RAND'S EXPERIENCE IN APPLYING ARTIFICIAL INTELLIGENCE TECHNIQUES TO STRATEGIC-LEVEL MILITARY-POLITICAL WAR GAMING

Paul K. Davis

April 1984

P-6977

The Rand Paper Series

Papers are issued by The Rand Corporation as a service to its professional staff. Their purpose is to facilitate the exchange of ideas among those who share the author's research interests; Papers are not reports prepared in fulfillment of Rand's contracts or grants. Views expressed in a Paper are the author's own and are not necessarily shared by Rand or its research sponsors.

The Rand Corporation, 1700 Main Street, P.O. Box 2138, Santa Monica, CA 90406-2138

RAND'S EXPERIENCE IN APPLYING ARTIFICIAL INTELLIGENCE TECHNIQUES TO STRATEGIC-LEVEL MILITARY-POLITICAL WAR GAMING*

Paul K. Davis**

April 1984

---

*This paper was prepared for an invited address at the Summer Computer Simulation Conference of the Society for Computer Simulation, to be held at the Copley Plaza Hotel in Boston, Massachusetts, July 23-26, 1984.

**Dr. Davis is currently Director of the Rand Strategy Assessment Center. He served in the Department of Defense between August 1977 and August 1981, as Director of Special Regional Studies and as Acting Deputy Assistant Secretary of Defense (Regional Programs).

SUMMARY

This paper highlights some recent experience in Rand's Strategy Assessment Center (RSAC), a large-scale DoD program to develop new concepts and techniques combining features of war gaming and analytic modeling. The centerpiece of the program is a system for automated war gaming in which some or all political and military national decisions can be made by automatons, and in which both force operations and combat are described by theater and strategic-level models.

The RSAC development program is providing a wealth of technical and managerial lessons in adapting and extending such artificial intelligence (AI) techniques as scripts, production rules, English-readable programming languages, goal-directed search, and pattern matching. Most previous AI applications have dealt with smaller and less-complex problems, and have not had to combine AI techniques with those of well-structured system programming and algorithmic combat modeling. Also, the RSAC integration effort has brought together professionals from at least a half-dozen cultures with good ideas but different notions of what constitutes good practice and natural logic. The experience has been illuminating, and the emerging synthesis is unlike previous simulations of which we are aware.

CONTENTS

SUMMARY iii

FIGURES vii

Section
I. INTRODUCTION 1

II. SELECTED PRECEPTS AND IMPLICIT MINDSETS FROM ARTIFICIAL INTELLIGENCE 3

III. BACKGROUND ON THE RAND STRATEGY ASSESSMENT CENTER 7

IV. MANAGEMENT OF COMPLEXITY 13

V. OBSERVATIONS AND CONCLUSIONS 21
A Review of Precepts 21
Management Challenges with a Diversity of Cultures 22
Suggestions 24

REFERENCES 27

FIGURES

1. Selected Basic Precepts from Expert Systems and AI 3
2. One View of the Anatomy of an Ideal Expert System 4
3. Steps in the Development of an Expert System 5
4. Automating a Strategic War Game 8
5. Generic RSAC Objectives for the Automated War Gaming System 10
6. Dimensions of Hierarchical Complexity 14
7. Decomposition of the Problem into Different Classes of Decisions..... 14
8. A Hierarchical View of the RSAC Game 15
9. A Hierarchical View of an Individual RSAC War Plan 17
10. Hierarchical Format of an Individual Theater War Plan 18
11. Mapping of Objectives into AI and other Techniques 19

I. INTRODUCTION

For the last two and one-half years I have directed a large Rand Corporation program attempting to develop concepts and simulation techniques for automated war gaming in support of strategic analysis.[1] The purpose of this paper is (1) to draw on experience in that program to make personal observations about precepts of current work in artificial intelligence (AI) and expert systems,* (2) to discuss some of the special challenges in attempting to develop a *large and complex* simulation with many AI features, and (3) to offer suggestions about concepts, techniques, and managerial measures that may be of use to others in the future.

It is often desirable to express one's point of view at the outset rather than let the reader infer it over time and possibly misunderstand it during the interim. First, I should note that my background is in theoretical chemistry and physics, strategic technology, policy analysis, and defense planning--and not artificial intelligence or even computer science. To be sure, I have had considerable experience with theories and models, but not as a programmer or simulation modeler. As one might expect, given my background, I have by no means been overwhelmed by the mystique of artificial intelligence (AI) as a subject apart--to the contrary, I entered the scene concerned with national security problems and interested in using whatever theories and tools seemed appropriate, preferably without too much baggage. Thus, some of my suggestions will be consistent with those who would debunk the whole business of AI.

That said, the fact is that I am very enthusiastic about what AI has already provided and what it offers for the future. One does not have to accept literally the exaggerated claims of the news media (or the AIers when waxing enthusiastic in news interviews) to recognize that there is something real, powerful, and different here. I reject out of

*Expert systems are artificial intelligence systems embodying enough rules extracted from experts to permit the systems to perform well at some difficult expert tasks. They also emphasize explanation capabilities and evolutionary system improvement.

hand the views that AI is "just another set of programming techniques," that "we've always done those things that people now call AI," and "AI is only for toy problems in universities." I have heard all of these claims--and have sometimes seemed to accept them in conversations where I sought no controversy with traditional modelers or analysts--but it is clear to me the empirical and intellectual evidence is much to the contrary. Certain giants of the AI field (I think particularly of Herbert Simon) have had a profound effect on the way we think about problems and the tools we bring to bear. They have indeed affected our very *paradigms*, and from my point of view the new paradigms meet a felt need and are far more natural and powerful than the old ones for many problems, including strategic analysis and synthesis.*

None of this means that I accept everything going on under the rubrics of artificial intelligence and expert systems. Indeed, I feel that there are some misconceptions prevalent, some considerable obfuscation through abstraction and jargon, and some red herrings under chase. I also know from direct experience about the hurdles in moving from concepts described in the vernacular of AI to operating man-machine simulations dealing with complex problems. Hence, this paper. In the subsequent sections I review my image of AI precepts, describe the program I direct, relate it to AI concepts and precepts, and offer some conclusions.

