also open academic questions about how to do the proper training of an

ANN. What is representative set of data? How will one know when the ANN has been trained enough? How does one know it has not found a local minima? How do I know the ANN will generalize its knowledge correctly?

The problems and strengths of ANNs stem from the complexity of an ANNs structure which makes it non-deterministic but fast (much like the biological model on which it is based). Many of the shortcomings are being addressed which allow the ANN to become an integral part of Command Decision Modeling. Stand-alone neural nets are already proving themselves capable of command and control behavior for limited engagements. Combined with other methods of AI within a proper architecture they will prove very useful for rapid response, adaptivity, and knowledge acquisition in the systems of the future.

REFERENCES.

- [i] DARPA Neural Network Study. *AFCEA International Press*. October 1987- February 1988.
- [ii] "The Artificial Neural Network Users Group Posting: Part 1", <http://198.232.145.1/>. March 1996.
- [iii] **McCulloch**, W.S., and **Pitts**, W. "A Logical Calculus of Ideas Immanent in Nervous Activity". *Bulletin of Mathematical Biophysics*. Vol 5, 1943.
- [iv] **Jain**, Anil K.; **Mao**, Jianchang; and **Mohiuddin**, K.M. "Artificial Neural Networks: A Tutorial". *Computer*, March 1996.
- [v] **Murray**, Alan F., editor. *Applications of Neural Networks*. Kluwer Academic Publishers. The Netherlands, 1995.
- [vi] **Knepell**, Pete. "Neural Net Technology / NTF / ALMC". Logicon RDA.
- [vii] (Submitting to STRICOM request for information.)
- [viii] **Weaver**, Wm. Bruce; and **William**, William J. "Simulating Generic Military Decision Making with an Empirically-Trained Neural Network". *Proceedings of the Fourth Conference of Computer Generated Forces and Behavioral Representation*. Institute for Simulation and Training Technical Report. Orlando, FL. May 4-6, 1994.
- [ix] **Jaszlics**, Ivan J.; **Jaszlics**, Sheila L.; and **Jones**, Stewart H. "Automated Battlefield Simulation Command and Control Using Artificial Neural Networks". *Fifth Annual Conference of AI, Simulation, and Planning in High Autonomy Systems*. IEEE Computer Society Press. Gainesville, FL, December 7-9, 1994.
- [x] **Neuman**, Gary. Submitting to STRICOM Request for Information. <http://>
- [xi] **Tan** Chew Lim; **Tong** Seng Quah; and **Hoon** Heng Teh. "An Artificial Neural Network that Models Human Decision Making". *Computer*. March 1996.
- [xii] **Hertz**, J.; **Krogh**, A; and **Palmer**, R.G. *Introduction to the Theory of Neural Computation*. Addison-Wesley, Reading, MA, 1991.
- [xiii] **Hertz**, J.; **Krogh**, A; and **Palmer**, R.G. *Introduction to the Theory of Neural Computation*. Addison-Wesley, Reading, MA, 1991.
- [xiv] **Pal**, Sankar K.; and **Srimani**, Pradip K. "Neurocomputing: Motivation, Models, and Hybridization". *Computer*. March 1996.

[Return to the Top of this Section](#)

TAA6 TECHNOLOGY AREA ASSESSMENT:

Lattice Automata

TITLE:Route Planning within Dynamic and Uncertain Environments

ASSESSOR:

Andrew Webster

Defence Research Agency
Aircraft Sector
FAX: (44) 1252-393091
Farnborough, UK
email:anwebster@dra.hmg.gb

OVERVIEW OF THE TECHNOLOGY AREA:

DESCRIPTION OF THE TECHNOLOGY:

Our approach to the design of real-time agents able to operate intelligently within a synthetic simulation environment has been a synthesis of three techniques: lattice based architectures, isomorphic representation systems, and dynamic programming algorithms.

The lattice architecture is a three dimensional construction consisting of a locally connected network of automata. The architecture itself is organised such that two dimensions represent surface space and the third represents time.

As such this architecture can support an isomorphic representation system able to describe and encode spatial-temporal knowledge within a computational formalism. In this form the spatial dynamics of a battlefield map can be described graphically.

Once the status of an on-going scenario has been defined the use of specialised dynamic programming techniques (developed at DRA Farnborough) enable goal directed problem solving to be performed through the use of relaxation techniques.

Results so far have demonstrated to UK professors and military that this combination of techniques is a powerful approach for deriving solutions to complex multi-agent route planning and route synchronisation problems.

BACKGROUND:

The above techniques were initially developed from a computational requirement to capture and manipulate the spatial-temporal knowledge which we observed were heavily relied upon by human experts when reasoning about complex route planning tasks.

However two important difficulties were encountered when attempting to apply traditional knowledge acquisition techniques for the capture of spatial-temporal knowledge. Firstly, human experts found it difficult to articulate the spatial knowledge they perceived they were applying to the problem solving process, and secondly our knowledge engineers evaluated the traditional symbolic representation schemes to be semantically weak when attempting to computationally encode spatial-temporal knowledge. In short we considered a radically new paradigm would be needed if progress in this area was to be forthcoming.

In view of this our approach has been to design a route planning decision tool in which human planners are able to reason in the form of graphical defined spatial-temporal constraints. Our design has attempted to ensure these constraints are perceived by the user to be both natural and congruent to the cognitive problem solving process itself.

The result of this work has been to demonstrate a very flexible mechanism for capturing, explicating and manipulating spatial-temporal knowledge in a goal directed manner. Significant extensions of this work now permit solutions to the co-ordination and synchronisation of multi-agent missions whose domains are both dynamic and uncertain.

The UK military customer is now funding further expansion of this work as part of development of the AH64L attack helicopter mission system.

NOTABLE IMPLEMENTATIONS AND VARIANTS:

As our research is still in its relatively early stages, the systems potential has not yet been fully delimited. Our current implementation therefore remains fairly generic, however a flavour of the type of mission problem we are currently able to address can be expressed as follows:

- Given a mission consisting of x agents in which each of these agents initially reside at different geographic locations, determine the of co-ordinated route plans that will ensure each of the agents will synchronise their arrival at pre-specified targets at a pre-specified time(s).
- These derived routes should intelligently exploit terrain topography whilst simultaneously avoiding hostile areas (be they static or dynamic), and utilise friendly areas opportunistically where possible.
- Furthermore, if new information is encountered during the mission itself, replanning will take place automatically and communication between agents will prioritise amended plans to ensure co-operative deconfliction.

Whilst the above describes only a single mission it is possible to invoke several multi-agent

missions simultaneously each with their own specific objective and criteria upon how they should perceive the battlefield environment. Once specified these missions can run in real-time and their movements animated upon a digitised map.

We see the strengths of this implementation being the ease in which a user can very rapidly model the changing circumstances of a battlefield environment, and the clarity offered by the system to interpret the system generated route plans as a rationale consequence of the information provided.

To reinforce user confidence as to the credibility of the systems planning performance we have also incorporated a manual mode that allows the human operator to evaluate their route planning skills against those of the system. Results to date suggest the system's performance is not only comparable, but will very often be superior to those of the human operator, especially when the scenarios increase in complexity.

TECHNICAL DETAILS:

COMPUTATIONAL REQUIREMENTS:

Currently the system is event driven via the HCI (Human-Computer Interface) and is written using the visualisation language PV-WAVE. However where there are heavy computational requirements remote calls are made to C language procedures, notably the core relaxation algorithms.

PERFORMANCE POTENTIAL:

At present DRA Farnborough's prototype system is designed with an infra-structure that permits any number of agents per mission and any number of missions per scenario. Current pre-mission plan generation for a co-ordinated 10 agent mission will be of the order of 3.0 seconds when operating upon a lattice of granularity 100x100x200 and multi-threading over 4 processors. Up scaling either lattice granularity, number of agents or number of missions is found to be a linear function of performance. The time taken for a single agent to re-plan during a mission (due to newly acquired information) is of the order of 0.25 seconds.

HARDWARE IMPLICATIONS:

The system currently runs on a Sun Sparc workstation operating under UNIX utilising just a single processor at any one time. However recent work includes the ability to multi-thread this code across several processors as a progression towards parallelisation. We see this as becoming a more important issue as we begin to scale-up to richer and more complex tactical domains.

NETWORK IMPLICATIONS:

Work concerning the ability to interface across a DIS architecture remains at the specification stage.

APPLICABILITY TO COMMAND DECISION MODELLING.***POTENTIAL AS A PRIMARY OR SOLE TECHNICAL APPROACH:***

We consider intelligent route-planning to be a fundamental element of both strategic and tactical command decision modelling. However it is our view that the reasoning processes necessary for route planning and route synchronisation are very different from those applied when actioning a specific engagement rationale.

As such we perceive that future command agent architectures will be required to reason both spatially/temporally AND propositionally, this mixed reasoning initiative being depending upon the specific nature of the task.

