

AD-A172 782

ARTIFICIAL INTELLIGENCE CONCEPTS AND THE WAR GAMING

1/2

ENVIRONMENT: A CASE S. (U) AIR FORCE INST OF TECH

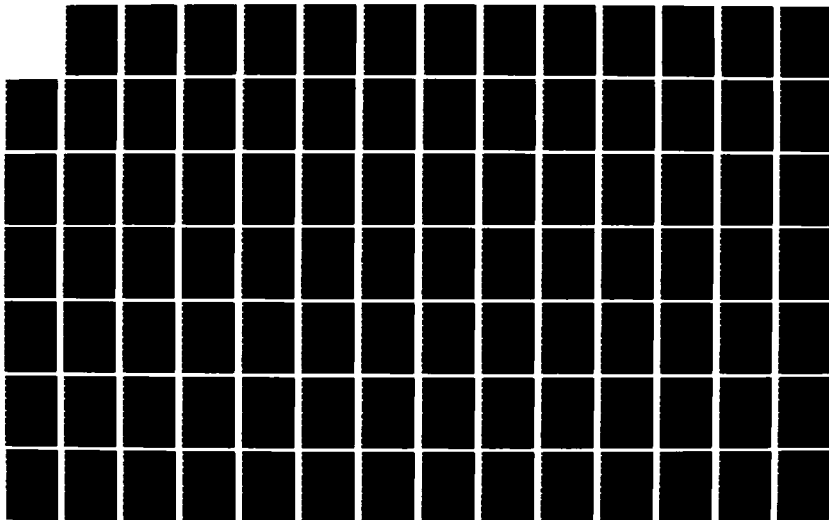
WRIGHT-PATTERSON AFB OH SCHOOL OF ENGI G B WHITE

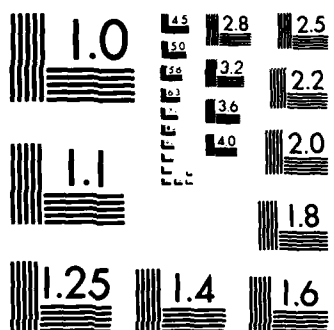
UNCLASSIFIED

MAR 86 AFIT/GCS/ENG/86M-1

F/G 15/7

NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

AD-A172 782



ARTIFICIAL INTELLIGENCE CONCEPTS
AND THE WAR GAMING ENVIRONMENT:
A CASE STUDY USING THE TEMPO WAR GAME

THESIS

Gregory B. White
Captain, USAF

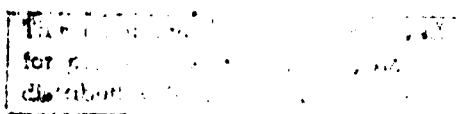
AFIT/GCS/ENG/86M-1

DTIC FILE COPY

OCT 1 1986

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio



86 10 10 091

AFIT/GCS/ENG/86

ARTIFICIAL INTELLIGENCE CONCEPTS
AND THE WAR GAMING ENVIRONMENT:
A CASE STUDY USING THE TEMPO WAR GAME

THESIS

Gregory B. White
Captain, USAF

AFIT/GCS/ENG/86M-1

Approved for public release; distribution unlimited.

AFIT/GCS/ENG/86M-1

ARTIFICIAL INTELLIGENCE CONCEPTS
AND THE WAR GAMING ENVIRONMENT:
A CASE STUDY USING THE TEMPO WAR GAME

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
Air University
in Partial Fulfillment of the
Requirements for the Degree of
Master of Science in Computer Engineering



Gregory B. White, B.S.
Captain, USAF

March 1986

Approved for	
By	
Distribution	
Available	
Dist	
A-1	

Approved for public release; distribution unlimited.

Preface

The purpose of this study was to develop a computerized-player version of the TEMPO force planning war game. The immediate outcome of this would be to provide the Squadron Officers School non-residence program with the capability to have TEMPO included in their curriculum in order for them to more closely resemble the residence program. Secondary to this, I wanted to show the feasibility of implementing war games on microcomputers and additionally to show how the use of artificial intelligence techniques could aid in making war games more realistic and useful.

I wish to thank Maj Steve Cross, my thesis advisor, and Maj Steve Woffinden, my thesis reader, for their assistance. They spent many hours helping me to organize and clarify my ideas on this subject and thesis. Finally, I would like to thank my wife Charlan for her never ending support and encouragement throughout my thesis effort.

Table of Contents

	Page
Preface	ii
List of Figures	v
List of Tables	vi
Abstract	vii
I. Introduction	1
Background	2
Thesis Objective and Scope	4
Assumptions and Game Characteristics	4
Equipment Required	6
II. Artificial Intelligence	7
Expert Systems	8
Intelligent Computer Aided Instruction	11
AI Tools	13
III. Computers and War Gaming	14
Current Versions of TEMPO	16
AI Efforts in War Gaming	18
IV. TEMPO	20
Game Instructions	20
Operation of Forces	22
Acquiring Forces	22
Research and Development	23
Modifications to Weapon Systems	23
Intelligence	24
Analysis	25
War	28
Overspending the Budget	29
Weapon System Effectiveness	29
V. The TEMPO Design	31
Design Concepts	31
TEMPO Module Description	33

VI.	The Intelligent Instructor	41
	Knowledge Engineering	42
	Concepts in TEMPO and their M.I rules	45
VII.	Conclusion and Analysis	50
	Enhancements to TEMPO	51
	Analysis of M.I and the intelligent instructor	52
	Recommendations and Areas for Further Study .	52
	The Future of War Gaming	55
	Bibliography	56
	Appendix A (Programmers manual)	58
	Appendix B (code)	75
	TEMPO code	75
	INITWSYS code	120
	TMPOICAI code	125
	Appendix C (Sample game turn)	132
	Vita	146

List of Figures

Figure	Page
1. Knowledge Based System	9
2. Current Force Report and Help Screen	34
3. R&D Report and Help Screen	35
4. Intelligence Report and Help Screen	36
5. Player Function Menu and Help Screen	37
6. War Report	39
7. Comparison of Weapon System	46
8. M.1 Code for Best Weapon System	46
9. M.1 Code for an Overspending Check	48
10. M.1 Code for a Budget Exercise	48
11. M.1 Code for an Overdefending Check	49
12. Deception Code	53
13. War Declaration Code	53

List of Tables

Table	Page
I. Expert System Tools for IBM PC's	43

Abstract

War gaming has long been an accepted practice in the mental preparation of officers for war. With the introduction of computers, ~~war games have become~~ the games have become increasingly sophisticated yet most current war games are either too slow, not realistic, or use the computer as a referee only and not as a player. An approach is discussed in the context of TEMPO, a force planning war game currently used by the Air Force at its Squadron Officers School. This thesis involved the development of a version of TEMPO in which a computer expert system takes the place of one of the players, and an intelligent computer instruction system that takes the place of the section leader. The system is implemented on a microcomputer allowing its use in professional military education seminar courses.



ARTIFICIAL INTELLIGENCE CONCEPTS
AND THE WAR GAMING ENVIRONMENT:
A CASE STUDY USING THE TEMPO WAR GAME

I. Introduction

Testing the theories relating to the art of war has been an accepted practice for several thousand years. Sun Tzu knew of it's importance in 500 BC when he wrote in his treatise The Art of War:

The general who wins a battle makes many calculations in his temple ere the battle is fought. The general who loses a battle makes but few calculations beforehand. Thus do many calculations lead to victory, and few calculations to defeat: How much more do no calculations at all pave the way to defeat! (Tzu, 43)

Eventually this testing of theories and calculations developed into what is now referred to as war gaming, a concept as important today as it was in Sun Tzu's time. Early games felt to mentally prepare individuals for combat included Chess, Go, and Checkers, but these games lost their place as newer war games, complete with maps and counters representing the different military units, were developed. In recent years, the addition of computers to war gaming has added a whole new dimension. One example of a modern war game is the strategic force planning exercise known as

TEMPO. Originally designed by the Technical Military Planning Organization (TEMPO) of the General Electric Company of Santa Barbara as a manually played simulation, this game has recently been programmed for several different systems including the PDP 11/70 and Apple III computers (Owens, 1982:1).

Background

The purpose of a war game is to mentally prepare an individual to make correct strategic decisions pertaining to some particular conflict. To be effective, the war game must therefore represent as closely as possible the conditions that can be expected. The game should also be made accessible to as many individuals as possible in order to gain the most benefit from it. Most current war games, however, are lacking in these areas. The force planning exercise entitled TEMPO is a good example.

TEMPO is currently played at several sites including the U.S. Army Management School, The Naval Postgraduate School, and all three Air Force Professional Military Schools at Maxwell AFB: the Air War College, Air Command and Staff College, and Squadron Officers School (SOS). Using SOS as an example, currently less than half of the Air Force company grade officers receive the opportunity to attend in residence. The rest take part in the non-residence program by participating in it by correspondence. Unfortunately, the correspondence course does not include

the TEMPO game as part of its curriculum. This means that fewer than half of the company grade officers in the Air Force will have had the opportunity to gain the kind of knowledge about the budgetary decisions made during force planning that is available by playing the game. TEMPO additionally suffers from another common drawback which is that the computer is only used as a referee or umpire. The game is played by two opposing sections with the computer calculating and reporting all pertinent game information. This inhibits the usefulness of the game since a single individual would not be able to play the game (a requirement if it were to be used in the correspondence version of SOS). Other war games suffer from this problem to an even greater extent as some require a group of "experts" to play the part of an opposing force (the "red team" concept).

Some war game developers have solved the problem of having to have a group of experts or an additional player by using the computer as both a referee and as one of the players (the idea of a "computerized player"). Many of these, however, suffer from a lack of realism as the computerized player works from a "canned" script or a set of algorithms from which it calculates its moves. This lack of realism is exactly the problem Lt General Raymond B. Furlong addressed when he suggested the incorporation of "unknown unknowns" into Air Force war games (Furlong, 1984:6). In

his article, Lt Gen Furlong discussed the need for uncertainty and chance to be modelled in war games in order to make them seem more realistic.

Thesis Objective and Scope

The objective of this thesis is to design and implement a version of TEMPO that can be used by the SOS non-residence program to give the majority of Air Force officers the opportunity to experience the force planning environment. Doing so also meets an Air Force requirement to write a version of TEMPO that can be modified to meet the varied needs of its different professional military schools. A requirement of this program is that it must run by itself without the need for an umpire or second player/team. This results in a computerized player requirement along with a computerized referee. In addition to implementing this program, the war gaming environment and how artificial intelligence (AI) techniques and ideas can be used in it will be examined. This thesis is not intended to produce a generic design for AI war games but rather to show how AI techniques are used in the implementation of one specific war game. As a side objective, an evaluation of an expert building tool, M.I., will be accomplished.

Assumptions and Game Characteristics

There are four game characteristics that are important in the development of a war game. First of all the war game

should be available to as many officers as possible. This requires that the game be playable on a machine that is available to as large a segment of the officer corps as possible. It is assumed that the majority of Air Force Officers have access to an IBM PC or an IBM PC compatible machine and therefore this machine was chosen as the target machine for this thesis effort. The second characteristic is that the game should be easily modifiable so as to easily reflect real world changes in weapon systems and doctrine. This characteristic (along with the first) drove the selection of the language to be used in this thesis. PASCAL was chosen because of its ease of use, adaptability to sound software engineering practices such as modularity, and the ability of an average programmer to learn it as compared with a more specialized language such as LISP. The third characteristic is that the game should contain a computerized player. This enables the game to be played by a single individual and additionally always provides an "expert" who understands the rules to play against. The later is important because when two novices play, certain concepts might be ignored through a lack of understanding on the participants parts. The last characteristic is that the game have some sort of intelligent tutor to take the place of a human instructor. This is necessary in order to provide the interaction needed to learn from an individuals mistakes. With an intelligent tutor, a novice can learn the concepts embedded in the game faster than if left alone.