---

*An example here is decisionmaking about conflict escalation, where policymakers rely heavily on qualitative heuristics summing up a body of values, psychological attributes, and general impressions about the quantitative issues. For a related discussion, see Glaser and Davis,[2] and Davis and Stan.[3]

II. SELECTED PRECEPTS AND IMPLICIT MINDSETS FROM ARTIFICIAL INTELLIGENCE

Because artificial intelligence and expert systems are new fields, there is still disagreement about what constitute basics. It is surely dangerous for a nonexpert to attempt a characterization, but Fig. 1 represents my attempt to identify some key AI precepts. It draws especially on the excellent books by Hayes-Roth, Waterman, and Lenat;[4] Barr and Feigenbaum;[5] and Simon.[6]

The last precept in Fig. 1 does not appear to me to be much emphasized in expert-system work but has extraordinary importance for large and complex problems. It is certainly an important precept in more general AI work.[5,6]

Turning now to the question of what an expert system is, it is interesting to view in Fig. 2 a particular idealization suggested by Frederick Hayes-Roth.[4] This consists of a *language processor* through which the user acts; a *knowledge base* including both facts and rules; a

- **The payoff is often on *knowledge, heuristics, and symbolic reasoning* rather than formal theories, algorithms, and numerical analysis**
- **One should exploit *domain-specific simplification and structures* rather than general problem-solving techniques**
- ***Man-machine interactions are important,* even after the expert system is operative**
- **The expert system should have *transparent reasoning* understandable to users and experts, not just programmers**
- **The *knowledge engineer plays a key role* by extracting knowledge from the experts, translating that knowledge into a well-structured expert system, and working with experts to improve the system iteratively**
- **One should decompose systems of knowledge into *modules* and exploit *hierarchical principles,* even if the decomposition is imperfect and there remain residual interactions**

Fig. 1 -- Selected Basic Precepts from Expert Systems and AI

blackboard consisting of a plan for problem solving, an agenda of steps, and both interim and final solutions; an *interpreter*; a *scheduler*; and a *consistency enforcer*. If this formulation of "the anatomy of an ideal expert system" appears mysterious in a first look, the reader is in good company. Let me merely say at this point that AIers tend to look at problems and systems differently from others (they also tend to use *notional* diagrams that upset system designers and modelers). More on this later.

If Fig. 2 represents in some notional sense the product, then Fig. 3 describes the idealized process by which one builds an expert system. This process is well discussed by Buchanan et al. in Chapter 5 of Ref. 4.

Without getting ahead of the story too much at this point, let me make a few observations about *my* impressions about the expert-system paradigm. Basically, the image is that a "knowledge engineer" goes out,

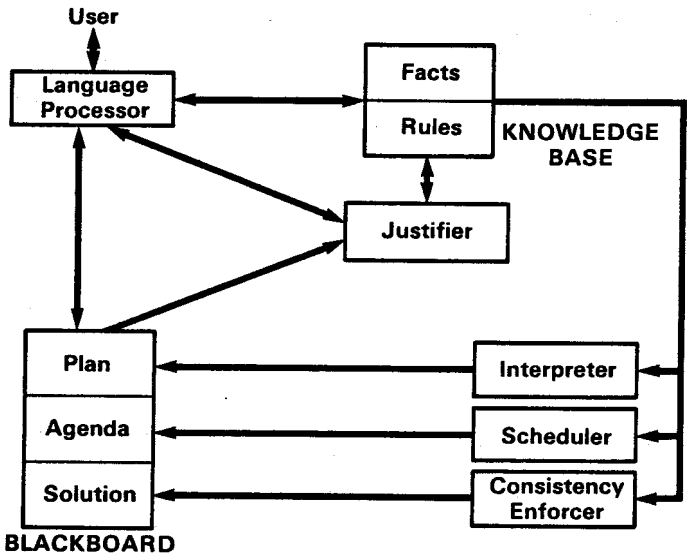


Fig. 2 -- One View of the Anatomy of an Ideal Expert System

- **Identify the problem and its characteristics (Identification)**
- **Find concepts to represent knowledge (Conceptualization)**
- **Design a structure to organize knowledge (Formalization)**
- **Formulate rules to embody the knowledge (Implementation)**
- **Validate rules (Testing)**

(with iterative procedures throughout)

— from Hayes-Roth, Waterman, and Lenat (1983:139)

Fig. 3 -- Steps in the Development of an Expert System

finds himself a tame expert, bleeds the expert dry of rules, builds a system, tests the system with his tame expert, and then polishes everything up. The expert must be knowledgeable, patient, and dedicated to the effort. However, he is not solving the problem so much as providing a data dump of unstructured knowledge for the knowledge engineer to work with. The image is also conveyed that while it is important to find natural representations of knowledge, it is less important to find a real *theory*. Indeed, proponents of expert systems often seem to have a veritable antipathy for theory, especially decision analysis or anything else deductive. This seems peculiar given the highly theoretical nature of AI research, but I have concluded it is merely an overreaction to the discovery of heuristics and procedural representations. Some, of course, will argue that I have misread and mislistened--but I don't think so. Emphasis on words like "knowledge" and "rules" rather than words like "models" and "theories" reveals a particular attitude.

Another element notable by its absence in expert-system discussions is the subject of system programming. To be sure, there may be diagrams discussing the idealized parts of the overall system and what appear to be flow charts (e.g., Fig. 2). However, the basic image is that once the knowledge is reasonably organized by the knowledge engineer, it should be possible to use standardized AI programming techniques to complete the system--e.g., special programming languages and software packages to aid in writing and organizing rules. Implicit here is the important assumption that the problems at issue will not be too large or complex. That assumption does not apply in my program's work.

III. BACKGROUND ON THE RAND STRATEGY ASSESSMENT CENTER

Having provided a list of allegedly key features from the world of AI and expert systems, I shall now describe the program I have been directing.[1] In doing so I shall attempt to relate the entities of our work to the abstractions of the AI literature.

I should make clear at the outset that the Rand Strategy Assessment Center (RSAC) is by no means structured as an AI research project. To the contrary, it is very much an applications program. To the extent possible consistent with the technical problems we are dealing with and the fact that many of the techniques are at the frontiers of research, discussion in the RSAC focuses on military issues and military problems. Moreover, we are not interested in "toy problems" with the potential for more to come in ten years: we hope for significant capabilities within a year or so.