SUBPROBLEMS ESPECIALLY SUITED TO THIS TECHNOLOGY:

In its present form DRA Farnborough's lattice based spatial-temporal reasoning approach can be implemented as a reactive route planner whose tasking objectives would be determined at higher echelons of command.

On-going work at DRA Farnborough is currently addressing the modelling issues concerned with representing the four dimensional battlefield environment and developing novel techniques in which tactical and strategic doctrine can be defined graphically.

CONCLUSION.

Results to date have demonstrated the lattice approach to be an extremely flexible architecture in which the richness of a complex, uncertain, and dynamic environment (typical of the synthetic battlefield) can be logically manipulated to derive credible real-time route-planning solutions.

[Return to the Top of this Section](#)

TAA7 TECHNOLOGY AREA ASSESSMENT:***AI Planning Systems***

ASSESSOR:

Christopher Elsaesser, Ph.D.

Principal Engineer

Artificial Intelligence Center.....Phone: 703-883-6563

Mail Stop W640.....FAX: 703-883-1279

The MITRE Corporation.....email:chris@ai.mitre.org
 1820 Dolley Madison Blvd.
 McLean, Virginia 22102-3481

OVERVIEW OF THE TECHNOLOGY AREA:

DESCRIPTION OF THE TECHNOLOGY:

AI Planning is a computational process of assembling a sequence of actions to accomplish a goal. This section concentrates on "generative planning," which is most relevant to command decision making, especially course of action generation. We do not discuss other planning problems such as route planning and "reactive planning" for resolution unit behaviors.

Generative planning systems strive to be application independent. Actions commonly are represented by templates called "operators." A knowledge engineer writes operators to model the activities that can be executed in the target environment (e.g., move, shoot, communicate). In hierarchical planners, operators also are used to represent abstract activities, such as "Attack" and "Defend." Case-based planners use a library of plans, that

are essentially examples of a network of instantiated abstract and concrete operators. Planning can be viewed as search through the space of possible plans. The search is for a sequence of instantiated operators that, if executed, are anticipated to bring about the goal state.

BACKGROUND:

Planning evolved from early work on means-ends analysis problem solving. Planning systems are distinguished from the general area of problem solving because they mainly deal with explicit representation of action and change. The purpose of a plan is to change the given state of the world to a desired goal state, rather than, say, diagnosing a disease.

Early planners were viewed as the "brains" of some agent that effected the changes in the world. The prototypical agent carrying out a plan is a robot. But it is not necessary that a planner be situated in the environment. In fact, a camp of researchers concerned with real time control of agents (usually robots) situated in the environment led an assault on the practicality of planning to fulfill its originally envisioned role. It is now accepted that planning is an unrealistic paradigm for controlling agents situated in an uncertain, rapidly changing environment. Planning is, however, ideally suited to model the

supervisors of such agents.

In general planning is computationally intractable. Practical planners, even when they are ostensibly domain independent, impose restrictions on the problems that can be represented in their action formalism. Such restrictions allow one to exploit search heuristics tuned to that formalism, but can make certain variants of planners less practical for a given domain.

NOTABLE IMPLEMENTATIONS AND VARIANTS:

There are three major variants of planning systems that are most suitable for modeling command decision making. They differ mainly in how they crack the nut of computational intractability.

TASK DECOMPOSITION (HIERARCHICAL).

ABSTRIPS is considered the first planner that used this approach. SIPE, O-Plan, and Adversarial Planner are typical of modern planners using this approach. Users represent problems as a hierarchy of abstraction spaces through the "operators" they write. Operators decompose abstract goals into more concrete subgoals, usually including ordering constraints to help the planning algorithm in its search. In effect, task decomposition breaks the search space into a series of layers. At each layer there are generally only a few ways to accomplish a given subgoal, each represented by an operator. Once an operator is selected, subgoals in its "plot" must be instantiated--again,

usually only in a few ways.

NON-LINEAR.

NOAH was the first non-linear planner. Nonlin, TWEAK, and

SNLP are modern examples. Conceptually, the planner uses means-ends analysis, trying to connect the goal situation to the initial situation by a series of partially instantiated operators. The process is "non-linear" because the planner tries to avoid backtracking by delaying commitment to action ordering as long as possible. Thus, the search is through a space of partial plans. These planners are theoretically cleaner and simpler to implement than task decomposition planners. But their basic representation is sometimes too weak (from whence the name "TWEAK") for many domains.

CASE-BASED.

Case-based planners try to avoid generating plans. Instead, they try to find a previous plan in the case-base that accomplishes a similar goal under similar circumstances. If such a plan can be found, the planner attempts to modify it to meet the given situation, resources, etc. This approach can be more efficient than generating a plan from scratch when many similar problems are encountered, and to some is a better cognitive model of planning. One difficulty of case-based planning is in case retrieval, which largely depends on how the case base is indexed. When a "similar" plan is retrieved, the planner has to transform it to match the problem. This is a form of analogical reasoning, which is notoriously difficult. Case-based planners also have to include a generative planner to be called when no similar case can be found. Successful instantiations of plans are stored in the case base, based on some similarity index, for later use.

SPECIAL REQUIREMENTS FOR COMMAND DECISION MAKING:

Military planning includes several aspects that are not addressed by all planning systems. The most important are: Multi-agent coordination, adversarial reasoning, reasoning under uncertainty, plan execution monitoring, and replanning.

TECHNICAL DETAILS:***COMPUTATIONAL REQUIREMENTS:***

Planning is computationally intractable, and each approach could run into computational difficulties. Careful matching of the planning technique to the application can usually address the problem. In practice, the computational bottleneck is due to repeated calls to domain-specific functions such as route planners and attrition functions that are made in the process of evaluating which step to add to the plan.

PERFORMANCE POTENTIAL:

Generative planning can be effective when it is specialized to perform cognitive activities that are normally outside the sense-react cycle of the simulated environment. Sequencing or situated reasoning techniques should be used instead of planning when automating problem solving behavior at the resolution unit level.

HARDWARE IMPLICATIONS:

Planning systems are comfortably implemented on standard workstation technology. Note that it is the problem of planning that is computationally intractable. The planning technique must address the intractability through use of heuristics. Use of specialized hardware such as parallel processors is a red herring.

REMARKS:

The hardest part about planning is knowledge engineering. This entails choosing how to represent goals, facts about the environment, operator preconditions, and so on. Many planning systems use something like propositional fluents:

$$\text{Location}(\text{agent2}, \text{situationN}) = (\text{lat}, \text{long})$$

or some variation. Choosing which are the base level relations (e.g., Location), which are the higher level goals, and so on is very difficult.

The most software development work incorporating a planner in a battlefield simulation comes in accessing simulation functions to get facts or predict effects of actions. Examples are environmental utilities like route finding, and attrition calculations. Once the application is designed, one sometimes finds the computational overhead of calling such functions hundreds of times accounts for the majority of planning time.

APPLICABILITY TO COMMAND DECISION MODELING.***POTENTIAL AS A PRIMARY OR SOLE TECHNICAL APPROACH:***

Planning is a primary role of a command agent. Using a planning system is ideally suited to automated plan cascading and simulating the behavior of the opposing forces. That is, automated planning essentially implements the "science of control." There are several

demonstrations of this point, including the Eagle-AP combination described later in this report, and the Command Entity work done by SAIC, USC-ISI, and Hughes for ARPA.

Planning systems are not cognitive models of command decision making, even when the primary product of command decision making is a course of action. Even planners that are inspired by observations of problem

solving behavior, such as case-based planners, should not be view as cognitive models. Human heuristics are simply an inspiration for the algorithms.

Besides planning, the other main activity of a command agent is situation assessment. There are strong arguments that basing situation assessment on the anticipations encoded in a plan, and the command agent's estimate of the enemy plan, is a viable approach.

REFERENCES.

A very good overview of AI planning can be gained from:

James Allen, Hendler, J., and Tate, A. (eds.) Readings in Planning (1990). Morgan Kaufmann Publishers, Inc. San Mateo, California.

Other essential books are:

James Allen, Kautz, H., Pelavin, R., and Tenenberg, J. Reasoning About Plans (1990). Morgan Kaufmann Publishers, Inc. San Mateo, California.

Thomas Dean, Wellman, M. Planning and Control (1991) Morgan Kaufmann Publishers, Inc. San Mateo, California.

David E. Wilkins. Practical Planning: Extending the Classical AI Planning Paradigm (1988). Morgan Kaufmann Publishers, Inc. San Mateo, California.

[Return to the Top of this Section](#)

TAA8 ***TECHNOLOGY AREA ASSESSMENT:***

PETRI NETS

AND

COLORED PETRI NETS

ASSESSORS:

Dirk Ourston, Ph.D. and Noemi Paz, Ph.D.