Equipment Required

The only required equipment for this thesis was a computer that was either an IBM PC or a IBM PC compatible, and a version of PASCAL. Borlands' TURBO PASCAL was chosen as the preferred version because of its cost and the fact that it is widely distributed and fairly portable. Its portability means that a whole new set of machines will be able to be used to play the game since versions of TURBO PASCAL exist for CPM based machines . The final item required is a copy of Teknowledge's M.1 so that its applicability to war gaming could be studied.

II. The Field of Artificial Intelligence

The field of artificial intelligence (AI) has been receiving ever increasing attention for the past decade. AI means many different things to different people. If someone were to interview people to find out what they felt AI was, answers might vary anywhere from "getting a machine to think" to "creating a robot like in Star Wars." For the purpose of this thesis, Elaine Rich's definition will be used. She describes AI as "the study of how to make computers do things at which, at the moment, people are better." [Rich, 1983:1] Put another way, AI is concerned with developing hardware and software that accomplishes some task that, up to this point, could only be done by a human.

If these definitions are used as guidelines for determining what is or isn't AI, then several fields of study could be thought of as being encompassed by the field of AI. These would include robotics, pattern recognition, natural language understanding, learning, intelligent computer aided instruction, and expert systems. The last two items, intelligent computer aided instruction and expert systems, are of particular importance for this thesis. Expert systems will be discussed first.

Expert Systems

Expert systems are programs (or machines) that, given some specific domain, "think" and "act" as an expert would. An example of one is the medical diagnosis program called MYCIN. MYCIN carries on an interactive dialog with the user, a physician, about a patient who has some sort of infectious disease. Based on the answers it receives, it asks other questions and eventually proposes a possible diagnosis along with an associated therapy for it. There are several elements in this program that make MYCIN an expert system.

The first element that enables MYCIN to be referred to as an expert system is its knowledge domain. Infectious disease is not an area in which everyone is versed, it is an area which is left to a group of experts--in this case physicians.

The second element making MYCIN an expert system is how it uses knowledge it has obtained to structure its line of "thinking". It does not simply go through a set number of canned questions which are the same every time the program is consulted. It instead uses the answers obtained from previous questions to tailor the rest of the questions it will ask to the specific problem at hand.

The third element deals with how MYCIN is based on a "knowledge base" made up of a number of rules which contain the knowledge and experience an expert in the field might

reasonably be expected to have. These rules are referred to as production rules and, along with facts gathered by the program, are the basis for all knowledge based systems (see figure 1). The rules express in simple IF... THEN ... format the knowledge the expert has. An English translation of one of MYCINs rules (they are written in LISP, a common AI language) is as follows:

```
IF    1) the infection is primary-bacteremia, and
      2) the site of the culture is one of the sterile
        sites, and
      3) the suspected portal of entry of the organism is
        the gastrointestinal tract,

THEN  there is suggestive evidence (.7) that the identity
      of the organism is bacteroides. (Barr, 1982:187)
```

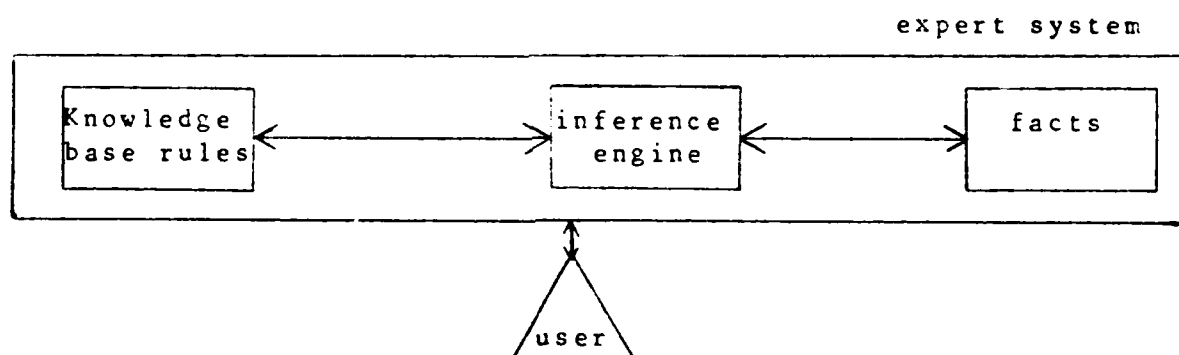


figure 1.

The last element concerns MYCINs' limited explanation capability (while an explanation capability is not a requirement of an expert system, it should be a requirement for a useful expert system). This explanation capability lets the user of the system see the line of reasoning MYCIN used to obtain its results. While it seems that having this explanation capability is a plus for MYCIN, the fact that it is limited to explanations based on its line of reasoning actually negates any possible advantage. Studies by such individuals as Clancey (Clancey, 1979) have shown that to be useful an expert system must be able to do more than just repeat the questions in its knowledge base for which it is seeking an answer. Without a more extensive capability, the program cannot effectively be used to train physicians. Instead it can only be used by physicians who already have a working knowledge of infectious disease. This same concept is true when dealing with Intelligent Computer Aided Instruction systems.

Having used MYCIN as an example of an expert system, it should be easy to now recognize that an expert system is simply a program that performs like an expert would in some, very specific field. One of the most difficult parts in creating an expert system is the collection of the knowledge for the knowledge base. There are several ways to do this which include interviews with an expert, watching an expert perform the task for which the expert system is to be

written, and having the expert "think out loud" as the task is performed. The acquisition and encoding of knowledge has led to a field of its own known as Knowledge Engineering.

Knowledge and knowledge engineering is not only important to expert systems but is also important to ICAI systems. In order for a CAI system to be considered intelligent some sort of knowledge base must exist and be used.

Intelligent Computer Aided Instruction

The concept of computer aided instruction (CAI) has been around for a number of years. Simply put, it's goal is to use a computer as an aid in the instruction of some idea, concept, or skill. Programmers in AI involved in computer aided instruction are interested in taking this one step further. They refer to their work as intelligent computer aided instruction (ICAI). To illustrate the difference between these two, consider a simple program designed to aid a young child in learning addition. The basic program would simply display math problems and wait for the child to enter the answer. Once the answer had been entered, a CAI system would determine if the answer is correct or not, and then report what it had determined to the child along with keeping some statistics on the number of correct and incorrect answers given. This, however, is not enough to be classified as an ICAI system. An ICAI system would attempt to determine

not only what mistake had been made but why it was a mistake and then report its findings. For example, does the child have trouble with addition problems with carries or subtraction problems where borrowing is necessary? Are there certain numbers that the child continually adds or subtracts incorrectly? etc . . . Once the child's problem with math is determined, the ICAI system would add additional instruction in this area, either through further instruction or simply by emphasizing these problems. In this way it uses knowledge about student errors along with some teacher strategies to design and implement a tailored tutoring session for the student. It should be obvious from this example that an ICAI system is more than some elaborate grading machine, but rather something more closely related to a personal tutor that can provide individualized attention to the student.

There are numerous ICAI systems currently in existence with such names as SCHOLAR, WHY, SOPHIE, and WEST (Barr, 1982: 224). One system called GUIDON is designed to teach diagnostic problem-solving using rules from the MYCIN system (Barr, 1982: 267). As was mentioned earlier, MYCIN with its simple explanation capabilities is not capable of instruction by itself. GUIDON's goal is typical of interactive ICAI systems which are designed to lead the student through the steps needed to solve the specific problem at hand. In doing so, it is hoped that the student learns the underlying concepts in the particular problem solving environment.

AI Tools

Currently, symbol-manipulation languages such as LISP and PROLOG are the choice of most builders of expert systems (as well as with other AI applications). A few systems, however, have been built using problem oriented languages such as FORTRAN and PASCAL (Waterman, 1986:82). The reasons for LISPs popularity include the ease in which symbol manipulation can take place and the fact that LISP treats code and data as one and the same thing. This later feature enables LISP programs to add to and modify themselves; a capability necessary in systems that "learn." As the number of expert systems increased, however, other languages (or tools) were designed to aid knowledge engineers in the development of new expert system applications. These tools include such packages as KEE, OPS5, and one developed by Teknowledge called M.1. These tools are designed to make expert systems development easier and faster by already providing the problem solving environment. The knowledge engineer then is tasked just with the formation of the knowledge base rules to be used by the system. An example of one M.1 rule from a small expert system used to help select the correct shutter-speed for a camera is as follows:

```
if asa = 100 and light = sunny
  then shutter-speed = 125.
```

As can be seen, the statement is english-like in format thus making it easy to develop and understand. This is the goal of all knowledge engineering tools.

III. Computers and War Gaming

There are basically two ways in which computers can be used in the war gaming environment. The first, which will be referred to as computer assisted war gaming, is to use the computer only as a referee. This referee performs the mundane calculations used to determine the outcome of moves made by the human players, and then in turn reports these outcomes. The computer is also used to introduce various factors to the game based either on some timetable or the current gaming environment. An area where AI can enhance computer assisted war games would be the addition of an intelligent computer aided instruction system which would evaluate and instruct the players of the game.

The second way of using the computer in war gaming is to actually have the computer take the part of one or more of the players. Use of a computer in this manner results in what is known as a computerized player. Currently, a number of computerized player type war games exist. These work either from a "canned script" or, at best, by using a few simple rules to guide the choice of moves. Canned scripts, while easy to program, suffer from a tremendous lack of realism. They are good only to teach some very basic concepts in war gaming since they totally ignore the ever

changing conditions found on the battlefield. Adding a few rules to increase variability improves the computerized player but still falls far short of what can be expected from a human opponent. However, by careful implementation of the computerized player as an expert system, a tremendous jump in realism can be obtained. Furthermore, an ICAI system can also be added, as was mentioned in the computer assisted war games, to make the war game a true learning/teaching experience.

Looking at computerized war games with an eye towards their implementation as expert systems led to categorizing war games into three areas. The first category contains those small scale simulations in which an algorithmic solution can be found. These games contain few rules, are generally very simple in concept, and are intended to teach just a few basic concepts. TEMPO fits into this category.

The second category consists of those games which contain numerous rules and are more complex. It is also here that General Furlong's concept of "unknown unknowns" becomes important. An example of one of these games would be a combined arms simulation in which numerous units with several different weapon systems represented do battle on varied terrain under ever changing conditions. As one can see, in these games the variables are numerous.

The last category consists of those simulations which rely on things such as "human intuition" and "gut feelings". These games, while not necessarily containing a large number of rules, would be harder to implement since it is hard to define human intuition and gut feelings in terms of production rules. An example of this is a game created by Avalon Hills called **Diplomacy** in which, during the course of the game, players lie to one another in order to gain certain advantages through deception. In order to be able to win a player has to be able to somehow determine who has lied about potential support and who has not. As often as not, alliances are made and broken based on personal reasons rather than qualitative reasoning--a real challenge to any potential programmer trying to simulate these lines of thought.

Current Versions of TEMPO

There are several versions of TEMPO currently written, none of which (as far as the author has been able to find) use the computer for anything more than a referee. As played at SOS, the game is played between two opposing sections of approximately twelve people each. There is a terminal in each section room that is connected to a central computer. Each section decides on what move is to be made and then submits a set of "orders" to an individual whose only job is to enter the orders into the computer. The system cannot be

considered a "friendly" system as the commands need to be exact with no room for mistakes. When both sides have determined their orders and they have been input, the computer calculates which side is ahead and then reports this information along with all other pertinent data. The game is the same every time it is played as it works from a canned script.

Another version of TEMPO, written as a Masters thesis by LT Christopher D. Owens, USN, also uses the computer solely as a referee (Owens, 1982). This version has had a few improvements made to it however. First of all, it was written in PASCAL for the APPLE III Microcomputer. This means that the program is a little more transportable since it is written in a common language for a fairly common system. It also has the advantage that it was designed to allow a third party, referred to as the game umpire, to select the game parameters thus making the game more flexible. Despite these advantages, this version of TEMPO has not only the inherent disadvantages associated with all computer assisted war games, but a few additional ones as well. The game umpire, already mentioned in passing, requires a third party to be present for at least the initial set up. The game itself, along with the data used in playing it, resides on three separate disks requiring the players/umpire to change disks during the game. A final problem with Lt Owens version of TEMPO lies in the physical

makeup of the system. Unless a printer is attached to the system, the players will have to take turns looking at their force posture and entering their orders since there is only one CRT and keyboard attached to the APPLE III computer.