The RSAC had its origins in a DoD initiative begun five years ago to develop fundamentally new concepts and methods of strategic analysis that would somehow combine the best features of war gaming and analytic modeling. The interest in war gaming, stimulated largely by DoD's Andrew Marshall, was due to the desire to increase the richness of analysis by enforcing a global view with strategic forces present in a context of military campaigns, political constraints, the actions and inactions of allies, escalation and de-escalation, and imperfect command and control.

Unfortunately, war games in the past have been slow, expensive, and narrow in scope: working through one war game, however interesting, is not sufficient to permit deducing policy conclusions because one doesn't usually have a good sense for what would have happened in a different scenario--with a different person playing the President or the Soviet leader, or a different Control Team making decisions about the results of combat operations or the decisions of third countries. The DoD hoped that some change of approach would permit greater rigor and produce something of analytic value.

In addressing this challenge, Rand concluded rather early-on, in preliminary work led by Carl Builder,[7] that to gain control over the enormous number of variables in a war game it would be necessary, or at least highly desirable and efficient, to *automate* the game. Figure 4 summarizes the basic concept, which involves replacing all the human players of a traditional military-political war game with automatons called "agents." In this paper I shall restrict my discussion almost exclusively to the Red, Blue, and Scenario Agents, which incorporate artificial intelligence concepts. The Force Agent models are highly interesting in themselves but are better discussed elsewhere.

Continuing, then, note that given an automated simulation, one could reintroduce humans as desired--e.g., with a Blue team playing against the Red Agent in the environment created by the Scenario Agent and Force Agent. If the various agents could be made intelligent enough, then the result could be a powerful mechanism for exploring concepts of strategy and the implications of different capabilities. There was substantial skepticism about the feasibility of achieving such intelligence, but a preprototype demonstration in January of 1981

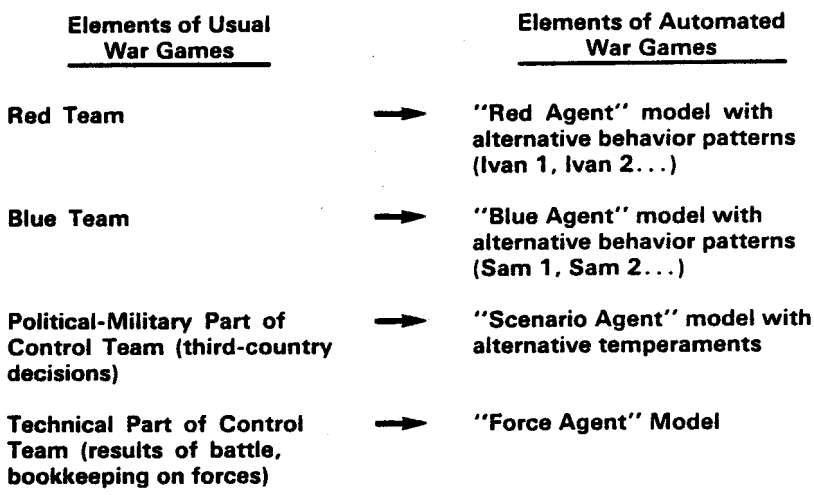


Fig. 4 -- Automating a Strategic War Game

convinced government officials to explore the concept further. The original Red Agent, developed by William Jones and James Gillogly, could choose among certain strategic-level force actions on the basis of a pattern-matching scheme. The Red Agent worked in a fifteen-dimensional space involving dimensions such as conflict location and level. For a relatively small number of points in that space, actions had been prescribed as background "data." When a new situation would arise in a war game, the Red Agent would choose the action prescribed for the data point closest to the new situation (a calculation based on a Euclidean metric and subjective scaling of the various "apples-and-oranges" dimensions). The Red Agent was essentially a black box, but it demonstrated to observers that an automaton could be programmed to make decisions rapidly and plausibly.*

Also demonstrated in 1981 was an early version of the Scenario Agent developed by James Dewar, William Schwabe, and Thomas McNaugher in Rand's ROSIE language.[9] This automaton made decisions for third countries such as the UK and France about their involvement in conflict and cooperation with superpowers. By contrast with the Red Agent, Scenario Agent's logic was at least superficially transparent because the underlying rules were written in the English-like ROSIE, could be retrieved upon command, and could even be changed interactively.**[10,11] Although the original Scenario Agent was by no means sophisticated or adequately substantive, it again suggested a remarkable potential. Subsequent work by Schwabe and others[12,13] has substantially improved the Scenario Agent with more and improved rules, improved structure, and a new and more specialized English-like language called ABEL, developed by Norman Shapiro, Mark LaCasse, and Edward Hall. ABEL is probably 500 to 1000 times faster than the original ROSIE. It is a preprocessor for C and is not interpretive.

---

*The Mark I system was not documented but is discussed along with details of a later intermediate system in W. Jones, J. LaCasse, and M. LaCasse.[8] The pattern-matching technique was based on a design by Norman Shapiro and can be used on a wide variety of problems.

**ROSIE is a general-purpose interpretive AI language discussed in Refs. 4, 10, and 11. Rand is reprogramming ROSIE to operate on a C base and expects to have a performance improvement of 50- to 100-fold by Fall 1984.

The Force Agent of the early RSAC work was only partly automated but demonstrated the value of having highly aggregated and flexible combat models for both conventional and strategic combat.

In summary, then, Rand concluded early-on that the best approach to the challenge posed was to build an automated war gaming system that could be used for analysis as a big model or into which human players could be inserted when desired. The automated war gaming system would employ a variety of artificial intelligence automatons as well as highly parametric force models. What was not established early was how to move from these initial ideas to an effective system for dealing credibly with the real-world complexities of strategy analysis. More concepts were needed.

In worrying about this issue, there have been several periods of deep soul searching to establish better the RSAC's challenges and options. For the purposes of this paper, let me simplify and aggregate to produce the set of generic requirements in Fig. 5.