Science Applications International Corporation
 3045 Technology Parkway
 Orlando, Florida 32826-3299
 (407) 282-6700
 Dirk_Ourston@cpqm.saic.com

OVERVIEW OF THE TECHNOLOGY AREA:

BACKGROUND:

Petri nets were introduced in the early 1960s by C. A. Petri [Petri, 1962]. Petri nets have been used as a mathematical tool for modeling distributed systems requiring specification of concurrency, non-determinism, communication and synchronization. They are an extension of finite state automata theory that allows the expression of simultaneously occurring events. Petri nets have been used extensively for modeling network protocols. Example applications include:

- A model for a Naval command and control system [Berger, 1993].
- An intelligent network simulation [Capellmann and Dibold, 1993].
- A model for work flow in a nuclear waste plant [Mortensen and Pinci, 1994].
- A model for a document storage system [Scheshonk, 1994].

Petri nets were developed to overcome a deficiency of finite state machines (FSM), that only a single event could be modeled at any given time using a finite state machine (i.e. the machine could only be in a single state). This makes it difficult for finite state machines to model coordinated actions, for example where the actions of tanks and dismounted infantry must be coordinated when taking an objective.

Much of the research in Petri nets has been conducted in Europe, most notably at the University of Hamburg (home of Carl Petri), and University of Aarhus (home of Kurt Jensen, one of the leading researchers in colored Petri nets). In the United States, George Mason University has been active in applying Petri Net modeling techniques to C3I applications. Key researchers include Alexander Levis and Abbas Kazim Zaidi. In addition, some work on Petri net applications has been done at the University of Central Florida [Guha, Bassiouni, and Schow, 1996].

Most Petri net models for command and control applications have been at a high level. They have primarily been used to describe information flow through the system, and the states that the system may be in. In its pure form, Petri net descriptions would simply say that, given that tokens exist at all of the input nodes for a transition, the transition will occur. This begs the issue of what reasoning is required to determine whether or not tokens should exist at input nodes. In short, Petri nets seem to be much better at describing actions, then at describing the reasons behind the actions.

Interest in Petri net research continues to be high within the academic community. Within industry, there have been notable applications of Petri nets to network problems and manufacturing control. However, there have only been a few applications of Petri net models to command decision making up to the current time. Most of the military applications of Petri nets have focused on information flow through the system being modeled - not on how the decisions were made.

DESCRIPTION OF THE TECHNOLOGY:

The simplest Petri net is known as a *black and white net*. The structure of this basic net is a digraph with nodes which may be *places*, drawn as circles, or *transitions*, drawn as rectangles or lines. Directed arcs which connect places to transitions are called *input arcs*, and the corresponding places are called *input places*. Similarly, arcs which connect transitions to places are called *output arcs*, and the corresponding places are called *output places*.

Petri nets are *marked* by placing tokens (drawn as dots) inside places. Tokens can move from place to place through the transitions. Transitions model activities that occur when tokens move through them.

The distribution of tokens in the places of a Petri net is called *state* or *marking*. The current state of a net is given by the number and type of tokens in each place. When all input places of the transition contain at least one token, a transition is eligible to be fired or enabled. If it does fire, one token is removed from all of its input places and one token is added to all of its output places.

Using places, transitions, and directed arcs one can model various concepts. For example, parallel or concurrent activities and conditional if-then decision or conflict are two easily modeled concepts. Two transitions are concurrent if one transition may fire before or after or in parallel with the other. Transitions that cannot occur simultaneously are called in conflict. One way to model concurrent activities is to draw a net containing a transition, say t_1 , with two output arcs. Then, let each output arc be followed by an output place and each of those outplaces be followed by a transition, say t_2 and t_3 . Transition t_2 and t_3 in this portion of a net model concurrent activities. Conflict, also called choice or decision, can be modeled by a Petri net structure where a place has two output arcs, each followed by a transition.

When a Petri net graph is complete the structural or syntactic properties of a system under study are specified. The Petri net can then be used to support the analysis of behavioral or dynamic properties of the concurrent system. Liveness, boundedness, cyclicity, and reachability of a marking are some of these properties. The following are informal definitions of these properties. A marking is reachable if it can be obtained from the initial marking by successive occurrences of transitions. Then, a Petri net is live if for every reachable marking and every transition t it is possible to reach a marking that enables t . A simpler way to express this is: liveness holds if every transition can always occur again. Deadlock-free is a property of liveness. A net is deadlock-free if every reachable marking enables a transition.

Another property is boundedness. This property is used to model capacity of a component or the number of resources. A place is k -bounded when it contains a maximum of k number of tokens. A special case of boundedness is safeness. When a place is 1-bounded it is said to be *safe*. Safe represents modeling of a flip-flop, on-off, or 1-0 feature of a system. If a Petri net is unbounded then overflows can occur.

A marking is a home marking if it is reachable from every reachable marking. If a Petri net's initial marking is a home marking then the net is cyclic. For example, cyclic nets model a system that returns to its initial state after a user interacts with it.

Analysis of a Petri net involves studying the behavioral properties of the net. The most straight forward technique for analysis is to do a complete enumeration of all reachable marking. This technique is limited to small nets due to the complexity of the state-space explosion. Other techniques have been developed, but each apply to special subclasses or extensions of Petri nets or to special situations. Extensions to Petri nets generally reduce the size, but are more complex to understand.

Colored Petri nets [Jensen, 1995] are an extension of black and white Petri nets that reduce the complexity of Petri nets for models where there are a well-defined set of actions that apply to a variety of objects. In a black and white net, a separate graph would have to be developed for each object, whereas with a colored net, each object could be associated with a different set of colored markers, thus reducing the complexity of the graph. Firing of transitions for colored Petri nets is dependent on not only having a token in every input place, but on those tokens being the specified color. The reduction in complexity provided by colored nets still preserves the system's properties to be analyzed.

In addition to colored Petri nets, the Petri net concept has also been extended to include probabilistic reasoning [Marsan, 1989] and temporal logic [Hillion, 1989]. Probabilistic reasoning is modeled by adding delays to the net that are randomly distributed. These nets are called stochastic nets. Nets that model deterministically distributed delays are called timed Petri nets. The delay is modeled by adding a time tag to either places or transitions. It is not important which alternative is chosen since they are equivalent. Colored Petri nets are extended for time by adding a time stamp to each token along with the color. Now for a transition to be enabled, a token of the right color must exist and that token must be ready (i.e., it is past the earliest time at which the token can be used.)

An important aspect of a system under study is the performance of the system. That is, the system designer is concerned with the maximal time used for the execution of certain activities and the average waiting time for certain requests.

Recently object-oriented technology has been applied to Petri nets. [Lakos, 1995] has proposed an extension to Petri nets that will model all of the object-oriented concepts, such as, inheritance, polymorphism, and dynamic binding. This net will allow direct modeling of complex systems with multiple levels of activity. It has been shown that colored Petri nets are behaviorally equivalent to the object-oriented Petri Nets [Lakos, 1995] and therefore, colored Petri net analysis techniques can be adapted for object-oriented nets.

NOTABLE IMPLEMENTATIONS AND VARIANTS:

This section surveys the status of Petri net tools and implementations, and closes with a discussion of the place for Petri nets in command agent design.

Tools

Petri nets are used for a wide variety of applications. This is due to the generality and permissiveness of the Petri net formalism. Essentially they can be applied to any area that can be modeled with a digraph and needs to express parallel or concurrent activities. However, Petri net models rapidly become too large for analysis. Software tools that assist with drawing, analysis, and simulation are necessary for practical use of Petri nets.

Many tools have been developed for Petri net applications. The vast majority of tools are the result of University research and are not commercially supported. The tools discussed below represent products with some level of maintenance and support. (See [Feldbrugge, 1993] for an overview of the characteristics of other Petri net tools.)

Design/CPN. This tool supports the development of Hierarchical Colored Petri nets. It also has the capability to model time constraints. The tool consists of three parts:

- the CPN editor that can be used to support construction, modification, and syntax check of CPN models,
- the CPN simulator that supports interactive and automatic simulation of CPN models,
- the Occurrence Graph Tool that supports reachability analysis for CPN models.

The tool runs on either UNIX workstations or Macintoshes with Mac OS. It is available through the University of Aarhus, <http://www.daimi.aau.dk/designCPN>.

ELSIR. This tool supports typed and valued tokens, conditions, actions, timed and stochastic transitions. A graphical editor is provided. The tool runs on UNIX work stations and uses X-Motif as the graphical interface. It is available from ADV Technologies (Richard.Dykiel@sp1.y-net.fr).

INCOME. INCOME is designed to analyze business processes. It provides an interface to the Oracle database. The INCOME tool set consists of a Dictionary for use in storing project information, the process modeller for capturing process models, and the simulator for process model analysis. INCOME runs on 486 and higher PCs and UNIX workstations. It is available from PROMATIS Consulting GmbH, hq@PROMATIS.de.