AI Efforts in War Gaming

Rand's Strategy Assessment Center (RSAC) has been doing some work in developing "automatons" to make some or all political and military decisions in a war game. (Davis, 1984:iii). Their emphasis has been in large, complex environments normal to category two type wargames. In fact, Davis talks in term of games that contain "thousands of rules" (Davis, 1984:12).

This is not the only work that Rand has done in wargaming (Klahr, 1982). They have also created a prototype air battle simulation called SWIRL (for Simulating Warfare In the Ross Language). The system was designed to allow users to easily make modifications to the code in order to examine the varied aspects of an air battle. While their work has resulted in a very good battle simulator, it is not a true war game in the traditional sense. Additionally, SWIRL was written for a DEC20 with 112K 36 bit words required for the interpreter alone. If graphics are desired, this is accomplished with the use of two additional graphics processors driven by a PDP 11/45. All told, although their

object oriented approach shows great promise in simulations and possibly war gaming, it is currently not a very transportable system.

IV. TEMPO

The TEMPO game was originally written by the Economic Analysis Section of the Technical Military Planning Organization, General Electric Company, Santa Barbara, California (SOS, 1983:3350-S-2). There have been numerous versions of the game created including the one currently played at SOS. The purpose of the game is to demonstrate the type of force planning that occurs in the U.S. military. The stated goal of the SOS version is "to teach each player something about military force planning and resource management under the constraints of time, uncertainty, and a limited budget." (SOS, 1983:3350-S-2)

Game Instructions

The human player (or section, hereafter referred to as the player) will be playing the computer with each side starting with identical forces. At the start of the game, the player will be given a budget, projected next year's budget, a list of the weapon systems currently in the inventory, a list of proposed research and development (R&D) projects, and an intelligence report. The budget is to be spent on improvements to the players forces and can be used to operate current forces, fund R&D projects, purchase new systems, make modifications to existing systems, or purchase intelligence reports.

There are four different types of weapon systems used in this game. These systems are broken into two categories, offensive and defensive systems. Offensive systems include bombers and missiles, and defensive systems include fighters and anti-ballistic missiles (ABM). Each weapon system has certain data associated with it. First of all it has an acquisition cost which reflects the price of purchasing one unit of that system and its operation for the year in which it is bought. Systems also have an operation cost associated with them. This is the cost to maintain that system in operational status for one year (note that the systems first year of operational cost is contained in the systems acquisition cost). The third figure associated with all systems is the Util value for that system. The Util value represents the "utility" of that system and represents the worth of the system. The final value associated with all systems is the purchase limit. This is the maximum number of that type of weapon system that can be purchased in any one year.

For those systems that are proposed R&D systems, additional information is given such as the number of years until completion of R&D, and the cost of R&D for each of those years. One important factor to note is that the figures given are only an estimate of the costs and values (with the exception of the purchase limit and the current years R&D cost). All other values may change during the research and development of that system.

The game lasts for eight turns, each turn representing one year. The goal of the game is to maximize the net offensive Utils, while operating a defensive force that will minimize the opponents offensive Utils. At the conclusion of each game year, an intelligence report is printed informing the player whether he is ahead or behind in the "Arms race" (whether he is ahead or behind in net offensive Utils). At the conclusion of the game, each side's yearly net offensive Utils count is totaled and the side with the greater total is declared the winner.

Operation of Forces

Each year a list of the current forces owned is printed and the total operations cost for these systems printed. Unless a system is SCRAPPED, the computer will assume that the system is to be operated and the appropriate costs deducted from the budget. Any forces that are scrapped are taken out of the game completely and are no longer available (no "mothballing" of systems). If later on systems that had been earlier scrapped are desired, they must be acquired by purchasing them as a new system.

Acquiring Forces

Acquiring forces is extremely easy. All that has to be done is to pay the acquisition cost and the system is considered in the inventory. As had been mentioned previously, the acquisition cost includes the first years

operational costs so operation costs do not have to be paid for a system the year that it is acquired. The purchase limit for a weapon system cannot be exceeded in any one year.

Research and Development

During the first four years of the game, each side will receive information about proposed weapon systems. These systems all have associated R&D costs which must be paid before a new system can be purchased. Depending on the system R&D can take from one to three years. The process cannot be sped up by paying all R&D costs in one year. Ongoing R&D can be halted after it has been started if the system is no longer desired. This process is referred to as "shelving" the system. Later, shelved systems can have research commenced once again, but there is an associated penalty in doing this. This penalty amounts to 1/2 of the cost of R&D for the year in which it is to be resumed.

Modifications to Weapon Systems

At certain points, R&D resulting in a modification of an existing weapon system (an F-1A for example) may be offered. Modifications always are a one year R&D project and the R&D cost must be paid before any systems can be modified. Additionally, if modifications to a system are desired, you must have some of the original system to modify or else no modifications can occur (there must be some F-1's in the

inventory before they can be modified to be F-1A's). Care must be taken to not scrap systems if modifications are desired on them. You may additionally purchase new units of a modified system by going through the normal acquisition process.

Intelligence

Each side can purchase certain types of intelligence reports which deal with the force posture of the opposing side. There are four types of reports, each costing \$100. These reports deal with the makeup of the opposing offensive forces, defensive forces, offensive R&D projects funded, and defensive R&D projects funded. In addition, there are two other intelligence reports, neither of which costs any money. The first is given at the beginning of each year and reports the chance of a war breaking out that year and the second is given at the end of each year and reports on who is currently ahead in the "Arms race".

The intelligence reports given are not precise, instead they are only a general idea of the opposing forces. The reports dealing with the current offensive and defensive forces provides a ten unit range for each weapon system in the appropriate category (i.e. if the opposing forces had 62 B-1's the intelligence report would state that there were 60-70 B-1's). The reports dealing with the current R&D in progress lists all projects that have been funded in the

appropriate category but does not tell how far along those projects are.

Analysis

The player can obtain an analysis of any weapon system desired. This analysis will be in terms of a Util per dollar percentage which can be compared with similar data obtained for the other weapon systems. This is the real heart of the game since it is based on these figures that the player will make the decisions as to which systems to fund or to drop. The value is obtained using the following equations: (SOS, 1983: 3350-S-13)

$$\text{Util/\$ percentage} = 100 \times \frac{\text{total utility}}{\text{total cost}} =$$

$$\frac{\text{utility year 1} + \text{utility year 2} + \text{utility year} \dots}{\text{R\&D cost} + \text{acquisition cost} + \text{operating cost}}$$

where, Acquisition cost for any given year = the number of units purchased that year X the acquisition cost per unit.

Operating cost for any given year = the number of unit operated that year X the operating cost per unit.

EXAMPLE:

Budget = 1000

	B-1	B-2
R&D costs	200	250
Acquisition Cost/unit	100	200
Operating cost/unit/year	50	75
Utils/unit/year	50	200

Doing a 3 year analysis we find the following.

Year 1		
Budget	1000	1000
Minus R&D	<u>-200</u>	<u>-250</u>
Budget left to buy systems	800	750

Number of systems we can buy	8	6
------------------------------	---	---

Year 2		
Budget	1000	1000
Minus Operations cost	<u>-400</u>	<u>-450</u>
Budget left to buy systems	600	550

Number of systems we can buy	6	4
------------------------------	---	---

Year 3		
Budget	1000	1000
Minus Operations cost	<u>-700</u>	<u>-750</u>
Budget left to buy systems	300	250

Number of systems we can buy	3	1
Total number of systems for B-1 =	17	
Total number of systems for B-2 =	11	

Now lets compute the Util/\$ percentage using

$$\frac{(\text{util year one}) + (\text{util year two}) + (\text{util year three})}{(\text{R\&D cost}) (\text{Acq cost yr1+yr2+yr3}) + (\text{ops cost yr1+yr2+yr3})}$$

For the B-1

$$\frac{(8 \times 50) + (14 \times 50) + (17 \times 50)}{(200) + [(100 \times 8) + (100 \times 6) + (100 \times 3)] + [(0) + (50 \times 8) + (50 \times 14)]} =$$

$$\frac{200 + \frac{400}{800} + \frac{700}{600} + \frac{850}{300}}{200 + 800 + 600 + 300 + 0 + 400 + 700} =$$

$$1950 / 3000 = .65$$

To get the percentage, $.65 \times 100 = 65$

For the B-2

$$\frac{1200}{250} + \frac{2000}{750} + \frac{2200}{550} + \frac{2200}{250} + \frac{2200}{0} + \frac{2200}{450} + \frac{2200}{750} =$$

$$5400 / 3000 = 1.8$$

To get the percentage, $1.8 \times 100 = 180$

It can be quickly seen from this example that the total number of weapon systems cannot be used as a good measuring

stick for how good a system is. In our example we were able to buy more B-1's than B-2's but after performing the analysis we found that we would be ahead in total number of utils if we purchased the B-2's instead.

Another form of analysis that can be made (there is no computer function to perform this, it is entirely left up to the player) is the analysis of the opponents force. This is important in order to decide what type of defensive weapon systems need to be purchased and how many of them are required. The basis for this analysis is the intelligence that was purchased in the previous year. If, for example, we received the following intelligence report

Offensive Forces ---> B-1's 10-20
M-1's 0-10

Defensive Forces ---> F-1's 20-30
ABM-1's 10-20

Offensive R&D -----> M-1A

Defensive R&D -----> F-2

an analysis could be performed as follows (based on the values for utils given at the start of the game):

OPPONENT'S OFFENSE

SYSTEM	INVEN	Acq Cost	Ops Cost	Utils unit	Util Range	Ops cost Range
B-1	10-20	50	20	10	100-200	200-400
M-1	0-10	40	20	30	0-300	0-200
M-1A	Research	100	50	70		R&D Cost 400-500-650

OPPONENT'S DEFENSE

SYSTEM	INVEN	Acq Cost	Ops Cost	Utils unit	Util Range	Ops cost Range
F-1	20-30	40	20	30	600-900	400-600
ABM-1	10-20	50	40	60	600-1200	400-800
F-2	Research	90	70	90		R&D Cost 100-200-120

After creating charts like these, a comparison between the computers and the players forces can easily be made. Charts such as these can aid the player in determining exactly what forces need to be purchased to counter a specific offensive threat. They are also valuable in determining whether there are any wasted defensive utils (sometimes referred to as "futile utils"). Since the players score is based on the net offensive point value, any defensive systems that were not needed to counter a threat are just wasted money.

War

Each year an intelligence report informs the player the percentage chance of a war occurring at the end of that year. The percentage is based on the disparity between the net offensive points of the two sides. The greater the difference between these figures, the greater the chance for war. This represents the likelihood that a war would break out when a side has an overwhelmingly superior offensive force. Even if the two sides are even, there is a certain base chance that a war will occur. If a war does break out,

the losing side will have their budget cut by \$800, and the winning side will have their budget cut by \$200. This will encourage each side to make sure that they have an operational offensive force each year.

Overspending the Budget

This only applies to the player because the computer has been programmed to never overspend it's budget. If the player should happen to overspend the budget in any one year, the next years budget will be reduced by that amount and the computer will arbitrarily eliminate operational forces from the players inventory until the budget is brought in line. One can quickly see that it is therefore not advisable to overspend the budget since the systems that the computer scraps may be the best systems available as opposed to the worst.

Weapon System Effectiveness

The total utils is not calculated simply by multiplying the number in inventory by the utils/unit figure for that weapon system. Instead, this figure is calculated by multiplying the number in inventory by the utils/unit figure for that weapon system and then multiplying that figure by a balance of forces figure. This balance of forces figure insures that there is a balance of forces between different types of offensive or defensive weapons. This does not have

to be a 50-50 balance although it could be. A 2-to-1 ratio is just as good a balance for this game's purpose. A 4 or 5-to-1 ratio, however, is not generally a good balance. The reason for insisting on a balance between forces is to recognize a concept called the Law of Diminishing Returns (SOS, 1983:3350-S-6). It also reflects the idea of the Strategic Triad which has been the basis for our country's strategic defense policy for a number of years.