I am always struck by how innocuous this list of requirements seems until one begins to worry about what the individual phrases really mean. In fact, these requirements are extremely challenging and have forced us

- ***Military realism*** **for global start-to-finish scenarios**
- ***Intelligent agents*** **making wise and/or realistic decisions, and learning in the course of conflict**
- **Some degree of *rigor and completeness***
- ***Transparency*** **of logic and interactiveness**
- ***Transportability*** **and *evolutionary potential***
- ***Speed and flexibility*** **in examining sensitivities and alternative scenarios, and in allowing human play**

Fig. 5 -- Generic RSAC Objectives for the Automated War Gaming System

to deviate fundamentally from standard modeling procedures. Without going through all of them in detail, let me just make some observations. First, I note that as a minimum achieving military realism implies: (1) discarding optimizing models based on simple-minded images of war; (2) dealing with political-military constraints and command-control problems; (3) relegating to secondary status quantitative criteria for actions such as nuclear escalation, in preference to using more qualitative assessments of risks and values; and (4) paying attention to cybernetic phenomena in which decisions are not really made consciously at the top at all. In the cybernetic view, processes systematically avoid the necessity of continual top-level decisions. Most of the time, actions follow what Simon has sometimes called recipes--those doing the actions don't really "think about" what they're doing, much less ask about cosmic goals or the relationship of their actions to what is going on elsewhere. Instead, they merely monitor a few feedbacks to permit adjustments--thus resulting in low-key decisions only tacitly related to goals.*

When described in this way, the cybernetic paradigm sounds realistic but also stupid. However, there is a different way to describe all this having to do with the *management of complexity*. Large organizations and organisms can exist effectively only because of near decomposability and, usually, hierarchical structures. The former phenomenon allows the various subsystems to go their own way and work their own problems most of the time.[6,14,15] The latter phenomenon allows the whole to depend upon weakly but importantly coordinated building blocks rather than on an infinite number of discrete and disorganized processes.

If military realism implies treating constraints and cybernetic processes, then achieving intelligence in the automatons also implies giving them the capability sometimes to take a broader view and make decisions that address global goals and options explicitly. The agents need also be able to game the opponent, to change assumptions about the

---

*There is a rich literature on related matters. See, for example, Simon,[6] and Steinbruner.[14] Note also that the cybernetic model is particularly well suited to decisionmaking dominated by competing organizations or committees rather than by a single rational actor.

opponent, and to learn more generally as the game proceeds. The agents should be capable of decision-analytic reasoning (e.g., maximizing expected utility) and various substitutes such as what I call *global satisficing* (i.e., using heuristics to find an adequate solution that addresses global issues).

Continuing through the list of Fig. 5, transparency and interactiveness suggest reliance upon English-like programming languages, displays, and other AI techniques, but the requirements for speed, flexibility, and transportability run in the opposite direction to some extent and rule out use of "standard" expert-system software. Thus, the RSAC problem demands specialized system software.[16]

Overall, the most important point to make is that the challenge we took on involved incredible complexity, and in this respect our work is quite different from that in most expert-system applications. There are other differences as well, notably: (1) the lack of experts (no one has fought a nuclear war and no one has even fought a modern war between NATO and the Warsaw Pact); (2) the difficulty in measuring system performance; (3) the importance of being sensitive to the system's limitations and capabilities (meta knowledge); (4) sheer scale (thousands of rules); and (5) the importance, in some parts of the work, of quantitative calculations and algorithmic approaches. *Thus, the RSAC simulation is not an expert system, although it may be regarded as having expert-system components.*

IV. MANAGEMENT OF COMPLEXITY

It should now be clear that at the heart of any success we have achieved or will achieve will be our management of complexity. There is, of course, the general principle of breaking the problem down into parts and hierarchies. Unfortunately, the principle never tells you *which* parts or hierarchies.

Finding the "right" hierarchical representation has been neither straightforward nor lacking in controversy--in part because there are several dimensions of hierarchical structure and in part because of cross-cultural problems within the program. In any case, given our emphasis on military realism and war gaming, *the most important dimensions of complexity* are those in Fig. 6: *command level, time, and descriptive detail.* Figure 6 should be viewed as having "trees" opening to the right. Within the dimension of command level we have: a strategic level, which coordinates among supertheaters; a supertheater level, which coordinates among theaters; and so on. Here you can think of Europe as a typical supertheater with three subordinate theaters (northern, central and southern Europe). Within the time dimension, we are concerned with the *campaign view of warfare* (the only natural view for a war gaming approach). Thus, we need to distinguish clearly among the phases of war, the moves within each phase, and so on. Finally, there is the dimension of descriptive complexity, something that every modeler or programmer understands as a problem: everyone agrees that top-down programming is virtuous but there is much less agreement on what is top and what is bottom.

With this background, then, Fig. 7 shows the conceptual way in which we have decomposed the general problem of writing decision rules for the automated war game. One can think of the game as nothing but a collection of decisions (and related actions), but the decisions come in many sizes and shapes. Fig. 8 then shows a hierarchical view of the RSAC war game. Figure 9 shows a hierarchical view of an individual multitheater war plan and Fig. 10 indicates the format of a single theater's war plan.

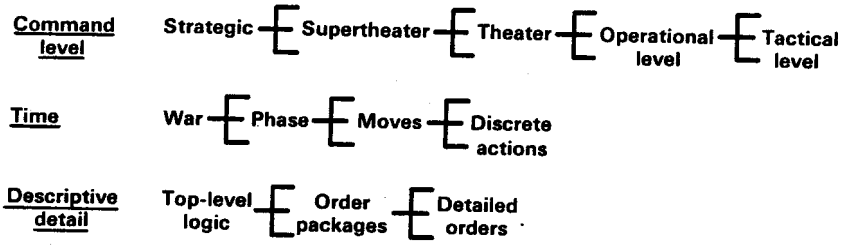


Fig. 6 -- Dimensions of Hierarchical Complexity

- Developing long-term-grand and grand strategies
  - Alternative "Ivans and Sams" with coherent decision patterns
  - These are inputs to a war game

- Choosing among war plans in conflict*
  - Rules structured by Ivan or Sam and *situation* in game
  - Corresponds to *National Command Level* decisions
- Developing alternative war plans*
  - Rules structured by command, time, and detail for a given plan
  - Corresponds to *Area Command Level* decisions
  - Includes guidance to operational and tactical commands
- Managing forces within constraints of a war plan
  - Algorithms and rules structured by functional activity
  - Corresponds to continual *Operational and Tactical Command Level* decisions

*To be treated with separate modules and English-like computer code.