MACROTEC. This tool supports colored, stochastic, timed Petri nets. A graphical user interface is provided. The focus for Macrotec is business modeling. Macrotec runs on UNIX workstations. Macrotec is available from CRIM (keller@crim.ca).

SYSTEMSPECS. This tool supports hierarchical high-level Petri net models. A graphical interface is provided. It runs on PCs, UNIX workstations, and Macintoshes. Systemspecs is available from Ivy Team (+41 42 23 20 20).

TemPRO. TemPRO supports colored, timed and prioritized Petri nets. An integrated graphical editor is provided. TemPRO runs on PCs. TemPRO is available from Software Consultants International (scil@interserv.com).

2.2 Applications

The following examples are applications of Petri modeling that relate to the command entity model.

Copernicus. The Copernicus system is being developed for the Navy to provide up to the minute command and control information to the Naval command officers. In this case, Colored Petri nets have been developed to analyze methods for obtaining relevant information within Copernicus. (For further details, see [Perdu, Spohnholtz, and Levis, 1994].)

Intelligent Command and Control. This work applies the Petri net model to the problem of modeling role based agents within an Army command and control structure. (For further details see [Bowden, Davies, and Dunn, 1995].)

C2PANTHRE. This system is a low resolution stochastic model of command and control processes. The emphasis is on queuing delays involved with the intelligence, fire support, logistics, air defense and maneuver Battlefield Functional Areas. Output consists of statistics indicating the time required for information to flow through the system.

Cockpit Assistant System. (CASSY) This work modeled commercial aviation pilots behavior regarding flight plans. It is a real-time system with a knowledge base of normative pilot behavior as documented pilot handbooks and air traffic regulations. Petri nets are used to encode the knowledge. The Petri nets are used to validate the knowledge base and to compare actual pilot actions against run time. If the system believes the pilot made an error, warnings are reported to the pilot. This system runs with a real-time net interpreter. (For further details see [Ruckdeschel and Onken, 1994].)

SUPPORT FOR COMMAND ENTITY MODELING:

Petri nets could be applied to the command entity model as a way of specifying the sequence of steps that would be followed to achieve a particular output action. The intelligence for the system would reside in the agent technologies responsible for determining when a specified decision should be made. Once the decision was complete, the Petri net could be used to move the system into the next state based on the input decision.

TECHNICAL DETAILS:

BANDWIDTH REQUIREMENTS:

The effect of integrating a Petri net with existing knowledge based reasoning techniques would have little impact on the overall bandwidth for the command entity reasoner. This is because the Petri nets would be used at the "meta" level of the reasoning process, responsible for managing the information flow through the command entity reasoner, but not involved with the details of the decision making process.

THROUGHPUT / PERFORMANCE POTENTIAL:

Petri nets would have a neutral effect on the system throughput and performance potential.

HARDWARE IMPLICATIONS:

Petri nets have been developed and applied on a variety of platforms, ranging from PCs and Macintoshes to high level UNIX workstations.

NETWORK IMPLICATIONS:

Petri nets can be subdivided hierarchically, permitting the distribution of the Petri net over local or wide area network.

REMARKS:

The use of Petri nets in command entity modeling has only been lightly addressed by current research. If they are of value at all, their main application would be in managing the information flow through the command entity reasoning process. They could be useful in addressing the coordination aspects of the command entity processing, where either local or distributed agents must coordinate their information flow prior to an operational decision being made.

APPLICABILITY TO COMMAND DECISION MODELING:*POTENTIAL AS A PRIMARY OR SOLE TECHNICAL APPROACH:*

This technology is not recommended as the single foundational technology for creating command entity models. It can serve as an integration mechanism for bringing together the technologies that are appropriate to particular decision making tasks. A strength of the Petri net approach are that it has a well-defined theoretical foundation, allowing a Petri net model to be verified using rigorous mathematical techniques. In addition, Petri nets are useful for graphically showing the information flow through the system, particularly when concurrent events are involved. The main weakness of Petri nets with regard to command entity applications is that they have little expressive power for capturing knowledge-intensive domains. Again, their target domain is information flow, not reasoning.

SUBPROBLEMS ESPECIALLY SUITED TO THIS TECHNOLOGY:

Petri nets are a valid technique for specifying real-time system requirements [Davis, 1990]. For the Close Combat Tactical Trainer (CCTT) System, military doctrinal literature was encoded into Command Instruction Sets (CIS). These CISs formed a basis for SAF behaviors [Bimson, Marsden, McKenzie, and Paz, 1994]. During the Requirements phase the CISs were translated into FSMs and then translated again into coded FSMs during the Implementation phase. Since FSMs are a subclass of Petri nets, the CISs are essentially modeled with Petri nets. However, FSMs have limitations that Petri nets do not have. For example, Petri nets naturally model concurrency while the FSM does not. Once the CISs were encoded in Petri nets, command decision making could use this knowledge base of analyzable cases when deciding on the next course of action.

Decision maker specifies a plan of action and then asks the question: How long will it take to execute this plan? A temporally augmented Petri net could be used to model timing requirements associated with coordinated troop activities. With the ability to predict a plan's execution time, a commander can compare alternative plans and include time constraints in the decision process.

Other questions a commander may ask about a specified plan of action are:

- Is this plan valid? (e.g., is doctrine followed? is chain of command maintained?)
- Are there enough resources? (e.g., tanks, ammunition, troops)
- Is the plan achievable? (e.g., troop coordination, time constraints, required permission)
- Which events must or must not occur to guarantee mission success?

Of course, to answer these questions by analysis of the behavioral properties of a Petri net, the net would have to model the concepts of interest.

WEAKNESSES AND CONCERNS:

As mentioned above, Petri nets should not be used as the method for accomplishing the command entity decision making per se. They are most useful for serving as an integrating tool in modeling and managing the information flow process, and for addressing timing issues associated with coordinated movements.

REFERENCES.

- Berger, J. and Lamontagne, L. "A Colored Petri Net Model for a Naval Command and Control System". In: M. Ajmone-Marsan (ed.): *Application and Theory of Petri Nets 1993. Proceedings of the 14th International Petri Net Conference*, Chicago, 1993, Lecture Notes in Computer Science Vol. 691, Springer-Verlag, 1993, pp.532-541.
- Bimson, Kent; Marsden, Craig; McKenzie, Rick; and Paz, Noemi "Knowledge-Based Tactical Decision Making in the CCTT SAF Prototype" *Computer Generated Forces and Behavioral Representation Conference Proceedings* May 1994, pp. 293-306.
- Bowden, F. D. J.; Davies, M. and Dunn, J. M. "Representing Role Based Agents Using Colored Petri Nets" *Fifth Conference on Computer Generated Forces and Behavioral Representation* Orlando, Florida, May 1995.
- Capellmann, C. and H. Dibold, H. "Petri Net Based Specifications of Services in an Intelligent Network. Experiences Gained from a Test Case Application" In: M. Ajmone-Marsan (ed.): *Application and Theory of Petri Nets 1993. Proceeding of the 14th International Petri Net Conference, Chicago 1993, Lecture Notes in Computer Science* Vol. 691, Springer-Verlag, 1993, pp. 542-551.
- Davis, Alan M. *Software Requirements Analysis & Specification* Prentice Hall, Englewood Cliffs, New Jersey, 1990.
- Feldbrugge, Frits "Petri Net Tool Overview 1992" In: G. Rozenberg (ed.) *Advances in Petri Nets 1993, Lecture Notes in Computer Science* 674, Springer-Verlag, 1993, pp.169-209.
- Guha, R. K.; Bassiouni, M. A. and Schow, G. "A Framework for Modeling High Level Architecture (HLA) Using Petri Nets" *Summary Report from the 14th Workshop on Standards for the Interoperability of Distributed Simulations*, Vol. 1, March 1996, pp. 515-519.
- Hillion, H. P. "Timed Petri Nets and Application to Multi-Stage Production Systems" *Advances in Petri Nets 1989, Lecture Notes in Computer Science* 424, Springer-Verlag, 1989.
- Jensen, Kurt *Colored Petri Nets Basic Concepts, Analysis Methods, and Practical Use* Vol. I & II, Springer-Verlag, New York, 1995.
- Lakos, Charles "From Colored Petri Nets to Object Petri Nets" In R. Valett (ed.): *Application and Theory of Petri Nets 1994. Proceedings of the 15th International Petri net Conference, Zaragoza 1994, Lecture Notes in Computer Science* 935, Springer-Verlag pp. 278-297.
- Marsan, M. A. "Stochastic Petri Nets, An Elementary Introduction" *Advances in Petri Nets 1989, Lecture Notes in Computer Science* 424, Springer-Verlag, 1989.
- Mortensen, K. H. and Pinci, V. "Modeling the Work flow of a Nuclear Waste Management Program" *Proceedings of the 15th International Petri net conference, Zaragoza 1994, Lecture Notes in Computer Science*, Springer-Verlag, 1994.
- Perdu, D. M.; Spohnholtz, C. L. and Levis, A. H. "Modeling Information Flow in the Copernicus Architecture Using Colored Petri Nets" *Proceedings of the 1994 Symposium on Command and Control Research and Decision Aids* Naval Postgraduate School, Monterey, California, June 21-23, 1994, pp.