V. Design

There were three major considerations in the design of the TEMPO war game. The first was to insure the use of software engineering principles such as modularity and documentation. Since the purpose of this thesis was to provide maintainable code, it must be easy to follow and understand. Few environments are stable enough to not require periodic modifications of a program to bring it up to date so careful implementation of the initial software could save countless man hours a few years from now. Both documentation of code and the use of a modular style of programming are accepted practices that aid in the maintenance of software as new requirements are made. Related to this maintenance issue is the issue of changing game parameters. Weapon systems change and for the game to remain useful the systems modelled in the game need to be easily modified. This was accomplished by having all of the weapon system parameters read in from a stored file at the beginning of the game. By having these values stored in a file as opposed to hard coding them, the upkeep of them is greatly facilitated. In fact, a small, seperate program was written which writes these values to the file. This seperate program permits extremely easy modification of the weapon system parameters.

The second important design concept was the design of the data base to be used. Already mentioned was how important it is to be able to modify the weapon system parameters, also important is the ease with which this information can be accessed. Since the entire game revolves around the weapon systems, it is necessary to have easy access to them by the rules and heuristics. If to access some weapon system data requires many convolutions, the time it would take the computer to select its moves would increase. This is not a desirable effect. Another data base consideration was to make it as understandable as possible so that future enhancements or changes would be as simple as possible. The data, therefore, was stored in three separate sections, the weapon systems owned by the human player, those owned by the computer, and a large section containing information on all weapon system parameters. This logical division of the data facilitates future enhancements.

The third design consideration was the interface that would be required to allow the use of an M.I.-based ICAI system. Since M.I. only reads ASCII data, all numeric data needed to be "translated" before it was written to a file. The interface takes the form of a single procedure that writes certain game information to one of eight files representing the eight game turns. Since the files are only written to by one procedure, modification to the files is simple. The type of information currently written includes

such things as whether the player overspent the years budget, and whether or not too much money was spent on a certain type of defensive system. This information, along with the other information written to the files, represent the type of problems that students were found to make. The module that wrote to the different files is called PRNTRPRT and is just one of seven major modules.

TEMPO Module Description

The main routine in TEMPO is very short and consists of a loop with several procedure calls (see Appendix A.). The first routine called is INIT which does all array and variable initialization. It is in this procedure that the file of weapon system parameters is read in. TEMPO then repetitively calls five routines in order; one time for each game turn.

The first routine called in the loop is REPORT. This procedure prints three screens of game reports consisting of a report on the current status of the players weapon systems, one on both the proposed and currently funded research and development projects, and one displaying what is known about the computers forces (known as intelligence reports). Each one of these screens has an associated help screen which explains the meaning of the different output variables. The help screen can be seen by typing an "H". When the player is

finished viewing the help screen, the original screen, from whence the call to the help screen originated, is re-displayed (see figures 2-4).

```

Start of Year 1
Your Budget is $ 9330
The estimated budget for next year is $ 9870

```

SYSTEM	INVEN	PER UNIT			TOTAL		PURCH LIMIT	MOD1 COST
		AQCOST	OPCOST	UTILS	OPCOST	UTILS		
B-1	40	50	30	10	1200	400	30	0
M-1	60	100	30	20	1800	1200	20	40
		Offensive Total			3000			
F-1	20	140	60	20	1200	400	30	60
ABM-1	100	30	20	10	2000	1000	30	0
		Defensive Total			3200			
Total present operating cost					6200			

(enter an E for help or press return to continue)

This screen contains information about your current force structure. It is split into two sections; offensive and defensive weapon systems. The column headings represent the following information:

SYSTEM - the name of a weapon system you have completed research on.
 INVEN - the number currently in your inventory.
 AQCOST - acquisition cost, the cost to purchase one of these weapon systems.
 (it also covers the operating cost for the first year)
 OPCOST - the cost of operating this weapon system for one year.
 UTILS - the util value for one of these systems.
 TOTAL OPCOST - the total operations cost if you keep all of the units you currently have in your inventory of this weapon type.
 TOTAL UTILS - the total utils you will receive if you keep all of these units.
 PURCH LIMIT - the maximum number of this type of weapon you can buy in a year.
 MOD1 COST - the cost of modifying one of these units to an upgraded version.
 (only valid if you pay the research costs for the new version,
 a 0 appears if there is no modification available)

€
 press enter to continue

figure 2.

Proposed R&D Programs								
All of the following are expected values except year 1 R&D cost								
		PER UNIT			R & D COST			PURCH LIMIT
SYSTEM	AVAILABLE IN	AQCOST	OPCOST	UTILS	YEAR 1	YEAR 2	YEAR 3	
B-2	3yrs	130	30	250	590	840	460	20
F-1A	1yrs	90	50	90	190	0	0	20
ABM-2	2yrs	130	60	110	550	420	0	20
M-1A	1yrs	100	40	70	90	0	0	25

REMEMBER---- Systems may be bought the same year final R&D costs are paid
(enter an E for help or press a return to continue)

This screen contains information about proposed research and development (R&D) projects as well as any projects you have funded in the past. This screen is for informational purposes only, you will have the opportunity to fund or shelve projects when you reach the PLAYER FUNCTION MENU. The column headings

SYSTEM - the name of the weapon system.
 AVAILABLE IN - how many years it will take to complete research on the project.
 AQCOST - the cost to acquire one of these units when research is complete.
 OPCOST - the cost to operate one of these units for a year when R&D complete.
 UTILS - the per unit util value for this weapon system.
 R&D COST YEAR x - the cost of funding research on this weapon system for that year. Only the present years R&D cost is known for certain, the rest are just estimates of the projected R&D cost.
 PURCH LIMIT - the maximum number of this weapon system that can be purchased in a year when R&D complete.
 If the word "shelved" appears, it signifies a system you at one time funded research on but later dropped. It will cost you 1 1/2 times the amount of the first years R&D cost to resume.

One final note, remember that systems can be bought the same year the final R&D costs are paid---in other words, when it says "lyrs" in the AVAILABLE IN column.

@
 press enter to continue

figure 3.

----- Intelligence report -----

Intelligence reports indicate chance of war this year = 19 2

Offensive system B-1	range =	40 to 50
Offensive system M-1	range =	0 to 10
Offensive system M-1A	range =	40 to 50
Defensive system F-1	range =	0 to 10
Defensive system ABM-1	range =	90 to 100
Defensive system F-1A	range =	10 to 20
Offensive R & D for B-2	in progress.	
Offensive R & D for M-4	in progress.	
Defensive R & D for F-2	in progress.	

enter an H for help or press return to continue

This screen indicates what your intelligence forces report to you. It is for informational purposes only, you will have the opportunity to dictate what type of intelligence you desire in the ORDERS section of the PLAYER FUNCTION MENU.

As a minimum, you will be told what the chance of war occurring this year will be. This information costs you nothing. The other types of intel will cost you a fixed amount each year you request it. This information will be in the form of the name of the weapon system and a range. This range indicates the upper and lower values for the possible number of weapon systems your opponent currently has in its inventory. Intel reports on R&D indicate projects your opponent has funded.

press enter to continue

figure 4.

The second procedure in the loop is PLAYERTURN. This procedure as its name indicates, handles the human players turn. It consists of a loop which displays an option menu and three procedures which can be executed any number of times. The three procedures, named REPORT(the same procedure as the first one in the main loop), ORDERS, and WPNALYSIS, print the reports, handle the players orders (the choices as to what to buy, scrap, or fund), and analyzes a weapon system respectively. A fourth option calls another routine named EXORDERS and breaks out of the players turn loop. EXORDERS executes the orders the player has made which signifies the end of the players turn and the beginning of the computers turn (see figure 5).

Player Function Menu

1. Reprint the reports.
2. Enter this years orders.
3. Perform weapon system analysis.
4. Finished, execute orders and proceed to next year.

Please enter the number of the option you wish to perform.
(or type H for help)

This is the main function menu. Your options are as follows:

- 1) Reprint the reports.
This option lets you view, once again, your current force structure, proposed R&D projects, and the intelligence reports.
- 2) Enter this years orders.
This is where you tell the computer which systems you want to buy and scrap, which R&D projects to fund or shelve, and what types of intelligence reports you want to purchase. These orders are not final and can be changed any number of times in a year. They become final only when option 4 is selected.
- 3) Perform weapon system analysis.
This option allows you to determine how valuable a weapon system is. The value is in terms of Utils per dollar. You will be asked to input all pertinent information about the system you wish to analyze (this information is obtained from the reports option).
- 4) Finished, execute orders and proceed to next year.
This option should be performed after you have finished any analysis and have entered your orders. It signifies the end of your turn.

€
press enter to continue

figure 5.

The computers turn is handled by the third procedure in the main loop and is called COMPTURN. (Despite the sequential nature of the program, the computer receives no advantage by going second as none of the information just entered by the human player is available to the expert system). COMPTURN is the computerized player and all heuristics governing the computers selections are contained in it. The flow of the routine is as follows: First the best offensive and defensive weapon systems of all the currently known systems are determined. Then, if the best systems are ones that need R&D funding, the appropriate costs are paid. Next the best operational offensive and defensive systems are determined and if there are any currently in the inventory, the appropriate operation costs are paid. If there are any excess systems they are scrapped but if there is any money left, new systems are bought. There was some initial concern about the time it would take the computer to take its turn, but as it turned out, this entire procedure takes no more than five seconds.

After the computer has selected its moves, the fourth procedure in the main loop, called ENDOFYEAR, is executed. It is in this routine that the results of the players and the computers moves is determined. (While the program runs the turn sequentially; as if the human player "moved" first, this routine actually handles them as if they were made simultaneously.) It is in this routine that the penalties

for unbalanced forces are applied. After the penalties are levied, the computer determines who is ahead at the end of the year and reports its findings. The last item this routine accomplishes is to determine if a war broke out this year and if one did, who the winner was (see figure 6).

```

      W      W      W      A      RRRRRR
      W      W      W      A  A      R  R
      W      W      W      A  A      RRRRRR
      W      W      W      A  A  A      R  R
      W      W      W      A      A      R  R
      W      W      W      A      A      R  R

```

Due to worldwide tensions, a war broke out this year. This brief war ended with you as the LOSER. Your budget, consequently, has been reduced by \$800.
press return to continue

figure 6.

The final procedure in the main loop is the already mentioned PRNTRPRT. The procedure prints the information used by the M.I.-based ICAI system to the appropriate file. Before it does this, it has to translate all numeric data to its ASCII representation.

After the routines in the loop have been executed eight times, the loop is complete and the final procedure, ENDOFGAME, is executed. The procedure prints the year by year until total along with anoting in which years a war occurred. It also calculates and reports who the ultimate winner of the game is. When this is done, the TEMPO war game is finished.

VI. The Intelligent Instructor

After completion of the game, there exists eight files of information. Gathered each game turn, these files contain data concerning the decisions made by the players in each of the game turns. This data is read in by a separate program, written in M.I, which evaluates the decisions made and displays information to the player on how well the game was played and where mistakes were possibly made. This is the intelligent computer aided instruction portion of the system. The reason an evaluation is not made during the course of the game, is that this system was modelled after the SOS version of the game, where the game is finished and then the section leader leads the students in a discussion of their performance, pointing out where mistakes might have been made. The ICAI system takes the place of this section leader.

The reason M.I was selected to write the ICAI system in was simply a matter of availability. The Air Force Institute of Technology currently has several copies of M.I which are used in a short course in artificial intelligence. M.I is not the only knowledge engineering tool available for the IBM PC and compatibles, nor is it the least expensive. There exists several different tools, including versions of OPS5, which vary in price from a few hundred dollars to a few

thousand dollars. The choice, as mentioned, was purely based on current availability. A list of other currently available knowledge engineering tools can be found in table 1.