Fig. 7 -- Decomposition of the Problem into Different Classes of Decisions

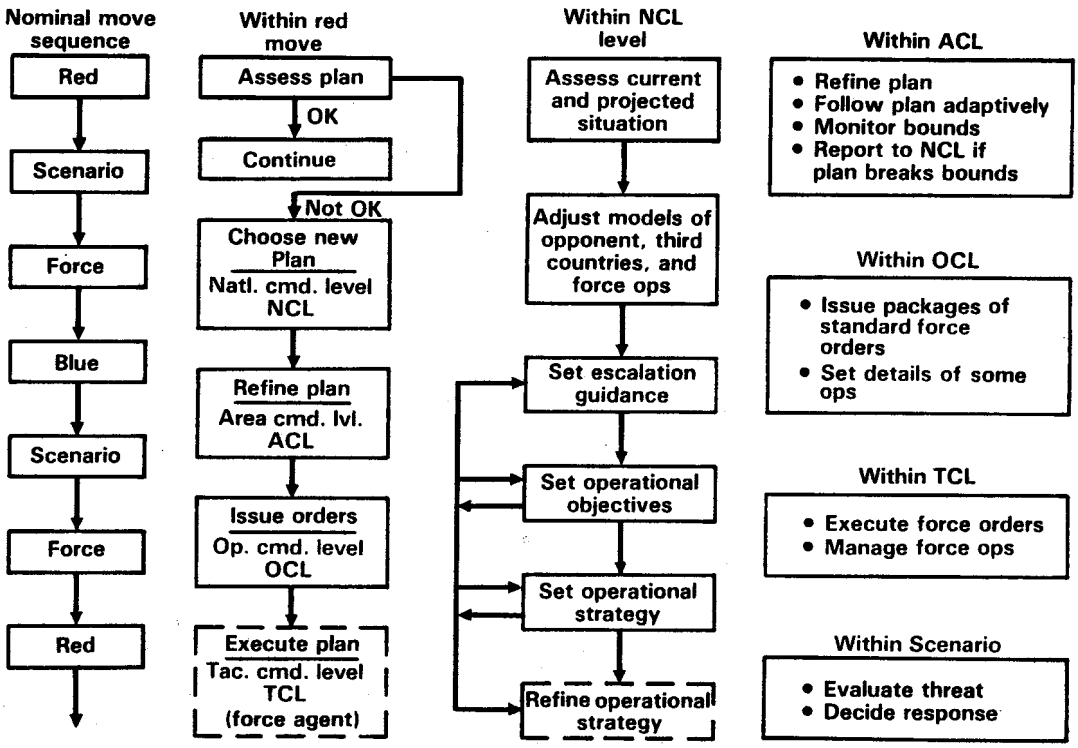


Fig. 8 -- A Hierarchical View of the RSAC Game

Figure 8 reveals a great deal about the overall simulation. The left column describes the *nominal* move sequence in which the Red Agent moves, the Scenario Agent makes political decisions for all the third countries, Force Agent executes all the orders to force elements and assesses probable combat outcomes, Blue moves, etc. The order is merely nominal because the real simulation's move sequence depends on events.[16]

The second column describes more fully what happens in a single Red move. First, let me emphasize that *Red and Blue are always following war plans*,* but these war plans can be changed. The issue when Red has an opportunity to move is whether the current war plan is proceeding successfully. If so (on the basis of criteria within the plan), Red

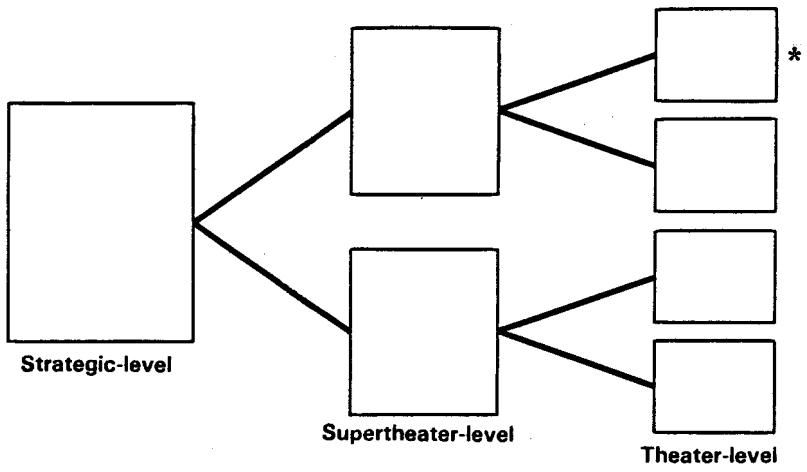
*The concept of focusing on war plans traces back to a paper by the author in early 1982.[17] The AI techniques for doing so were described by Randall Steeb and James Gillogly.[18] The overall structures of Figs. 7 and 8 are due to work by the author, Peter Stan, and other colleagues.[1,3,17,18,19]

skips his move. If not, he must choose a new plan. This means that control in the computer program is transferred to a module associated with National Command Level (NCL) functions. That module's rules have the responsibility for choosing a new war plan. In the simplest case, the choice is clear-cut and the NCL's chosen plan is transferred to a module associated with the Area Command Levels (roughly, the equivalents of real-world theater commands). This module's rules fill out certain details of the plan in accordance with details of the combat situation. Then, yet a third module, the Operational Command Level, issues force orders to be executed by the Force Agent, which embodies in its models information about standard operating procedures, orders of battle, and tactical decision rules.

The decision process is not usually so simple. Instead, the NCL's choice of plans is initially tentative and subject to further analysis. Let us assume that a single plan appears adequate on the basis of some heuristic rules within the NCL. The next step usually is a *lookahead*, which consists of a game within a game using Red's assumed model of his opponent (Red's Blue), his assumed models of third countries (Red's Scenario), and his assumed models for the results of force operations, (Red's Force). If the result of the lookahead is success for the tentatively chosen plan, then Red implements the plan. If not, he chooses a second one and tries again. Thus, control moves through the sequence shown in the second column at least twice in most cases--once to test a tentative plan and once to execute the chosen plan. We have also allowed for *comparing* plans before making the final decision, although we do not employ dumb search techniques, which would waste information and time.