115-129.

Petri, C. A. *Kommunikation mit Automaten* Bonn: Institut für Instrumentelle Mathematik, Schriften des IIM Nr. 3, 1962. Also, English Translation, *Communication with Automata* New York, Griffiss Air Force Base, Technical Report RADCTR-65-377, Vol. 1, Suppl. 1, 1966.

Scheshonk, G. "Simulation and Analysis of a Document Storage System" In R. Valett (ed.): *Application and Theory of Petri Nets 1994. Proceedings of the 15th International Petri net Conference, Zaragoza 1994, Lecture Notes in Computer Science*, Springer-Verlag, 1994.

Ruckdeschel, W. and Onken, R. "Modelling of Pilot Behaviour Using Petri Nets" In: R. Valette (ed.) *Application and Theory of Petri Nets 1994, Lecture Notes in Computer Science 815*, Springer-Verlag, 1994, pp.169-209.

[Return to the Top of this Section](#)

[Return to the Section 3 Index](#)

[Return to the Table of Contents](#)

[Return to the SCC Command Decision Modeling Page](#)

[Return to the WARSIM 2000 Page](#)

[Return to the NSC Home Page](#)

COMMAND DECISION MODELING TECHNOLOGY ASSESSMENT SYSTEM ASSESSMENTS:

SA

1. EAGLE ADVERSARIAL PLANNER
2. GEKNOFLEXE
3. SOAR

SA1: *SYSTEM ASSESSMENT:*

Eagle-Adversarial Planner

ASSESSOR:

Christopher Elsaesser, Ph.D.

Principal Engineer
Artificial Intelligence Center
Mail Stop W640
The MITRE Corporation
1820 Dolley Madison Blvd.
McLean, Virginia 22102-3481
Phone: 703-883-6563
FAX: 703-883-1279
email:chris@ai.mitre.org

OVERVIEW OF THE SYSTEM:

DESCRIPTION OF THE SYSTEM:

Eagle-AP is a combination of Eagle, a state-of-the-art ground combat analysis simulation, and Adversarial Planner (AP), an AI system designed to address key aspects of battlefield planning, including multi-unit coordination, adversarial planning (wargaming a course of action), and replanning.

BACKGROUND:

Eagle represents a significant advance in the state of the art in ground combat simulation. Key features not found in earlier simulations include intelligent reactions by resolution units and an explicit model of the command and control hierarchy.

Intelligent reaction means that each resolution unit (corresponding to the icons on the situation map display) has a local rule base that senses its environment and reacts in an intelligent manner. For example, an Eagle resolution unit can be sent an order in a standard Battle Management Language to maneuver to a named objective. The unit will determine its own route and proceed to maneuver, sending situation reports as it progresses. If the unit encounters an enemy unit, it will react appropriately, perhaps commencing an attack, or possibly retreating if the enemy has an overwhelming advantage in combat effectiveness.

There are two obvious benefits to intelligent reaction of resolution units. First is the partial elimination of "scripting" unit actions to fulfill scenario requirements. Units know how to carry out operations specified at a higher level of abstraction than exists in many combat simulations. Second, unit behaviors are programmable in an explicit, high level language more easily maintained, traced, and understood by an analyst or subject matter expert than if the activities were controlled by low level code.

Eagle's command and control representation is enabled by the intelligent reaction of the resolution units. Since units respond to military orders, higher echelon command can be simulated by specification of operations orders. This level of abstraction models real command and control, with benefits both in scenario development and in validation and verification.

Reaction will only get you so far, and higher echelon replanning is inevitable in real combat, and so it should be in realistic combat simulation. The motivation for adding AP to Eagle was to add replanning at the echelons above the reactive resolution units. Because realistic replanning tries to maintain the momentum of the original plan, it was necessary to use AP's planning capability to "cascade" operations orders from the analyst's specification to the orders to resolution units. Plan cascading has the potential of reducing planning during scenario development by an order of magnitude.

TECHNICAL DETAILS:***COMPUTATIONAL REQUIREMENTS:***

A typical configuration for Eagle-AP is an HP 700 series or a SPARC Station 10 with 64MB of RAM and 200 MB of swap space.

PERFORMANCE POTENTIAL:

Both Eagle and AP are written in Common Lisp. AP is portable to any computer for which a Lisp compiler can be purchased, including PCs and Macintosh. Eagle depends on Intellicorp's Knowledge Engineering Environment (KEE). This dependency could be removed by reimplementing the objects and methods now in KEE in the Common Lisp Object Systems. This is about a 3 staff-month job that would considerably improve Eagle's performance.

Eagle is still under development. There are many performance improvements that could be made to the code to get it to equal or surpass the performance of any other model with similar capabilities now available.

AP's code is highly optimized. Its performance depends mainly on how well the planning problem is represented, and how clever the knowledge engineer is at deferring calls to low level Eagle functions

HARDWARE IMPLICATIONS:

Eagle-AP demonstrates that very complex command decision making and execution can be simulated on standard workstation technology. It proves that exotic hardware is not necessary, even though the processes simulated are, in general, computationally intractable.

APPLICABILITY TO COMMAND DECISION MODELING.*POTENTIAL AS STAND ALONE COMMAND AGENT/MODEL::*

Eagle-AP embodies the main capabilities required of a "command agent" for ground combat simulation. Command and control may be quite different in air or naval operations.

The AP implementation in Eagle is similar to the company-level Command Entity developed by SAIC for DARPA's Command Forces program. The primary difference is that AP is capable of adversarial reasoning, of which no other multi-agent planning system is capable.

SUBPROBLEMS ESPECIALLY SUITED TO THIS SYSTEM:

Ground combat simulation; automated opposing forces.

WEAKNESSES AND CONCERNS:

In order for AP to plan battle tactics, the tactics had to be represented with Operators in the AP plan representation (see the Technology Area Assessment on *AI Planning* in this document). It is very difficult to write a robust set of operators, just as it is difficult to write an expert system.

CONCLUSION.

Eagle-AP disproves the misconception in the simulation community that speed of execution can be gained by the choice of an implementation language or faster hardware. Eagle-AP demonstrates that intractable problems can be addressed by heuristic algorithms.

Eagle-AP shows that the real bottleneck in combat simulation is not in programming behaviors or algorithms. Rather, it is in choosing and implementing an adequate representation of the combat domain. This is a problem that is inherent to combat modeling; the particular implementation either makes it explicit, as does Eagle-AP, or obscures it, like many other models.

REFERENCES.

1. Elsaesser, Christopher and T. Richard MacMillan, iRepresentation and Algorithms for Multiagent Adversarial Planning, MTR-91W000207, The MITRE Corporation, December 1991.

[Return to the Top of this Section](#)

SA2: ***SYSTEM ASSESSMENT:***

GEKNOFLEXE

ASSESSOR:

Helen Lankester

Simulation, Training and AI Research Group
 Phone: +44 (0)1959 514724
 Defence Evaluation & Research Agency
 Fax: +44 (0)1959 516084
 Fort Halstead
 email: hlankester@dra.hmg.gb
 Sevenoaks, Kent TN14 7BP
 United Kingdom

OVERVIEW OF THE SYSTEM:

DESCRIPTION OF THE SYSTEM:

The GeKnoFlexE system explicitly represents the linkage between command and control decision-making and battlefield operations so that C2 and C3I effectiveness in a range of circumstances can be studied. A combination of object-oriented, multi-agent and knowledge-based systems (KBS) techniques are used to simulate a fully automated two-sided engagement.

BACKGROUND:

Development of the GeKnoFlexE system started in September 1991. Since then GeKnoFlexE has been used for a number of studies into the effectiveness of C2 and C3I in a range of circumstances.

The GeKnoFlexE system has two main components: a generic model and a study model. The generic model encompasses the scenario independent aspects of the system which include the battlefield simulation, metrics collection tool, Graphical User Interface, C2 shell and generic Command Agent knowledge bases. The study model is derived from a particular scenario and is made up of Orders of Battle (ORBATS), Orders of March (OOM), initial locations, terrain and scenario specific Command Agent knowledge bases. The level of aggregation can be chosen to suit the scenario being modelled but this is usually company level for most units with a few specialised forces represented at a lower level of aggregation. The complete GeKnoFlexE system is covered in more detail in Lankester and Robinson (1994).

Each Command Agent represents a command post (a commander and his staff) and responds to its perceived battlefield situation, exchanges command and control messages with other Agents and entities, and thus controls battlefield operations. Each Command Agent is represented using a rule-based system, which encapsulates the tactics the Command Agent needs to use to perform its role in the battle, and an object structure in which it stores its perception of the battlefield. The rules are grouped into tasks which cover a specific function or procedure. Each rule-base is partitioned into knowledge sources which equate to the cells within a command post.