Knowledge Engineering

Knowledge engineering systems, which M.1 is designed to help build, are designed around two major components; a knowledge base and an inference engine. The knowledge base contains the rules and facts necessary to accomplish the given task and the inference engine is the part of the system that uses these facts and rules to arrive at some decision. In the case of knowledge engineering tools such as M.1, the inference engine is part of the package and the job of the programmer is to obtain the necessary rules and form them into statements useable by the knowledge engineering tool.

The acquiring of the knowledge for the system is, therefore, the primary task of the programmer. A special name is given to these individuals, who are now known as knowledge engineers. The knowledge engineer interacts with an expert in the subject to obtain the information needed by the system. Once obtained, this information can then be formalized into the structure needed by the system. The obtaining of the knowledge from the expert is the most difficult portion of the knowledge engineer's task. This is due to the fact that experts generally express their reasoning in terms too broad for use in the system. The knowledge engineer must then, by observing the expert "in

The following systems run on IBM PCs

Company	Phone number	Product Name	Price	Written In	Maximum Rules
Artelligence Inc. 1402 Preston Road Dallas TX 75240	214/437-0361	OPS5+	\$3000.00	C	1500
California Intelligence 912 Powell Street San Francisco, CA 94108	415/391-4846	XSYS	\$1,000.00	IQ LISP	Systems can be linked
Digitalk, Inc. 5200 W. Century Blvd. Los Angeles, CA 90045	213/645-1082	Methods	\$250.00	Assembler & Basic	Systems can be linked
Dynamic Master Systems P.O. Box 566456 Atlanta, GA 30356	404/425-7715	TOPSI	\$75.00	Turbo Pascal	5,000 Systems can be linked
Expert Systems Int'l. 1150 First Ave. King of Prussia, PA 19406	215/337-2300	ES/P ADVISOR	\$1895.00	PROLOG	400, systems can be linked
EXSYS, Inc. P.O. Box 75158 Albuquerque, NM 87194	505/836-6676	EXSYS	\$295.00	C	5,000
General Research Inc. 7655 Old Springhouse Road McLean, VA 22102	703/893-5900	TIMM	\$9,500.00	Fortran 77	500
Human Edge Software Inc. 2445 Farber Place Palo Alto, CA 94303	415/493-1593	Expert Ease	\$695.00	UCSD Pascal	300, systems can be linked.
Level 5 Research Inc. 4980 S-Ala Melbourne Beach, FL 32951	305/729-9046	INSIGHT 1	\$95.00	Turbo Pascal	2,000
PPE Inc. P.O. Box 2027 Gathersburg, MD 20879	301/977-1489	Expert System	\$20.00	Basic	5,000
Radian 8501 MO-Pac Blvd Austin, TX 78766	512/454-4797	Rule Master	\$5000.00	C	200, systems can be linked
Software A & E Inc. 1500 Wilson Blvd. Arlington, VA 22209	703/276-7910	KES	\$4,000	IQ LISP	Systems can be linked

table 1. (Schwartz, 1985:1263)

action", and by asking questions and interviewing the expert, determine what basic rules were used in the larger reasoning process. As can be inferred from this discussion, the knowledge engineer and the expert are two separate individuals, indeed, most literature encourages that this be the case. In some cases, however, it is advantageous or necessary that the expert and the knowledge engineer be the same individual. Generally this is when the rules generated are "cut and dry" (such as devising an expert system to assist in the filling out of travel vouchers) or when there is a language available to assist the domain expert in the development of the knowledge base (such as M.I) or when the most available expert is the knowledge engineer. Such is the case for the TEMPO system, where the best expert on the system was also the individual programming the system. The reason this still worked was that the knowledge needed for the ICAI system concerned the types of mistakes that students make in the course of the game, and why they are mistakes. The knowledge obtained for this portion of the program came from repeated observations of individuals playing the game. The task of the knowledge engineer was then to simply determine why the decision made was a mistake and how and why the student made it. In TEMPO there are a few basic concepts that are being taught. Mistakes generally involve a misunderstanding of these concepts.

Concepts in TEMPO and their M.1 Rules

As was explained earlier, TEMPO is a force planning exercise. The game was designed to teach the importance of comparing the value of weapon systems in order to obtain the "most bang for the buck". Additionally, the game brings out other concepts such as the point of diminishing returns and the idea of a balanced force posture. When any one of these concepts is not fully understood by the student, mistakes are made which lead to a less than optimum force structure. The PASCAL program makes no determination as to the worth of the human players moves, it simply carries out the orders given to it and at the end of each turn records information by writing it to one of eight files representing each of the different game turns. These files, written in ASCII, are then read in after the game by the M.1 ICAI system and are used to evaluate the comprehension of the simple game concepts by the student.

One of the game concepts, referred to as the "more bang for the buck" concept, is demonstrated in figure 7. In this figure two weapon systems are compared, ABM-1 and ABM-2. At first glance they seem to be roughly equal since both offer twice the number of utils as the yearly operation cost (a good rule of thumb when choosing the best weapon system). In fact since they are so close, the player might be tempted to go with the system that is currently available and

actually costs less per unit. As can be seen this is not the correct choice since the more expensive system actually is the better of the two. This type of mistake is the most common one made in TEMPO, especially if the player is trying to hurry. Since this error is so common it is one of the ones that is checked for in the ICAI portion of the program (see figure 8).

```

Enter the name of the weapon system. ABM-1
Enter the number of years to run the analysis. 10
Enter the number of years of R&D left. 0
Enter the system acquisition cost. 30
Enter the system yearly operation cost. 10
Enter the system util value. 20
Enter the 1st year R&D cost (enter a 0 if none). 0
Enter the 2nd year R&D cost. 0
Enter the 3rd year R&D cost. 0
Enter the purchase limit. 30

```

The util/s percentage for the ABM-1 is 146 %

press return to continue

```

Enter the name of the weapon system. ABM-2
Enter the number of years to run the analysis. 10
Enter the number of years of R&D left. 2
Enter the system acquisition cost. 80
Enter the system yearly operation cost. 50
Enter the system util value. 110
Enter the 1st year R&D cost (enter a 0 if none). 560
Enter the 2nd year R&D cost. 1090
Enter the 3rd year R&D cost. 0
Enter the purchase limit. 20

```

The util/s percentage for the ABM-2 is 186 %

press return to continue

figure 7.

```

if do(leadcache year,X) and
  bestbmb = B and
  bestmsl = M and
  B > M and
  display('The best offensive system this',
    'year was a bomber.',n1)
then the answer for cycle X =
  'the best offensive system is a bomber'.

if bestbmb = B and
  bestmsl = M and
  M > B and
  display('The best offensive system this',
    'year was a missile.',n1)
then the answer for cycle X =
  'the best offensive system is a missile'.

```

figure 8.

Another very common mistake is to overspend the budget. This is a very crucial error because it causes the players next year's budget to be reduced (this is so crucial that the computer expert system is designed so that it never goes over its budget). Because this is a common error, not only is this checked for in the ICAI system (figure 9) but an example is also included in the event that the system determines that this is a re-occurring problem (figure 10). This is the type of thing that separates an ICAI system from a CAI, the analysis of the students problems and the tailoring of the session to him.

One final example of a game concept checked for in the ICAI portion of the program is the concept of not overdefending. If so much money is spent on a defensive system type that it completely overcompensates for an opponents offensive system, then money is being wasted. The ideal situation is to have exactly the number of defensive utils as your opponent has offensive. Since this is very rare, the next best thing is to have slightly fewer defensive utils than your opponents offensive. The M.1 code checking for this is shown in figure 11.

```

if overspent = OV and
  OV = 1 and
  display(['You overspent your budget this year. ',
    'This is a critical',nl,'mistake as it ',
    'will result in a reduced budget the next',nl,
    'year as well as causing you to lose ',
    'some of your systems.',nl,
    'The systems lost may not be ones you ',
    'would have wished to',nl,'lose, they',
    'are picked by the computer at random.',nl,nl])
  then cycle X overspent check is complete.

if overspent = OV and
  OV = 2 and
  display(['You overspent your budget again. ',
    'This is a critical',nl,
    'error and it is extremely important ',
    'that you ensure that',nl,
    'you remain within budget each year.',nl,nl])
  then cycle X overspent check is complete.

```

figure 9.

```

if overspent = OV and OV > 4 and
  display(['You have once again overspent ',
    'your budget.',nl,nl])
  then cycle X overspent check is complete.

if ovready = yes and
  display(['If you have the following, what ',
    'is your budget cost?',nl,nl,
    'B1 --> Acq cost = 100, Ops cost = 50',nl,
    'F1 --> Acq cost = 50, Ops cost = 25',nl,nl,
    'Your current inventory contains ',
    '5 B1s and 2 F1s.',nl,nl,
    'You wish to purchase an additional',
    '5 B1s and 4 F1s.',nl,nl,
    'You want an intelligence report ',
    'on offensive R&D.',nl,nl,
    'You wish to fund R&D on a B2, cost 200 ',
    'for 1st year.',nl,nl]) and
    ovcost = 1300 and display(['Correct',nl])
  then ovspent is complete.

```

figure 10.

```

if overdefend = OD and
  OD = 0
then cycle X overdefend check is complete.

if overdefend = OD and
  OD = 1 and
  display([You overdefended this year. ',
    'This means that you spent',nl,
    'more money on defense than you ',
    'needed to. Money spent',nl,
    'in this manner is referred to as ',
    '"Futile utils" since you',nl,
    'receive no benefits from any extra ',
    'defensive utils (they',nl,
    'are wasted.',nl,nl])
then cycle X overdefend check is complete.

```

figure 11.

VII. Conclusion and Analysis

This thesis has resulted in the implementation of a war game which meets four characteristics: (1) that it be implemented on a machine chosen so as to make it available to as many officers as possible, (2) that it be written in a manner that facilitates changes in the system (make it modifiable), (3) that it be implemented with a computerized player expert system, and (4) that it contain an intelligent instructor to aid in the learning process. The game has been tested a number of times and is an extremely competent opponent, especially against first time players of the game (it is undefeated against them). This is especially important since it will probably be played only once or twice by most individuals. Against experienced opponents the expert system doesn't fare as well, winning just over fifty percent of the time. This, however, is more a function of this game, or any game for that matter, since if two experienced opponents were to play each would be expected to win half the time. The outcome of the game is only one way to measure the success of the implementation of the game. A second measure is the impact it can have on professional military education seminar courses. As a result of this thesis, it has now been shown that it is possible to implement useful war games, on widely available microcomputers, and have them stand alone using the expert

system computerized player concept. Other war games, now only available at the residence courses, can also be implemented thus making seminar courses that much more meaningful. Another measure of success is the modifiability of the program which was tested as enhancements were made to the original version of the game.

Enhancements to TEMPO

Conversations with War Gaming Center personnel from Maxwell AFB in Alabama, led to a list of several ways to enhance the original version of TEMPO. Among these enhancements was the idea of providing the players with the opportunity to deceive their opponent by funding non-operational systems that only showed up in the intelligence reports. This idea was added to the TEMPO program (see figure 12) and took only about two hours to complete. While it would take someone who does not know the code significantly longer to accomplish, it does give a rough idea of how long it should take.

Another enhancement made to the game was to provide the participants the ability to declare war at a time of their own choosing. Both this and the previous enhancement are implemented so that they either can be used or ignored at the discretion of the individual setting up the weapon system data file (both options have flags set in this file, see figure 13). While this second enhancement took longer to

implement, it was much more involved and required the addition of several new rules.

Analysis of M.1 and the Intelligent Instructor

The intelligent instructor is an extremely important portion of the system. If no instructor is available, it takes significantly longer for the player to grasp the concepts contained in the program. The intelligent instructor portion of this system suffers from one main problem; players are continually finding new ways to make mistakes and to lose the game. This means that the instructor portion of the system is continually evolving. While it was decided that M.1 was not capable of handling the war game itself, it is suited to the evolving instructor. M.1 is awkward to use when large amounts of data are to be analyzed (number crunching) and it doesn't provide for easy ways to loop or nest loops. It is, however, very easy to add rules to once they are structured in an IF . . THEN . . format.