Let us now examine in more detail what happens within the NCL module (the third column of Fig. 8). As mentioned earlier, the NCL's rules are organized by situation, with situations corresponding roughly to levels of a generalized escalation ladder (although with no biases built in about rungs necessarily being higher or lower than others).[3] For a given situation such as a world state involving conventional wars in Southwest Asia and Europe, the NCL starts with a body of potential war plans developed off line by human analysts drawing upon realistic concepts of war plans. The successive NCL rule modules filter the

acceptable plans by ruling out (or selecting, depending how one writes the rules) classes of war plans on the basis of decisions on escalation guidance, operational objectives, and strategy objectives. Before doing so, the NCL has the opportunity to learn from experience so far in the war by modifying his models of opponent, third countries, and force operations. In any case, after going through a process as in Fig. 8's third column, the NCL emerges with a tentative war plan to be tested and, if successful, implemented. As the feedback arrows indicate, if a plan fails, the NCL attempts to find a new plan by first adjusting operational strategy, then operational objectives, and finally escalation guidance.* Although I will not discuss them here, the fourth column shows simplified descriptions of what happens within the other rule-based modules. Development of the ACL procedures and related war plans has been due to William Jones, Norman Shapiro, and William Schwabe.



* Note: NCL picks a multitheater plan

Fig. 9 -- A Hierarchical View of an Individual RSAC War Plan

*The reader should note the anthropomorphic nature of this system description, which is both natural substantively and a characteristic of expert-system programs.

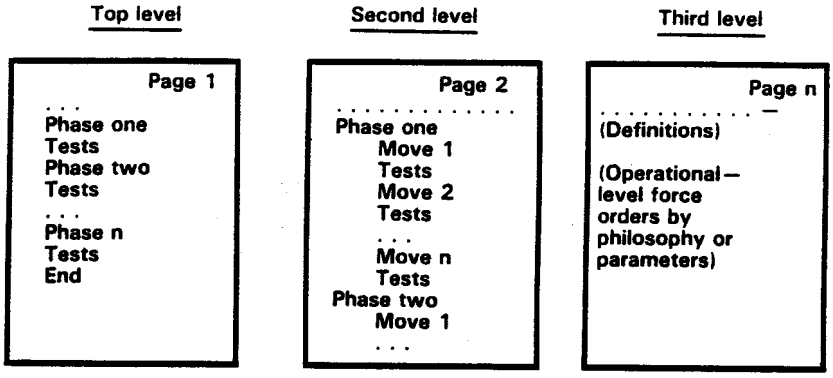


Fig. 10 -- Hierarchical Format of an Individual Theater War Plan

Let me now draw on Figs. 7 and 8 to describe how we have attempted to accomplish goals of Fig. 5 in terms of AI (and other) techniques. This mapping is summarized in Fig. 11. Not shown are the goals of transportability and evolutionary structure, which we are trying to accomplish with good system programming and reliance on the highly flexible C programming language and Unix operating system. These are especially suitable for AI applications and have also proven quite acceptable for our combat modeling--even as judged by people trained in the Fortran and PL/I schools.[16]

Working through Fig. 11 would be an adequate basis for an entire talk. Let me merely touch upon a few highlights here:

- *We associate war plans with an extension of AI scripts*--time-tagged sequences of events with production rules (if...then...rules) dictating choices at branch points, mappings to allow the war plan to adapt to circumstances, and bounds to determine when the war plan/script is no longer applicable.*[18]

---

*The original notions about *scripts* (and the related *frames*) were presented in work by Minsky [20] and by Schank and Abelson.[21] RSAC scripts are substantially more sophisticated, and include procedures for attachment, synchronization, self-evaluation, branching, and adaptation.

	Requirement			
	Military Realism	Intelligent Agents and Learning	Rigor and Completeness	Transparency and User Friendliness
Relevant Features of Approach	• Overall *game structure* • Rules consistent with a nations's imputed attributes and grand strategy • AI *scripts* • Models organized around processes • Emphasis on constrained satisficing	• Mechanisms for decision analytic, global satisficing, and process cybernetic rules • Coherency enforced by agent attribute checklists and idealized grand strategy decision trees • Both qualitative and quantitative heuristic criteria for decisions, including criteria based on inherent values • Basic game design, which permits lookaheads, adaptation, and learning	• Formal state spaces and transition matrices • Itemized objectives, escalation guidances, and strategies • Structured decision trees	• Top-down structures • Natural-language programming • Process models and other modularities • Interactive programming • Displays

Fig. 11 -- Mapping of Objectives into AI and other Techniques

- The war plans/scripts are organized hierarchically by command level and, for each theater command, by campaign, phase, move, and action (Figs. 9 and 10). Actions in the various theaters are imperfectly coordinated. As a whole the war plan is "almost decomposable" in the sense of Simon.[6]

- The NCL rules for choosing new war plans at key decision points (determined by the bounds within the war plans) are organized by situation to permit more nearly optimal decisions with a global view. However, even at the NCL level we can have the mix mentioned earlier of decision-analytic, global satisficing, and process-cybernetic decisions.

- The coherence of NCL decision rules is achieved by characterizing the Ivan or Sam for which the rules are being written using attribute checkoff lists (Is Ivan adventurous, operationally flexible, etc.?) and blackboard-quality grand-strategy decision trees. These do not in themselves imply the individual rules, but they give the expert rule-writers an image to which to be faithful.

- Because of the nature of our work it is unacceptable to have unrecognized holes in the rule base. There will always be holes, but we need to recognize them as well as possible. Hence, we emphasize formalized state spaces and decision trees/tables structured to make it relatively easy to see what

cases have been covered. The agents complain to the user when asked to make decisions in situations for which they have only the most general guidance. This emphasis on system structure, process models, transition matrices, state spaces, and decision tables provides substantial *meta knowledge*.