Each type, or class, of Command Agent has a set of rules and domain knowledge structures which define its behaviour. In the latest study model there are 10 different types of Command Agent knowledge base (Blue Div, Bde, Bn and Sqn; Red Div, Bde, Bn, Arty Regt, Arty Bn, AD Regt) with 46% of the tasks common to all Command Agents. Instances of Command Agents are created which have their own perceptions of the battlefield (in the latest study model there are 34 Command Agent instances). The Command Agents maintain a perception of the battlefield based on reports they receive as the simulation progresses.

At present two full study models for completely different scenarios have been developed, validated and studied. Further development of the study models is still in progress, and use of the system and further study model development is expected to continue.

The GeKnoFlexE project is funded by the UK Ministry of Defence and is directed by the Defence Research Agency. Software has been developed in association with Logica UK Ltd and scenarios have been developed and validated in association with SCS Ltd, UK.

TECHNICAL DETAILS:***COMPUTATIONAL REQUIREMENTS:***

The GeKnoFlexE system is implemented using Common Lisp, the Common Lisp Object System (CLOS) and the Common Lisp Interface Manager (CLIM). Currently either a Sun SPARCstation 10 or 20 operating with Solaris 2, 128 MB of RAM and 300 MB swap space is used to run the tool set.

PERFORMANCE POTENTIAL:

The GeKnoFlexE system is currently able to model an engagement between a Blue division and a Red division where a total of 308 aggregate sub-units are employed. These sub-units are in the most part of company size but smaller sub-units are represented where necessary, e.g. for recce. In total the simulation currently represents 472 event predicting entities. Running 20 hours of simulation time on a Sun SPARC 20 with the above configuration takes approximately 24 hours of real time and on a Sun SPARC 10 it takes approximately 36 hours of real time.

There is no requirement for the GeKnoFlexE system to run at real time as it is used for studies where the metrics collected by the system can be analysed after the simulation run has been completed. However, there is scope for enhancing the overall performance of the system should it be required.

HARDWARE IMPLICATIONS:

The GeKnoFlexE software has been ported from the Sun SPARCstations to a Silicon Graphics Indigo 2 with minimal effort and could be ported to other hardware for which Common Lisp, CLOS and CLIM are available.

APPLICABILITY TO COMMAND DECISION MODELLING.

POTENTIAL AS STAND ALONE COMMAND AGENT/MODEL:

A number of simulation based projects within the UK Defence Evaluation and Research Agency became interested in the Command Agent component of the GeKnoFlexE system in late 1993/early 1994 as an approach to reducing the manpower requirements for running exercises. This led to a study which addressed the feasibility of collaborating to develop a generic Command Agent software function for use by each of the simulations. The study scoped the problem and analysed the requirements of the application areas which ranged from a command and staff training system to an Operational Analysis (OA) divisional level wargame. The study concluded that establishing a collaborative Command Agent development programme was both feasible and likely to provide significant savings in comparison to each project pursuing independent programmes (Heard et. al. 1994).

Two proof of principle demonstrators were produced which used the GeKnoFlexE system to investigate how they could use Command Agents to represent command and control to reduce controller workload. One of the demonstration systems linked the GeKnoFlexE software to the Corps Battle Simulation (CBS) and is described in more detail in Kendall and Page (1995). This demonstrator has successfully used the CBS generic interface to enable the GeKnoFlexE Command Agents to control the opposition forces within CBS thus reducing the number of controllers required, albeit within a constrained scenario.

The GeKnoFlexE software has also been connected to a divisional level war game used for Operational Analysis. The GeKnoFlexE system acts as a virtual player on the simulation's network so that Command Agents are able to control recce units. Recce units were automated because they are used extensively early in the simulation, when the external players controlling the game are not very familiar with its operation. This work is aimed at reducing player workload and ultimately the number of players, thus helping to solve the problem of finding sufficient military personnel to staff the game. This work led directly to a Command Agent capability which is now routinely used to control the movement of subordinate forces.

The GeKnoFlexE system has its own battlefield simulation in which the Command Agents operate. The simplest way to adapt the GeKnoFlexE system to get it's Command Agents to control units in another simulation is to make the GeKnoFlexE battlefield simulation emulate the target simulation. In this way the Command Agents can continue to work within the simulation designed for them but have their orders forwarded to the new one. This approach has been used to produce both of the demonstration systems described above. Further development of the demonstrators was not practical and running two battlefield simulations is an unnecessary overhead. As a result of this a collaborative project was set up in December 1994 with the aim of extracting the Command Agents from the GeKnoFlexE system, making them accessible to the other simulation systems as well as extending their use and enhancing the

tools provided for the user and developer (Lankester, 1995).

The collaborative project adopted a spiral development approach which started by developing a prototype system which helped to further identify the requirements of the applications and research possible solutions. In early 1995 the prototype was demonstrated linked to the UK Army's Higher Formation Trainer, ABACUS. In this ABACUS demonstration exercise an OPFOR Special Forces Battalion was controlled by a Command Agent able to conduct a defence, a withdrawal in contact and call for appropriate artillery support. A man-in-the-loop facility was also provided so that the controller could change the Command Agent's orders or take over from it completely if required.

Full development of the Command Agent software is now underway and it will be demonstrated in the UK component of the US STOW97 ACTD.

SUBPROBLEMS ESPECIALLY SUITED TO THIS SYSTEM:

The GeKnoFlexE system is able to model the linkage between battlefield decision-making and tactical operations. Changes in the C3I environment are reflected on the battlefield. A range of comparative studies could be made using the system, for example new tactics or computer systems could be represented and the impact of these on the overall battle outcome could be assessed.

The current approach taken to implementing Command Agents is particularly suited to representing tactics, doctrine and the more reactive type of C2 decision-making. Subject Matter Experts (SME) are able to validate Command Agent decisions by looking at the Command Agent's perception of the battlefield, and tracing through the knowledge sources and rules used. Many other knowledge representation techniques do not allow sufficient visibility of the decision process.

WEAKNESSES AND CONCERNS:

The process of knowledge acquisition, knowledge base development and validation is time consuming and consequently costly. The resulting Command Agents will only be able to behave appropriately in situations for which they have rules.

The rule-based approach does not represent longer term planning and data fusion adequately. The design of the multi-application Command Agent software addresses this by producing an architecture which enables different knowledge representations to be used to supplement the core multi-agent KBS approach. This hybrid approach is still at a very early stage of research.

The current Command Agents are deterministic, however human interactions with the simulation and in some cases the simulation itself are non-deterministic. This provides some apparent variability in Command Agent behaviour as it is rarely presented with exactly the same information twice. In some cases more human like variability is required, e.g. to represent different styles of command, stress and morale. A hybrid approach to Command Agent development will enable this type of variability to be represented more easily. It is worth noting that some applications will still require the Command Agent's behaviour to be deterministic, for example some operational analysis systems.

CONCLUSION.

Command Agents based on the GeKnoFlexE system have been successfully used to command and control units within three different simulations, reducing controller/player workload. These simulations are used for different applications ranging from training to operational analysis, and yet they all require a similar command decision-making model and associated facilities for users. This method of representing C2 decision making has been shown to be a suitable starting point for developing a common Command Agent software function and facilities for use by different applications.

REFERENCES.

Heard, R. J. B., Lankester H. G., and Robinson, P. K. (1994) iCommand Agents research feasibility/scoping study reportî *Technical report DRA/CIS/CSS1/TR94006/1.0*, Defence Research Agency.

Kendall, G., and Page, I. (1995) iAn Automated CBS OPFORî, *Proceedings of the Fifth Conference on Computer Generated Forces and Behavioral Representation*.

Lankester, H. G. (1995) iMulti-Application Command Agentsî, *Proceedings of the Fifth Conference on Computer Generated Forces and Behavioral Representation*.

Lankester, H. G., Robinson, P. K. (1994) iGeKnoFlexE: A Generic, Flexible Model of C3Iî, *Proceedings of the Fourth Conference on Computer Generated Forces and Behavioral Representation*.

[Return to the Top of this Section](#)

COMMAND DECISION MODELING
TECHNOLOGY ASSESSMENT

SA1: System Assessment:

SOAR

ASSESSORS:

Dr. Jonathan Gratch

Research Computer Scientist, USC-ISI
Research Assistant Professor, USC CS

Dr. Randall W. Hill, Jr.

Research Computer Scientist, USC-ISI
Research Assistant Professor, USC CS

**Information Sciences Institute and
Department of Computer Science
University of Southern California
4676 Admiralty Way
Marina del Rey, CA 90292
Phone: (310) 822-1510/11
email: gratch@isi.edu, hill@isi.edu**

OVERVIEW OF THE SYSTEM:

DESCRIPTION OF THE SYSTEM:

Soar is a unified software architecture being developed as a basis for integrated intelligent systems (Rosenbloom et al., 1991; Laird et al., 1987; Rosenbloom et al., 1993). Soar is also a developing unified theory of cognition (Rosenbloom & Laird, 1994; Newell 1992a, 1992b, 1990). This theory entails a number of constraints (for example, concerning human reaction times, knowledge representation, and learning) that bear directly on potential solutions for command decision modeling.