Recommendations and Areas for Further Study

AI can help solve several problems that exist in current war games. First and foremost is the idea of the "unknown unknowns". There are actually two issues here, how to create the unknowns to be used in the games and how to handle them

For the B-1 weapon system, you currently have 40 in your inventory.
 How many do you wish to BUY? 0
 How many do you wish to SCRAP? 0
 How many do you wish to purchase for DECEPTION? 10

Line 829 Col 1 Insert Indent A:ETEMPO.PAS

```
modlim:  write('How many do you wish to be MODIFCATIONS? ');
         readintnum(modfy[xx]);
         if modfy[xx] > systvals[psys[xx,1],9] then
           begin
             write('Your modify order exceeds the limit for this ');
             writeln('system of ',systvals[psys[xx,1],9]);
             goto modlim;
           end;
         end;

if deception = 1 then
  begin
    writeln;
    write('How many do you wish to purchase for DECEPTION? ');
    readintnum(tmparray[xx,3]);
    if tmparray[xx,3] < 0 then tmparray[xx,3] := 0;
  end;
```

figure 12.

```
write ('    Defensive R & D ? ');
readln(templet);
if ((copy(templet,1,1) = 'Y') or (copy(templet,1,1) = 'y')) then
  tintel[4] := 1;
end;
```

```
if warflag = 1 then
  begin
    tmpdeclwar := 0;
    clrscr;
    write('Do you want to declare war? (Y/N) ');
    readln(templet);
    if ((copy(templet,1,1) = 'Y') or (copy(templet,1,1) = 'y')) then
      tmpdeclwar := 1;
  end;
```

figure 13.

once they have been introduced. The first of these can be solved by having a programmer input the ideas from some expert. The problem still remains, however, that the unknowns are limited to some humans experience or creativity. It would be much more interesting, and also much harder, to devise an AI system which could develop these unknowns. This entails creating a system that can be creative and reverts back to one of the basic research areas in AI. The second issue involved in the use of unknowns, how to handle them once they are introduced, is a much more obtainable goal and is one that is more useful to war gaming. Currently, unknowns are unpopular because they are considered "unfair" by the participants (but then, war is unfair) and to try to consider all of the possible factors that could affect a move results in an information overload problem. Since computers are good at reducing the problems created by information overload, there is a real use for them to do so. This then is the next area for study; to implement a category two game in which the capability to handle unknowns is created.

Finally, it is recommended that the Air Force take a look at its residence courses in order to identify war games and simulations that could be implemented on microcomputers to make seminar courses more valuable. By doing so, officers could receive professional military education in seminar courses that more closely reflects the current curriculum at the residence schools. This same idea, of using computers to

help teach the seminar course, could be applied to other areas besides gaming and simulation as well.

The Future of War Gaming

This thesis has shown that it is possible to implement a useful war game on a microcomputer thus making it available to a larger segment of the officer corps than had access to it before. In the future, it is envisioned that entire courses could revolve around a series of war games that the student pulls off of a shelf and plays. Each game would be designed to teach a different concept and an ICAI system could help facilitate the learning process. As the computer expert systems get better at handling the unknowns, which are the last bastion of human capability, more and more of the battlefield decision processes can be assumed by computers. At the very least, the proliferation of computer war games would increase the war fighting capability of our military officers.

Bibliography

- Barr, Avron and Edward A. Feigenbaum. The Handbook of Artificial Intelligence, Vol 2. Stanford: Heuristech Press, 1982.
- Clancey, William J. "The Epistemology of a Rule-Based Expert System: A Framework for Explanation." Stanford University, November 1981 (AD-A112 672).
- Clausewitz, Carl von. On War, edited and translated by Michael Howard and Peter Paret. Princeton: Princeton University Press, 1976.
- Davis, Paul K. Rand's Experience in Applying Artificial Intelligence Techniques to Strategic-Level Military-Political War Gaming. Santa Monica: The Rand Paper Series, 1984 (AD-A147 272).
- Furlong, Lt Gen Raymond B. (RET). "Clausewitz and Modern War Gaming", Air University Review, July-August 1984. Washington DC, GPO.
- Klahr, Philip and others. SWIRL: Simulating Warfare in the ROSS Language. Santa Monica, The Rand Corporation: September 1982 (Rand Note N-1885-AF).
- Murawski, Capt Robert J. Jr., USMC. First to fight: A Battle Simulation at the Squad and Fireteam Level. MS thesis, LSSR 65-81. School of Systems and Logistics, Air Force Insititute of Technology (AU), WPAFB OH, September 1981 (AD-A110 320).
- Owens, Lt Christopher A., USN. TEMPOA: An Interactive Simulation for the Apple III Microcomputer. MS thesis, Naval Postgraduate School, Monterey CA, October 1982 (AD-A124 656).
- Schwartz, Tom J. "Artificial Intelligence in the Personal Computer Environment," Proceedings of the Ninth International Joint Conference on Artificial Intelligence. 1261-1264. Los Angeles, 1985.
- Squadron Officers School, Area Three, Book 1. Class 84-B. Maxwell AFB: Air University, August 1983.

Tzu, Sun Wu. The Art of War, translated by Lionel Giles.
Harrisburg: The Military Service Publishing Company.

Waterman, Donald A. A Guide to Expert Systems. Reading:
Addison-Wesley Publishing Company, 1986.

APPENDIX A.
(Programmers Manual)

TEMPO (Main routine)

Purpose: This is the main driver for the game.

Parameters Passed: none.

Calling Routines: none.

Routines Called: ENDOFGAME, ENDOFYEAR, INIT, PLAYERTURN,
PRNTRPRT, REPORT.

Description: The main routine starts off by calling INIT to initialize the arguments used in this program. It then sits in a FOR loop for eight repetitions. Each repetition consists of calculations to determine each players budget and calls to REPORT, PLAYERTURN, ENDOFYEAR, and PRNTRPRT. After the loop is complete, ENDOFGAME is called. The budget calculated is the same for both the computer and the human player unless the player exceeds the budget or a war occurs. In any case, the amount that each budget is increased by is the same for both sides. The budget is never allowed to go below 1500, this insures that the player will always have some money to work with.

Global Variables:

budget -- the current years computers budget.
budgetny -- next years projected computers budget.
chancewar -- the chance of a war occuring this year.
bestbmb -- best bomber (a pointer).
bestdef -- best defensive system (a pointer).
bestmsl -- best missile (a pointer).
cabmtot, cabmcnt -- computer abm total/count.
cbmbtot, cbmbcnt -- computer bomber total/count.
cfttot, cftcnt -- computer fighter total/count.
cmsltpt, cmslcnt -- computer missile total/count.
csys -- an array which contains the information about the
weapon system the computer owns or is developing.
There are a maximum of 20 systems and the info
contained on each is as follows:
 [xx,1] --> pointer to systvals for which system
 this is.
 [xx,2] --> the number of this weapon system
 currently in the inventory.

[xx,3] --> a flag which tells if the system
 is operational (=0), currently having
 R&D funded (=1), or shelved(-x with x
 being the year shelved in).

csysutil -- (reserved for future development).

intel -- an array of four flags, one for each type of intel
 report, which represent the type of reports funded
 for this year (used in REPORT procedure to tell
 what to print).

lineprt -- a variable to be filled to be used to print to
 a file.

modify -- array (one location for each wpn system) used to help
 with the modification of weapon systems.

overdefend -- flag to set if player overdefended this year.

overspent -- flag to set if player overspent budget this year.

overspentcnt -- number of times player overspent budget.

pbudget -- players budget this year.

pbudgetleft -- used in calculations to determine the amount
 of the players budget left.

pbudgetny -- projected next years players budget.

pabmtot, pabmcnt -- player abm total/count.

pbmbtot, pbmbcnt -- player bomber total/count.

pfttot, pftcnt -- player fighter total/count.

pmsltot, pmslcnt -- player missile total/count.

psys -- players systems, same layout as csys.

rndarray -- array used in ORDERS procedure to take players
 orders concerning R&D. There is one entry
 for each weapon system and each has three parts:

- [xx,1] --> equal to 1 if R&D to be funded.
- [xx,2] --> if last year for R&D and it was
 funded, then player can buy
 systems, this contains the number
 of systems player wants to buy.
- [xx,3] --> number of systems player wants to
 be modifications (if applicable).

sysdata -- for reading in the weapon system data.

sysrprt -- for writing out the ICAI information.

systemp -- temporary variable of record type fileline.

systname -- the array containing the weapons system names.
 there is a 1-to-1 correspondance with this and
 the systvals and systutils arrays.

systutil -- the array which receives the calculated util/\$
 value for the systems, there is a 1-to-1
 correspondance between this and systvals and
 systname.

systvals -- the array which contains all of the values
 associated with the weapon systems. There are
 11 values associated with each as follows:

- [xx,1] --> acquisition cost
- [xx,2] --> year1 operations cost
- [xx,3] --> system yearly util value
- [xx,4] --> year to be offered for R&D
- [xx,5] --> number of years of R&D to do.

[xx,6] --> first year R&D cost.
 [xx,7] --> second year R&D cost.
 [xx,8] --> third year R&D cost.
 [xx,9] --> the yearly purchase limit.
 [xx,10] --> the modification cost.
 [xx,11] --> flag to say if mod available.
 has a 1-1 relation with systutil & systname.
 tempbudget -- temporary value to help determine budget.
 tintel -- temporary array used to hold intel orders until
 they are made final by EXORDERS.
 tmparay -- temporary array used to hold buy and scrap orders
 until they become final by EXORDERS.
 utilaray -- array to hold computers and players yearly util
 totals (used to determine game winner).
 utlsperdol -- figure to hold analysis figure.
 warcount -- count of number of wars/2
 x -- temporary looping variable.
 year -- the game turn (the year).

COMPTURN

Purpose: this procedure handles the computers turn.

Parameters Passed: none

Calling Routine: PLAYERTURN (done after player has finished but before players orders are executed).

Routines Called: NYRANLYS

Description: The procedure starts off by determining the utils per dollar value for all available weapon systems up to that point. With these values determined, it uses them to check its operational systems and then the proposed R&D systems to determine the best bomber, missile and defensive system. If the best system is an R&D one then funds are expended, otherwise R&D is stopped for any current R&D projects. After any R&D costs are paid the procedure determines the best operational systems for each type (both defensive types included), ops cost paid for any currently in the inventory, 1500+ taken out of budget to help assure some sort of balance offensive and defensive wise, and any other systems scrapped. Then with the remaining money, buy offensive systems maintaining the 2-to-1 ratio of best to second best. Then any remaining money is determined, the 1500+ added to it and the same thing done for the defensive systems. At the end the procedure checks to make sure that it can't buy even just one more system, thus always trying to spend as much of the budget as possible.

Routine Specific Variables Used:

bestbmbflag -- flags used to mark whether the best systems need R&D funding or not.
bestdefflag "
bestmslflag "
bstabminv -- temporary location to contain the current inventory of the best operational weapon systems.
bstbmbinv "
bstftrinv "
bstmslinv "
bstopabm -- pointers to the best operational systems
bstopbmb "
bstopftr "
bstopmsl "
budgetleft -- used to keep track of the budget left.
dptr -- temporary pointer
numbuy -- number of weapon systems to buy.
ofopscost -- offensive operations cost.
optr -- temporary pointer
templet -- temporary letter.
tempval -- temporary integer value.
xx,yy -- temporary integers for looping purposes.

DISPLAYMENU

Purpose: To display the primary option menu:

Parameters Passed: none.

Calling Routine: PLAYERTURN.

Routines Called: none.

Description: The procedure does nothing more than to clear the screen and then display the Player Function Menu.

Routine Specific Variables Used: none.

ENDOFGAME

Purpose: This procedure is called at the end of the game to display the yearly totals and declare an overall winner for the game.

Parameters Passed: none.

Calling Routine: TEMPO.

Routines Called: none.