In summary, even though the RSAC is not really an AI research project per se and is by no means a single expert system, it incorporates a broad range of AI techniques and has extended them considerably. Moreover, it may be regarded as involving a number of separate expert systems (Red Agent NCL, Red Agent ACL,...Scenario Agent). We have enough experience so far to know that virtually everything mentioned so far "works"--with one important caveat: it is still too early to claim that we have succeeded in managing the complexity and building adequately substantive rules and war plans. The program is still in the infancy of its field and what we have achieved so far is only the beginning of what is possible. The automated war game "runs," with all the processes referred to in the various figures I have shown. However, we do not yet have anything like an adequate number of substantive war plans or NCL rules. Also, we have a long way to go before we will be satisfied with our force models, interactive capabilities, and so on. Nonetheless, we are far enough along to draw some conclusions and observations--the subject of the last section.

V. OBSERVATIONS AND CONCLUSIONS

A REVIEW OF PRECEPTS

At the outset of this paper I noted precepts from AI and expert systems (Fig. 1). I also paraphrased descriptions on what an expert system should consist of and what steps one should go through in building one (Figs. 2 and 3). Let us now review them and see how they compare to Rand's experience. My principal observations here are:

- I would agree heartily with all the precepts of Fig. 1 except that involving the knowledge engineer (more on this later). Moreover, I would assert that in my experience these precepts have simply not been emphasized by pre-AI modelers. Had it not been for the AI influence, we would have had difficulty forming the right paradigms to work with qualitative issues, command structures, cybernetic mechanisms of decision, etc. Moreover, we would not have emphasized the clarity of logic and interactiveness that are becoming a major factor in our work.

- The AI precept of hierarchical modularity and nearly decomposable systems is fundamental to our management of complexity.

Now, two caveats and exceptions:

- Our work has necessarily mixed heuristic and algorithmic approaches to a far greater degree than might seem natural to a dyed-in-the-wool heuristics buff.

- My experience has *not* supported the image of a "precertified" knowledge engineer as the essential intermediary. In my experience, *pure* AI experts have provided excellent paradigms and ideas, and have been useful consultants. However, they have not been the ones to build the system. Instead, individuals deeply interested in the problem area, but with no particular background in AI or computer science, picked up the requisite paradigms, studied expert-system techniques, immersed themselves in the problem, and became de facto knowledge engineers. They have worked closely with less specialized computer scientists on programming and system design.

With respect to what a system should look like, I would say first that neither the RSAC simulation nor any of its parts in any sense "looks like" the Hayes-Roth idealization in Fig. 2, in which one sees no mention of the underlying model, theory, or conceptual structure and is encouraged to see knowledge as a mere collection of rules. In the RSAC, the image is different: the model and structure are all-important and directly related to software entities. Thus, entities such as the Red Agent NCL and the many Red Agent ACLs are all represented by separate coroutines.[16]

To be sure, this is a bit unfair. Taking Fig. 2 less literally, and instead using it to identify *functions* of a well-conceived expert system, then I can surely identify with it:

- We use natural-language programming (in Rand's ABEL) and data editors for the language processor.
- We separate out parts of the knowledge base (e.g., the various war plans or scripts) from the structure of the system.
- To a significant degree we treat rules as data--or at least patterns of rules (e.g., one can readily change Ivan or Sam).
- The blackboard function, which I will not elaborate upon here, is also present. For example, Fig. 8 shows in the form of a hierarchical process model how the Red Agent goes about solving his problem: the process model constitutes a plan with a systematic agenda and intermediate solutions. And, of course, there must be control software to enforce the logic of this problem-solving concept. Thus, one can find evidence of distributed interpreters, different schedulers, and even consistency enforcers.

MANAGEMENT CHALLENGES WITH A DIVERSITY OF CULTURES

By this time it should come as no surprise to learn that we have had certain management challenges in trying to apply AI techniques to the RSAC problem. A factor here has been the existence of multiple cultures, which have included: AI experts, non-AI computer scientists, applications modelers and programmers, political scientists, international relations experts, physical scientists, economists,

mathematicians, engineers, retired military officers, historians, and behavioral psychologists.

I shall not characterize each of the above groups for fear of providing unintended insult (to my own parent culture as well as others), but I suspect that the reader can fill in himself. In any case, given this diversity in a single program, remarkable phenomena emerge. I have observed all of the following:

- Problems with consistency in level of description.
- Difficulties in moving from conceptual designs in the style of AI[18] to a rigorous system design.
- Utter confusion in discussing feedback phenomena familiar to control theorists and physical scientists, but intuitively anathema to system designers who view any diagrams with upward pointing arrows as something bordering on sin, and who may never have used a text with "feedback" in the index.
- Disagreements between analysts preferring intuitive models described by undisciplined flow charts with arrows and Go To statements, and aficionados of structured programming with multiple nested structures.
- Disagreements about programming languages and programming features (LISPers versus non-LISPers, AIers versus Fortraners and PL/Iers, strong typers versus weak typers...).
- Emotional attachments to inefficient natural-language techniques that are transparent line-by-line but opaque as a whole. Resistance to decision tables, which can appear to natural-language enthusiasts as a step backward.
- Disagreements about whether decision rules should be or can be highly structured, analytical, and complete, or whether they should be heuristic and ad hoc.*
- Conflicting images of who the user would be, and related disagreements about the top, middle, and bottom of top-down programming (e.g., to a programmer, declaration statements may be very important).

---

*There is a general tendency to believe that expertise can be captured by a moderate number of well-structured rules--except in your own domain of expertise.

- Lengthy and often esoteric debates about software design engendered by fear of theoretically feasible but substantively bizarre circumstances that could be assumed away or dealt with *once the problem was recognized*.

- The relative lack of concern by some AIers toward *system* problems and the stubbornness of some traditional programmers to appreciate some of AI's abstractly described power.

- Pure problems of jargon counter to the objective of transparency to users ("Push," "Pop," etc.).

This partial list should be adequate to convey the following image: even with high professionalism, good will, and a fine work environment, interminable man-hours have been spent over issues with their origins in culture gaps. In some cases it has taken literally months to recognize that certain communication problems existed.