BACKGROUND:

Soar is a mature, state-of-the-art AI architecture that has evolved since its conception in the early 1980s. Since 1992 the Soar/IFOR research group has utilized Soar as the primary architecture for building intelligent automated agents for tactical air simulation. In recent years Soar has seen greater influence in the simulation community, being applied to the simulation of Navy, Marine, Air Force, and Army air operations. Most recently, Soar has been applied to the development of command entities under DARPA's Command Forces and Army air operations. Most recently, Soar has been applied to the development of command entities under DARPA's Command Forces (Soar/CFOR) effort (Hartzog & Salisbury, 1996; Salisbury et al., 1995), including command and control nodes (such as an AWACS) and command decision makers (such as an RWA company commander).

Soar is an outgrowth of GPS, Simon and Newell's attempt to build a General Problem Solver (Newell & Simon, 1963). All tasks in Soar are formulated as attempts to achieve goals in problem spaces. Each problem space consists of a set of states and a set of operators, states representing situations and operators representing actions. Operators perform the basic deliberate acts of the system. They can perform simple, primitive actions that modify the internal state and/or generate primitive external actions, or they can perform arbitrarily complex actions, such as executing a mission. The basic processing cycle is to repeatedly propose, select, and apply operators of a given problem space to a state. Operator selection, application, and termination are all dynamically determined by the system's knowledge, stored in the form of productions (condition-action rules). Any changes in goals, states, and perceptions can cause these productions to fire. There is no conflict resolution, so all applicable productions are allowed to fire in parallel.

Whenever there is insufficient or conflicting information on what activity to do next, Soar reaches what is called an impasse. This triggers the architecture to create a subgoal in which to resolve the conflict. Productions devoted to the subgoal can then fire, possibly leading to new subgoals, before the conflict is finally resolved. These subgoals disappear whenever the impasse is resolved. Subgoals serve three important functions. First, they dynamically make explicit the lack of knowledge in problem solving, making it possible to deliberately reason about how to resolve an impasse at a higher level. Second, they serve an organizing function. Much as traditional programming languages modularize code into separate procedures, Soar productions are organized based on the subgoals to which they are relevant. Third, subgoals facilitate Soar's ability to learn. As an impasse is resolved, Soar records the dependencies underlying this resolution and creates one or more new productions that prevent similar impasses from arising in the future. This process is known as chunking and is a form of explanation-based learning (Mitchell, et. al 1986; Dejong and Mooney, 1986).

\footnote{Soar's ability to learn has not been exploited in the STOW related agents described here and is only utilized in experimental versions of the agents}.

Soar's architecture makes it well suited for developing agents that must reason in complex and dynamic domains. Flexible and adaptive behavior is one of Soar's primary strengths. In addition, Soar allows the smooth integration of planning and reaction in decision-making (Pearson et al., 1993; Laird & Rosenbloom, 1990).

NOTABLE IMPLEMENTATIONS:

Soar has been used as the foundation for the implementation of two significant aspects of synthetic decision-making -- the first aspect addresses the control of individual vehicles and the second aspect is concerned with group command and control. (Soar has also been applied to a large number of synthetic domains outside of the domain of computer generated forces including intelligent tutoring, simulated soccer players, and game playing.) The Soar/CFOR project is associated with the Synthetic Theater of War (STOW) program and is part of DARPA's Command Forces Program (Hartzog & Salisbury, 1996). The Soar/CFOR project is actively working on both types of decision-making through the development of Soar/IFOR (IFOR stands for Intelligent Force) and Soar/CFOR (CFOR stands for Command Force) agents, which are described in greater detail below.

Soar/IFOR pilot agents have been developed to fly fixed-wing aircraft (FWA) and rotary-wing aircraft (RWA) in battlefield simulation environments (Tambe et al., 1995; Laird et al., 1995). This effort focuses on the appropriate command and control decisions at the vehicle level. Soar/IFOR pilots participated in STOW-E, a simulated combat exercise that included expert human pilots, and in the ED-1 exercise, and they are expected to participate in the upcoming STOW97 exercise. Recent work has expanded to include the modeling of command and control vehicles, such as a forward air controller and an E2-C for the Airborne Early Warning (AEW) mission.

Soar/CFOR agents have been developed to explicitly simulate the command and control level of decision-making in a battlefield simulation environment. So far, an attack helicopter (AH-64 Apache) company commander has been implemented as a Soar/CFOR agent (Gratch ref?). In contrast to the issues faced by vehicle or platoon level units, Soar/CFOR agents must model decision-making from a broader perspective and over longer time scales. Whereas vehicle-level decision-making tends to be more reactive in nature, higher-echelon units must deliberate about alternative courses of action, project effects into the future, and detect harmful (or beneficial) interactions between subordinate units and enemy forces.

Although Soar/IFOR and Soar/CFOR share the underlying Soar architecture, the demands of command decision-making at different levels have led to differences in the higher-level organization of these agents. The greater focus on temporal and interaction reasoning has led the Soar/CFOR development to incorporate methods from the AI planning literature, particularly, work on hierarchical task-decomposition, or hierarchical task network (HTN) planning (Stefik 1981, Erol, et al. 1994, Ambros-Ingerson and Steel, 1988). The Soar/IFOR entities, though they do have deliberation capabilities, are more focused on reaction than

planning and bear more similarity to so-called reactive planning systems (as in Agre & Chapman, 1987; Suchman, 1987).

The Soar/CFOR agents integrate the Soar problem solving architecture with the Command Force (CFOR) Infrastructure developed by Mitre for DARPA (Salisbury et al., 1995). The CFOR Infrastructure provides several services that are used by Soar/CFOR: (1) a library of terrain reasoning utilities, (2) the Command and Control Simulation Interface Language (CCSIL) and a set of communications services for sending CCSIL messages to other entities, including entities in ModSAF, and (3) an interface to the commander's vehicle in ModSAF.

The current Soar/CFOR agent implements the company-level Command Entity (CE) for an Attack Helicopter Company (Apache, AH-64). The Soar CE receives and analyzes an Operations Order from its command headquarters. Using its HTN planning capabilities, the CE performs the necessary course of action analysis to generate an Operations Order for its company. In the course of developing the company plan, it makes calls to functions in the CFOR infrastructure to analyze aspects of the terrain. For example, during its initial planning the Soar/CFOR commander may seek to find a route to a battle position that maximizes cover and concealment while minimizing the distance traveled and the exposure to threat by known enemy forces. After the company has begun its mission, it may come across an unexpected enemy force, and, depending on a number of different factors, the situation may require the commander to quickly re-plan its route to minimize the threat from the enemy force. The route re-planning may not take into consideration the same factors as the initial route, so it may call the CFOR route planning function with a different set of criteria that are more appropriate for the current situation. So, if the commander decides to engage the unexpected enemy, it may want a route that maximizes cover and concealment near the point of engagement. But if the commander decides to evade, then it may seek a route that maximizes the distance from the weapons of the enemy force instead of seeking the shortest route to its destination, while still staying within the timing constraint of the mission.

Once the Soar/CFOR CE has developed a plan, it transforms the plan into an Operations Order and issues it as a CCSIL message to its company of Soar/IFOR agents and commences monitoring the execution of the plan. As described in the previous paragraph, the commander will often have to re-plan portions of the mission when unexpected situations occur. The commander makes tactical decisions about whether to evade, suppress, or attack an unexpected enemy force, and it attempts to generate changes to the current plan to appropriately alter the company's route and behavior.

TECHNICAL DETAILS:

COMPUTATIONAL REQUIREMENTS

Soar is implemented in 'C', so it is easily ported to a variety of different computer platforms. Though Soar has primarily been used on Unix platforms, it has also been ported to the Macintosh, and an earlier version of Soar also ran under Windows. The port to the Windows could be done again, if desired. Soar can also be run on Linux on a Pentium computer.

The Soar/IFOR executable code is directly linked with ModSAF so that the two run in a single

process. The requirements for Soar/IFOR are driven mostly by the size and portability of ModSAF and the number of Soar/IFOR agents one wants to run within a single ModSAF. A minimum hardware set-up for Soar/IFOR (with ModSAF) includes 96 MB RAM (128 MB or more is preferred), 500 MB of disk space (1 GB is preferred for development purposes), and a processor with a speed of 150 MHz or greater. Depending on the amount of RAM and the clock speed of the processor, three to eight Soar/IFOR agents can be run on one machine. The hardware systems on which Soar/IFOR is currently implemented include the SGI Indigo and the Sun Sparcstation. It is possible that Soar will be run on Pentium Pro's running Linux for STOW-97.