Description: The procedure starts by printing the yearly util totals for both the computer and the human player and also reports on which years a war occurred in. It also adds up all util figures for both sides and, based on these figures, declares an overall winner for the game.

Routine Specific Variables Used:

comptot --	variable used to	accumulate computers
	total util values.	
playtot --	variable used to	accumulate players
	total util values.	

ENDOFYEAR

Purpose: This procedure is executed at the end of each year (game turn). It calculates, stores, and reports on the yearly winner as well as the likelihood of war.

Parameter Passed: none

Calling Routine: TEMPO

Routines Called: none

Description: The routine starts off by determining the winner (who is ahead in total offensive utils) for the year. This is calculated by summing each forces offensive and defensive utils by weapon system (i.e. adding all utils from all bombers up and putting it in one variable and all utils from all missiles in another). The forces are then checked to determine if a balance is maintained (a 2-to-1 ratio at least) or else a penalty is applied. After any penalties have been applied the net offensive util value (your offensive minus your opponents defensive utils) is applied--all values less than zero rounded up to zero--and the yearly winner determined. The winner is simply the individual with more net offensive utils. Once this is determined it is reported to the player. The procedure then possibly changes r&d values for systems that have at least 2 years remaining. This is to reflect the inhering uncertainty in such figures as well as to model things such as cost overruns. Next, the procedure determines if a war occurred, and if one did, the result. The last thing the routine does is to calculate the chance of war occurring the next year. NOTE: one programming problem occurred since TURBO PASCAL only allowed values up to 32700 in their integer variables. A check is made to determine we don't go over this (which shows up as a negative total in cofftot and pofftot).

Routine specific Variables Used:

cofftot -- 'computer offensive total'; used to hold the value of the computers total offensive utils after any penalties have been applied.

pofftot -- 'players offensive total'; same purpose as cofftot except for human player.

ptr -- a temporary pointer into the weapon system arrays, used only for ease of typing/understanding.

templet -- a temporary character variable used for character input ('temporary letter').

t1,t2 -- temporary variables used in calculations of offensive util totals.

xx -- a temporary integer valued used in loops.

EXORDERS

Purpose: To execute the orders entered by the player.

Parameters Passed: none.

Calling Routine: PLAYERTURN.

Routines Called: none.

Description: This procedure is the last one called by the player each game turn and it signifies the end of the turn. Actually, it is not called until after the computer selects its moves so that all of its decisions are based on the present player configuration and not the players orders for the next year. It first takes care of all scraps, then operation costs for current systems, then buy orders. It then takes care of ongoing R&D projects, and then new R&D projects. Then modification orders and intel reports are handled. Finally, it is checked to see that the player did not go over budget. If so, then the computer will start throwing out weapon systems until the budget is reached, it will start with the newest systems first. This is done as a penalty to the player to force compliance with the budget.

Routine Specific Variables Used:

Tempval: temporary value

templet -- temporary letter

xx,yy -- temporary looping variables.

HELPSCR

Purpose: To display the appropriate helpscreen.

Parameters Passed:

helpscn -- an integer indicating which help screen is to be displayed.

Calling Routine: REPORT

Routine Called: none

Description: This procedure first opens the appropriate file based on the screen number passed into the procedure. It then reads in and prints the file line by line until a '@' is reached which signifies the end of the file. It then waits for the player to hit a return before returning to the calling routine.

Routine Specific Variables Used:

syshelp -- file variable for reports.
helpln -- variable to read in report.
templet -- temporary letter.

INIT

Purpose: This procedure initializes all values for the game.

Parameters Passed: none.

Calling Routine: TEMPO.

Routines Called: none.

Description: The procedure starts by initializing the budget figure, the chance of war variable, and a few of the arrays used in the game. It then reads in the system variables from the file WPNSYS.DAT. A few of the variables are randomly modified so that the game is different each time it is played. After all weapon systems are read in, the procedure gives each side its initial set of forces (which is the same for both sides) and zeroes out the rest of the array.

Routine Specific Variables Used:

x -- used for a looping variable.

NYRANLYS

Purpose: To perform multi-year analysis of a weapon system.

Parameters Passed:

Nyr -- number years to run analysis.
Avlin -- system available in number of years.
Acqcst -- acquisition cost.
Opkst -- operations cost.
Utls -- util value.
Rd1 -- first year R&D cost.
Rd2 -- second year R&D cost.
Rd3 -- third year R&D cost.
purlim -- purchase limit.

Calling Routines: COMPTURN, WPNANALYS

Routines Called: none.

Description: This procedure performs an analysis of a weapon system and returns a value representing its utils per dollar value. The calculation is straightforward, dividing all utils obtained for the given period by the total cost during that time.

Routine Specific Variables:

Acqbugt -- amount of budget available for acquisitions.
Opscost -- total opscost for year.
Sysbght -- integer value of Sysbgt.
Sysbgt -- number of systems bought this year.
Totacqcst -- total acquisition cost (all systems).
Totlutils -- total util values (all systems all years).
Totrd -- total r&d costs.
Totopscst -- total all operations costs.
wpnsbght -- total number of systems bought all years.
X -- temporary looping variable.
Yrsutls -- this years utils.

ORDERS

Purpose: To take the orders of the player.

Parameters Passed: none.

Calling Routine: PLAYERTURN

Routines Called: READINTNUM

Description: This procedure starts by asking, for each weapon system currently in the players inventory (or that has all R&D costs paid for), how many should be bought, scrapped, and if applicable, modified. It then displays a synopsis and asks if it is correct, if it is it goes on, if it isn't, the player gets the chance to go back and correct anything. Next, for each proposed R&D project the player is asked if funding is desired, then if funding is to be continued on the systems with R&D in progress, then if R&D should be resumed on any shelved systems. As before, the player is asked to check to make sure it is correct, if not another chance is given. Finally, the player is asked if any intelligence reports are desired. If some are, each type is asked for in turn. One final NOTE ***** this routine does not execute the orders and can be performed as many times as desired before the turn is over. The turn is only over when option 4 from the primary function menu is selected.

Routine Specific Variables Used:

answer -- temporary variable to receive players typed answer to a question.
flag -- used to help determine what to display.
ptr -- used to help relieve some typing.
templet -- temporary letter, (for answers to y/n).
xx -- temporary looping variable.

PLAYERTURN

Purpose: To handle the players turn.

Parameters Passed: none.

Calling Routine: TEMPO

Routines Called: COMPTURN, DISPLAYMENU, EXORDERS, ORDERS,
REPORT, WPNANALYS

Description: This procedure consists of a while loop that is executed until the player enters a '4' signifying the end of the game turn. If any response is entered except one of the requested ones, an error message is displayed. Otherwise, the appropriate routine is called for the response entered.

Routine Specific Variables Used:

answer -- the response entered by the player

PRNTRPRT

Purpose: To print the values to be used by the ICAI portion of the overall system.

Parameters Passed: none.

Calling Routine: TEMPO

Routines Called: none.

Description: This procedure simply prints the values that will be used by the ICAI system. It writes to 8 separate files, one for each year. The files are called 'YEAR.x' with x being the year. The only different thing about the routine is that it has to translate all data to its ascii form so that M.I can read it (thats the only format that the knowledge engineering tool created by Teknowledge can read).

Routine Specific Variables Used:

tempstr, tempstrg -- temporary strings used in the conversion processes.

READINTNUM

Purpose: This procedure is used to insure that an integer is read in, any other input is flagged as an error and not allowed (this is not left up to the system since it would cause the game to abnormally terminate if a mistake were made).

Parameters Passed:

num -- the integer entered by the player.

Calling Routines: ORDERS, WPNANALYS

Routines Called: none.

Description: This procedure simply reads in a character input and converts it to its integer equivalent using the PASCAL function VAL. If it is not a correct character it prints an error message and waits for the player to try again. It will remain in this procedure until a correct input has been made. Once an integer is obtained, it is returned in the parameter NUM.

Routine Specific Variables Used:

innum -- the character input by the player to be converted to an integer.

result -- will contain the result of the conversion process (if not equal to zero, it means an error occurred, if it contains a zero then no error occurred and the number is an integer).

REPORT

Purpose: This procedure prints the reports that lets the player know what the status of his forces are, any R&D projects projected or pending, and any intelligence reports that were paid for.

Parameters Passed: none.

Calling Routine: PLAYERTURN, TEMPO

Routines Called: HELPSCR.

Description: The procedure first displays the screen which contains the players current force structure. It then waits for the player to either request the help screen for this display (by entering an 'h' or an 'H'; the procedure calls the help screen function HELPSCR) or to go on. The next screen displayed is the one showing any proposed, current or scrapped R&D projects. If there were none then this screen is skipped. There is a help screen for this display also. The final report displayed is the intelligence screen which also has an associated help screen.

Routine Specific Variables Used:

deftop -- defensive total operations cost
flag -- flag used to help determine when to print headers.
inven -- inventory count for individual wpn system.
offtop -- offensive total operations cost.
ptr -- temporary pointer to make it easier to type.
temparay -- temp array used to hold R&D values
templet -- temporary letter.
tempval,tempval2 -- temporary values for calculations to determine the R&D costs for the current years.
topcost -- total operations cost.
totalop -- total operations for both off and def.
tutils -- total util figure for a weapon system
xx -- temporary looping variable.

WPNANALYS

Purpose: The interface for the player to perform a multi-year weapon system analysis.

Parameters Passed: none.

Calling Routine: PLAYERTURN

Routines Called: READINTNUM, NYRANLYS

Description: This procedure simply asks all questions needed to obtain the information that NYRANLYS needs to perform the multi-year weapon system analysis. After it receives the answer it displays it for the player.

Routine Specific Variables Used:

- acost -- acquisition cost.
- nyrs -- number years to run analysis
- ocost -- operations cost.
- prlm -- purchase limit.
- rndlft -- number of years of R&D left.
- rnd1,2,3 -- the three years r&d costs.
- tempval -- temporary value.
- utlvl -- util value.
- wpnname -- name of weapon system.

Appendix B
(TEMPO Code)

```
program Tempo;

type
  name = string[6];
  linechar = string[20];
  filechar = record
    linechr : linechar;
  end;
  fileline = record
    sysname : name;
    acqcost,
    opscost,
    utilval,
    yrforrd,
    availin,
    rndyr1,
    rndyr2,
    rndyr3,
    purchlm,
    modcost,
    modflag : integer;
  end;
  letter = string[1];
  full_line = string[79];

var
  budget : integer;
  budgetny : integer;
  chancewar : real;
  cbmbtot, cmsltot, cftrtot, cabmtot : integer;
  pbmbtot, pmsltot, pftrtot, pabmtot : integer;
  cbmbcnt, cmslcnt, cftrcnt, cabmcnt : integer;
  pbmbcnt, pmslcnt, pftrcnt, pabmcnt : integer;
  bestbmb, bestdef, bestmsl : integer;

  csysutil : array [1..10] of real;
  csys : array [1..20, 1..3] of integer;
  intel : array [1..4] of integer;

  lineprt : filechar;
```

```

modify : array [1..20] of integer; {array used for modify
                                     systems}

overdefend : integer; {flag set if player overdefends ie.
                       excess defense}
overspent : integer; {flag set if player overspent budget
                      this yr}
overspentcnt : integer; {cntr for # of times overspent}
pbudget : integer;
pbudgetleft : integer;
pbudgetny : integer;

psys : array [1..20, 1..3] of integer;
rndaray : array [1..20, 1..3] of integer; {array used for
                                             r&d funding }

sysdata: file of fileline;
sysrprt: file of filechar;
systemp: fileline;

sysname : array[1..20] of name;
systutil : array[1..20] of real;
systvals : array[1..20, 1..11] of integer;

terpbudget : integer;

tintel : array[1..4] of integer;
tmparay : array[1..20,1..2] of integer; { array used for
                                           buy and scrap}
utilaray: array[1..10,1..2] of integer; { array for yrly
                                           util totals }

utlsperdol : real;
warcount : real;
x : integer;
year : integer;