SUGGESTIONS

In spite of all, we have had enough success so far that I feel capable of offering up a few tentative suggestions for those of you who may be considering AI applications to highly complex problems rather than the smaller-scale (e.g., 100-rule) expert-system problems more commonly discussed:* In addition, I can assure you that such applications can be exciting and challenging. With that observation, then, my suggestions are as follows:

- In spite of the power of heuristics and the risk that attempts to be rigorous will result in paralysis, *theoretical structures are important*. Without them, one gets systems with little knowledge of their own shortcomings (little meta knowledge, to use the AI term), only superficial transparency, and the potential for disaster.

- *There should be early agreement on hierarchical structuring of problems where appropriate*.

---

*I also recommend Buchanan's advice in Chapter V of Ref. 4.

- *Focus on the real problem* to avoid wasting time on the search for unreasonably general solution techniques.

- Because of cross-cultural problems, *emphasize highly specific real-world examples* to illustrate anything under debate. Otherwise, one may get nonconverging abstract debates.

- *Complement modelers and AI experts with experts in system software.* The skills are different and complementary.

- *Pictures are worth a thousand words*--even for discussions among highly intelligent people who "ought" to be able to understand each other without decision-tree diagrams written on a blackboard. Be aware, however, diagrams, like slang, can have different interpretations by software experts and modelers.

- *Construct simple systems early*--to clarify the vision and provide a base of common experience to team members.

- *Enforce issuance of quasi-documentation along the way* rather than waiting for respectable documentation.

- *Use natural-language programming* but include explanatory figures as comments and make good use of economical and transparent mathematical expressions and decision tables.*

- *Consider breaking decision problems down into process models for clarity* even if that imposes a logic not everyone finds natural.

- *There is no substitute for deep interest in the subject area.* Training people with some subject-area experience to serve the knowledge-engineer function may often be more practical and productive than searching at length for the right precertified knowledge engineer.

---

*In Rand's ABEL language, decision tables appear directly as computer code. The tables are essentially decision trees rotated 90 degrees. The result is easy to understand if the variables are clearly named and at the right level of abstraction.

REFERENCES

1. Davis, Paul K., and James A. Winnefeld, *The Rand Strategy Assessment Center: An Overview and Interim Conclusions About Potential Utility and Development Options*, The Rand Corporation, R-2945-DNA, March 1983.

2. Glaser, Charles, and Paul K. Davis, *Treatment of Escalation in the Rand Strategy Assessment Center*, The Rand Corporation, Santa Monica, N-1969-DNA, April 1983.

3. Davis, Paul K., and Peter J. E. Stan, "Concepts and Methods for Modeling Conflict Escalation," The Rand Corporation, forthcoming.

4. Hayes-Roth, Frederick, Donald A. Waterman, and Douglas B. Lenat (eds.), *Building Expert Systems*, Addison-Wesley Publishing Company, Inc., Reading, MA, 1983.

5. Barr, Avron, and Edward A. Feigenbaum (eds.), *The Handbook of Artificial Intelligence*, Vol. I, William Kaufmann, Inc., Los Altos, CA, 1981.

6. Simon, Herbert A., *The Sciences of the Artificial*, 2nd ed., The MIT Press, Cambridge, MA, 1981.

7. Graubard, Morlie H., and Carl H. Builder, *Rand's Strategic Assessment Center: An Overview of the Concept*, N-1583-DNA, The Rand Corporation, September 1980.

8. Jones, William, Jean LaCasse, and Mark LaCasse, *The Mark II Red and Blue Agent Control Systems for the Rand Strategy Assessment Center*, N-1838-DNA, The Rand Corporation, October 1983.

9. Dewar, James A., William Schwabe, and Thomas L. McNaugher, *Scenario Agent: A Rule-Based Model of Political Behavior for Use in Strategic Analysis*, N-1781-DNA, The Rand Corporation, January 1982.

10. Fain, J., D. Gorlin, F. Hayes-Roth, S. J. Rosenschein, H. Sowizral, and D. Waterman, *The ROSIE Language Reference Manual*, N-1647-ARPA, The Rand Corporation, December 1981.

11. Fain, J., F. Hayes-Roth, H. Sowizral, and D. Waterman, *Programming in ROSIE: An Introduction by Means of Examples*, N-1646-ARPA, The Rand Corporation, February 1982.

12. Schwabe, William, and Lewis M. Jamison, *A Rule-Based Policy-Level Model of Nonsuperpower Behavior in Strategic Conflicts*, The Rand Corporation, R-2962-DNA, December 1982.

13. Shlapak, David, et al., "The Mark III Scenario Agent for the RSAC," The Rand Corporation, forthcoming.

14. Steinbruner, John D., *The Cybernetic Theory of Decision*, Princeton University Press, Princeton, 1976.

15. Pattee, Howard H. (ed.), *Hierarchy Theory*, George Braziller, New York, 1973.

16. Shukiar, Herbert J., "Overview of RSAC System Software: A Briefing," The Rand Corporation, N-2099-NA, forthcoming. See also a related paper in the Proceedings of the Summer Computer Simulation Conference of the Society for Computer Simulation, forthcoming.

17. Davis, Paul K., and Cindy Williams, *Improving the Military Content of Strategy Analysis Using Automated War Games--A Technical Approach and an Agenda for Research*, The Rand Corporation, N-1894-DNA, June 1982.

18. Steeb, Randall, and James Gillogly, *Design for an Advanced Red Agent for the Rand Strategy Assessment Center*, The Rand Corporation, R-2977-DNA, May 1983.

19. Davis, Paul K., Peter J. E. Stan, and Bruce W. Bennett, *Automated War Gaming As a Technique for Exploring Strategic Command and Control Issues*, The Rand Corporation, N-2044-NA, November 1983.

20. Minsky, M., "A Framework for Representing Knowledge," in P. Winston (ed.), *The Psychology of Computer Vision*, McGraw-Hill, New York, 1975.

21. Schank, R. C., and R. P. Abelson, *Scripts, Plans, Goals, and Understanding*, Lawrence Erlbaum, Hillsdale, NJ, 1977.

RAND/P-6977

RAND'S EXPERIENCE IN APPLYING ARTIFICIAL INTELLIGENCE TECHNIQUES TO STRATEGIC-LEVEL MILITARY-POLITICAL WAR GAMING

Paul K. Davis