The Soar/CFOR executable code is not linked directly to ModSAF, rather, it communicates with ModSAF entities using CFOR Infrastructure library routines that use remote procedure calls (RPC). Consequently, the Soar/CFOR computational requirements are somewhat less than for Soar/IFOR. The Soar/CFOR executable is, however, linked to the CFOR Infrastructure, which includes libraries of functions for communicating with ModSAF and for performing terrain analysis. Though the executable Soar/CFOR code is less than 3 MB, the amount of memory required at run-time can be much greater, depending on the size of the terrain database that is loaded and used for terrain analysis and route planning. Since some terrain databases are quite large -- on the order of 50 MB -- the run-time RAM requirements can be on the order of 64 MB, independent of other system requirements. The Soar/CFOR agent is currently implemented on a Unix platform (SGI Indigo) with 96 MB RAM.

PERFORMANCE POTENTIAL

HARDWARE IMPLICATIONS

Soar is very portable and currently runs on a standard Unix-based workstation or Macintosh. Soar/IFOR has the same constraints and requirements as ModSAF. Soar/CFOR has similar requirements to Soar/IFOR, though it requires somewhat less memory. Soar/CFOR has the same constraints as Mitre's CFOR Infrastructure software, which is generally restricted to a Unix platform.

NETWORK IMPLICATIONS

For the Soar/CFOR software to communicate with ModSAF, there must be a way for it to make RPC connections over the network.

REMARKS

Soar is available for distribution at no charge-- it has been placed in the public domain and includes an extensive manual.

APPLICABILITY TO COMMAND DECISION MODELING.

POTENTIAL AS A PRIMARY OR SOLE TECHNICAL APPROACH

Soar is very well suited for implementing the higher level reasoning functions of the command agent, but it does not, by itself, support all the requirements for terrain reasoning and vehicle

dynamics used in our simulations. These capabilities were achieved by linking Soar with other software packages designed for these purposes (a relatively simple process). Soar/CFOR, for example, integrates the Soar problem solving architecture with the terrain reasoning utilities implemented in the CFOR Infrastructure.

SUBPROBLEMS ESPECIALLY SUITED TO THIS TECHNOLOGY

Soar/CFOR is suitable for modeling agents that require sophisticated reasoning capabilities, decision making, as well as plan execution and reactive planning in simulated environments. This technology is applicable to both Blue Force and Opposing Force entities.

Soar is a good architecture for solving problems requiring large bodies of knowledge about behavior, such as extensive doctrine and tactics. Soar/IFOR and Soar/CFOR are both such systems.

WEAKNESSES AND CONCERNS

A Soar command entity is only as good as its knowledge. Considerable knowledge acquisition is required to implement a Soar entity that makes good decisions.

In the current version of Soar/IFOR only a limited number of Soar agents (approximately eight) can be run on the same machine simultaneously. This is the case because of the amount of memory and computation required by each entity. The trade-off to consider in choosing to use a Soar/IFOR agent over a simpler model is to pay a higher cost for the execution environment to get a higher quality simulation.

The Soar problem solving architecture is somewhat difficult to learn, though a recent complete rewriting of the manual, with the needs of the novice particularly in mind, should help somewhat. In addition, a one week Soar tutorial is now offered, which is also accompanied by a large amount of documentation and examples.

REFERENCES

[Agree & Chapman, 1987]

Agree, P. E., and Chapman, D., "Pengi: An Implementation of a Theory of Activity," National Conference on Artificial Intelligence, 1987, pp. 268-272.

[Ambros-Ingerson & Steel, 1988]

Ambros-Ingerson, J.A. and Steel, S., "Integrating planning, execution, and monitoring," Proceedings of the National Conference on Artificial Intelligence, 1988, pp. 83-88.

[DeJong & Mooney, 1986]

DeJong, G. and Mooney, R., "Explanation-based Learning: An Alternative View," Machine Learning 1, 2, 1986, pp. 145-176/

[Erol et al., 1994]

Erol, K., Hendler, J., and Nau, D.S., "HTN Planning: complexity and expressivity," Proceedings

of the National Conference on Artificial Intelligence, 1994, pp. 1123-1128.

[Gratch, 1996]

Gratch, J.A., "Task-decomposition planning for command decision making," Proceedings of the Sixth Conference on Computer Generated Forces and Behavioral Representation, STRICOM-DMSO, July, 1996, pp.37-45.

[Hartzog & Salisbury, 1996]

Hartzog, S. M., Salisbury, M.R., "Command Forces (CFOR) Program Status Report," Proceedings of the Sixth Conference on Computer Generated Forces and Behavioral Representation, Orlando, Florida, July, 1996.

[Laird et al., 1995]

Laird, J.E., Johnson, W.L., Jones, R.M., Koss, F., Lehman, J.F., Nielsen, P.E., Rosenbloom, P.S., Rubinoff, R., Schwamb, K., Tambe, M., Van Dyke, J., van Lent, M., Wray, R.E., III, "Simulated Intelligent Forces for Air: The Soar/IFOR Project 1995," Proceedings of the Fifth Conference on Computer Generated Forces and Behavioral Representation, STRICOM-DMSO, May, 1995, pp. 27-36.

[Laird et al., 1987]

Laird, J.E., Newell, A., Rosenbloom, P.S., "Soar: An architecture for general intelligence," Artificial Intelligence 33(1):1-64, 1987.

[Laird & Rosenbloom, 1990]

Laird, J.E. and Rosenbloom, P.S., "Integrating execution, planning, and learning in Soar for external environments," in Proceedings of the National Conference on Artificial Intelligence. Menlo Park, California: The AAAI Press, July, 1990.

[Mitchell et al]

Mitchell, T. M., Keller R. M., and Kedar-Cabelli, S. T., "Explanation-based Generalization: A Unifying View", Machine Learning 1 (1): 1986, pp. 47-80.

[Newell, 1990]

Newell, A., Unified Theories of Cognition. Harvard University Press, Cambridge, Massachusetts, 1990.

[Newell, 1992a]

Newell, A., "Unified Theories of Cognition and the role of Soar," In Michon, J. and Akyurek, A. (editor), Soar: A Cognitive Architecture in Perspective. Kluwer Academic, Cambridge, Massachusetts, 1992.

[Newell, 1992b]

Newell, A., "Precis of Unified Theories of Cognition," Behavioral and Brain Sciences. 15:425-492, 1992.

[Newell & Simon, 1963]

Newell, A., and Simon, H.A., "GPS, a program that simulates human thought, in Computers and Thought," E.A. Feigenbaum and J. Feldman (Eds.), McGraw-Hill, New York, 1963.

[Pearson et al., 1993]

Pearson, D.J., Huffman, S.B., Willis, M.B., Laird, J.E., and Jones, R.M., "A symbolic solution to intelligent real-time control," IEEE Robotics and Autonomous Systems, 11:279-291.

[Rosenbloom et al., 1991]

Rosenbloom, P.S., Laird, J.E., Newell, A., and McCarl, R., "A preliminary analysis of the Soar architecture as a basis for general intelligence," Artificial Intelligence 47(1-3):289-325, 1991.

[Rosenbloom et al., 1993]

Rosenbloom, P.S., Laird, J.E., Newell, A. (editors), The Soar Papers: Research on Integrated Intelligence. MIT Press, Cambridge, MA, 1993.

[Rosenbloom & Laird, 1994]

Rosenbloom, P.S., Laird, J.E., "On Unified Theories of Cognition: A response to the reviews," In W.J. Clancey, S.W. Smoliar, & M.J. Stefik (editors), Contemplating Minds: A Forum for Artificial Intelligence, pages 141-165. MIT Press, Cambridge, MA, 1994.

[Salisbury et al., 1995]

Salisbury, M.R., Booker, L.B., Seidel, D.W., Dahmann, J.S., "Implementation of Command Forces (CFOR) Simulation," Proceedings of the Fifth Conference on Computer Generated Forces and Behavioral Representation, 423-430, Orlando, Florida, May, 1995.

[Stefik, 1981]

Stefik, M., "Planning with constraints," Artificial Intelligence, 16(1981), pp. 111-140.

[Suchman, 1987]

Suchman, L. A., Plans and Situated Actions: The Problem of Human/machine Communication, Cambridge University Press, 1987.

[Tambe et al., 1995]

Tambe, M., Johnson, W.L., Jones, R.M., Koss, F., Laird, J.E., Rosenbloom, P.S., and Schwamb, K., "Intelligent agents for interactive simulation environments," AI Magazine, 16(1):15-39, Spring, 1995, AAAI

[Return to the Top of this Section](#)

[Return to the Section 4 Index](#)

[Return to the Table of Contents](#)

[Return to the SCC Command Decision Modeling Page](#)

[Return to the WARSIM 2000 Page](#)

[Return to the NSC Home Page](#)