```

```

procedure init;
var
  x : integer;

begin
  randomize;
  budget := 9000 + (Trunc((1000.0 * Random)/10.0) * 10);
  pbudget := budget;
  chancewar := 0.15;
  overspentcnt := 0;

  for x := 1 to 4 do
    begin
      intel[x] := 0;
      tintel[x] := 0;
    end;

  for x := 1 to 20 do
    begin
      rndarray[x,1] := 0; rndarray[x,2] := 0;
      tmparray[x,1] := 0; tmparray[x,2] := 0;
    end;

  assign(sysdata, 'WPNSYS.DAT');
  reset(sysdata);

  { ***** read in the systems to be used, ***** }
  { from off file }

  with systemp do
    for x := 1 to 15 do
      begin
        read(sysdata, systemp);
        sysname[x] := sysname;
        acqcost := Trunc((acqcost/1.7 +
          (acqcost * Random)) / 10.0) * 10;
        systvals[x,1] := acqcost;
        opscost := Trunc((opscost/1.7 + (opscost * Random))
          / 10.0) * 10;
        if opscost > acqcost then opscost := acqcost - 10;
        systvals[x,2] := opscost;
        utilval := Trunc((utilval/2 + (utilval * Random))
          / 10.0) * 10;
        systvals[x,3] := utilval;
        systvals[x,4] := yrforrd;
        systvals[x,5] := availin;
        rndyr1 := Trunc((rndyr1/1.7 + (rndyr1 * Random))
          / 10.0) * 10;
        systvals[x,6] := rndyr1;
        rndyr2 := Trunc((rndyr2/1.7 + (rndyr2 * Random))
          / 10.0) * 10;
        systvals[x,7] := rndyr2;
      end;
    end;
  end;

```

```

        rndyr3 := Trunc((rndyr3/1.7 + (rndyr3 * Random))
                        / 10.0) * 10;
        systvals[x,8] := rndyr3;
        systvals[x,9] := purchlm;
        modcost := Trunc((modcost/1.7 + (modcost * Random))
                        / 10.0) * 10;
        systvals[x,10] := modcost;
        systvals[x,11] := modflag;
        systutil[x] := 0.0;
    end;
    close(sysdata);

{ ***** give each side the initial
      systems in inventory ***** }

    for x := 1 to 4 do
        begin
            csys[x,1] := x;
            psys[x,1] := x;
        end;

{ ***** initialize inventories ***** }
{ also initialize R&D flag }

        csys[1,2] := 40;    psys[1,2] := 40;    csys[1,3] := 0;
        psys[1,3] := 0;
        csys[2,2] := 60;    psys[2,2] := 60;    csys[2,3] := 0;
        psys[2,3] := 0;
        csys[3,2] := 20;    psys[3,2] := 20;    csys[3,3] := 0;
        psys[3,3] := 0;
        csys[4,2] := 100;    psys[4,2] := 100;    csys[4,3] := 0;
        psys[4,3] := 0;

{ ***** initialize rest of the array ---
      zero it out *****}

        for x:=5 to 20 do
            begin
                csys[x,1] := 0;    csys[x,2] := 0;    csys[x,3] := 0;
                psys[x,1] := 0;    psys[x,2] := 0;    psys[x,3] := 0;
            end;

        randomize;

    end;

```

```

procedure readintnum(var num: integer);

  { *** this procedure simply insures that
    a number is input ***** }

  label
    badnum ;
  var
    innum : name;
    result : integer;

  begin
badnum: readln(innum);
    val(innum,num,result);
    if result <> 0 then
      begin
        write('ERROR** input should be an integer number. ');
        goto badnum;
      end;
    end;
  end;

```

```

procedure helpscr(var helpscn: integer);

  { *** this procedure handles the help screens *** }

label
  agn, done;

const
  blank_line='
';

var
  syshelp: file of full_line;
  helpln: full_line;
  templet: letter;

begin
  case helpscn of
    1: assign(syshelp, 'REPORT1.HLP');
    2: assign(syshelp, 'REPORT2.HLP');
    3: assign(syshelp, 'REPORT3.HLP');
    4: assign(syshelp, 'REPORT4.HLP');
  end;
  reset(syshelp);
  clrscr;
agn:  helpln := blank_line;
      read(syshelp, helpln);
      writeln(helpln);
      if helpln = '@' then goto done;
      goto agn;
done: write('press enter to continue');
      readln(templet);
      close(syshelp);
end;

```

```

procedure report;

label
  help1, help2, help3;
var
  deftop, flag, inven, offtop, ptr, templet,
  tempval, tempval2, topcost, totalop, totalut, tutils,
  xx : integer;
  temparay : array [1..5] of integer;

begin
  help1: clrscr;

  writeln('Start of Year ',year);
  writeln('Your Budget is $ ',pbudget);
  writeln('The estimated budget for next year is $ ',
    pbudgetny);
  writeln('
                                PER UNIT
TOTAL');
  writeln('
----- PURCH MODI');
  writeln('SYSTEM  INVEN  AQCCOST  OPCOST  UTILS
OPCCOST  UTILS  LIMIT  COST');
  totalop := 0; deftop := 0; offtop := 0;
  for xx := 1 to 20 do
    if psys[xx,3] = 0 then
      begin
        templet := copy(systname[psys[xx,1]],1,1);
        if (templet = 'B') or (templet = 'M') then
          begin
            ptr := psys[xx,1];
            inven := psys[xx,2];
            topcost := systvals[ptr,2] * inven;
            tutils := systvals[ptr,3] * inven;
            offtop := offtop + topcost;
            write(systname[ptr], inven:6, systvals[ptr,1]:7);
            write(systvals[ptr,2]:8, systvals[ptr,3]:8,
              topcost:9);
            writeln(tutils:7, systvals[ptr,9]:8,
              systvals[ptr,10]:6);
          end;
        end;
      writeln('
          Offensive Total', offtop:19);
      for xx := 1 to 20 do
        if psys[xx,3] = 0 then
          begin
            templet := copy(systname[psys[xx,1]],1,1);
            if (templet = 'F') or (templet = 'A') then
              begin
                ptr := psys[xx,1];
                inven := psys[xx,2];
                topcost := systvals[ptr,2] * inven;
                tutils := systvals[ptr,3] * inven;

```

```

        deftop := deftop + topcost;
        write(systname[ptr], inven:6, systvals[ptr,1]:7);
        write(systvals[ptr,2]:8, systvals[ptr,3]:8,
              topcost:9);
        writeln(tutils:7, systvals[ptr,9]:8,
              systvals[ptr,10]:6);
    end;
end;
writeln('          Defensive Total', deftop:19);
totalop := offtop + deftop;
writeln('Total present operating cost',totalop:16);
write('          (enter an H for help or press',
      'return to continue)');
read(templet);
if((templet='H') or (templet='h')) then
    begin
        flag := 1;
        helpscr(flag);
        goto help1;
    end;
help2: clrscr;

flag := 0;
for xx := 1 to 20 do
    if psys[xx,3] <> 0 then
        begin
            if flag = 0 then
                begin
                    writeln('          Current',
                        'R&D Programs');
                    writeln('          PER UNIT',
                        'R & D COST');
                    writeln('          AVAILABLE -----',
                        '-----');
                    write('PURCH');
                    write('SYSTEM          IN          AQCOST  OPCOST',
                        'UTILS          YEAR ',year);
                    writeln('YEAR ', year+1, 'YEAR ',
                        year+2, 'LIMIT');
                    flag := 1;
                end;
            ptr := psys[xx,1];
            temparay[1] := systvals[ptr,6];
            temparay[2] := systvals[ptr,7];
            temparay[3] := systvals[ptr,8];
            temparay[4] := 0;      temparay[5] := 0;
            if psys[xx,3] > 0 then tempval := year -
                systvals[ptr,4]
            else tempval := -psys[xx,3] -
                systvals[ptr,4];
            if (psys[xx,3] = 999) or (tempval > 2)
                then tempval := 2;
        end;
    end;

```

```

tempval2 := systvals[ptr,5] - tempval;
write(systname[ptr], tempval2:8, 'yrs',
      systvals[ptr,1]:9);

write(systvals[ptr,2]:8, systvals[ptr,3]:7,
      temparay[tempval+1]:10);
write(temparay[tempval+2]:9,
      temparay[tempval+3]:8);
if psys[xx,3] > 0 then writeln(systvals[ptr,9]:8)
      else writeln(' shelved');
end;

if flag = 1 then writeln;
flag := 0;
for xx := 5 to 20 do
  if systvals[xx,4] = year then
    begin
      if flag = 0 then
        begin
          writeln('                               Proposed',
                  ' R&D Programs');
          write(' All of the following are expected',
                ' values except year ');
          writeln(year, ' R&D cost');
          writeln('                               PER',
                  ' UNIT                               R & D COST');
          writeln('                               AVAILABLE -----',
                  '----- PURCH');
          write(' SYSTEM          IN          AQCOST  OPCOST',
                ' UTILS          YEAR ', year);
          writeln(' YEAR ', year+1, ' YEAR ', year+2,
                  ' LIMIT');
          flag := 1;
        end;
      write( systname[xx],      systvals[xx,5]:8,
             'yrs', systvals[xx,1]:9);
      write( systvals[xx,2]:8, systvals[xx,3]:7,
             systvals[xx,6]:10);
      writeln(systvals[xx,7]:9, systvals[xx,8]:8,
              systvals[xx,9]:8);
    end;
  writeln;
  writeln('REMEMBER---- Systems may be bought the same',
          ' year final R&D costs are paid');
  writeln(' (enter an H for help or press a ',
          ' return to continue)');

```

```

read(templet);
if((templet= 'H') or (templet='h')) then
begin
    flag := 2;
    helpscr(flag);
    goto help2;
end;

{ **** now print intel reports ****}

help3: clrscr;
writeln(' ----- Intelligence report',
        ' -----');
writeln;
flag := trunc(100.0 * chancewar);
writeln('Intelligence reports indicate chance of war',
        ' this year = ',flag,' %');
writeln;
if intel[1] = 1 then
    for xx := 1 to 20 do
        if (csys[xx,1] <> 0) and (csys[xx,3] = 0) then
            if (copy(systname[csys[xx,1]],1,1) = 'B') or
                (copy(systname[csys[xx,1]],1,1) = 'M') then
                begin
                    write('Offensive system ',
                        systname[csys[xx,1]]);
                    tempval := trunc(csys[xx,2]/10) * 10;
                    writeln(' range = ',tempval:4,' to ',
                        tempval + 10);
                end;
if intel[2] = 1 then
    for xx := 1 to 20 do
        if (csys[xx,1] <> 0) and (csys[xx,3] = 0) then
            if (copy(systname[csys[xx,1]],1,1) = 'F') or
                (copy(systname[csys[xx,1]],1,1) = 'A') then
                begin
                    write('Defensive system ',
                        systname[csys[xx,1]]);
                    tempval := trunc(csys[xx,2]/10) * 10;
                    writeln(' range = ',tempval:4,' to ',
                        tempval + 10);
                end;
if intel[3] = 1 then
    for xx := 1 to 20 do
        if (csys[xx,1] <> 0) and (csys[xx,3] <> 0) then
            if (copy(systname[csys[xx,1]],1,1) = 'B') or
                (copy(systname[csys[xx,1]],1,1) = 'M') then
                begin
                    write('Offensive R & D for ',
                        systname[csys[xx,1]]);
                    writeln(' in progress. ');
                end;
end;

```

```

if intell[4] = 1 then
  for xx := 1 to 20 do
    if (csys[xx,1] <> 0) and (csys[xx,3] <> 0) then
      if (copy(systname[csys[xx,1]],1,1) = 'F') or
        (copy(systname[csys[xx,1]],1,1) = 'A') then
        begin
          write('Defensive R & D for ',
            systname[csys[xx,1]]);
          writeln(' in progress. ');
        end;
      writeln;
      write('      enter an H for help or press return',
        ' to continue ');
      readln(templet);
      if((templet= 'H') or (templet='h')) then
        begin
          flag := 3;
          helpscr(flag);
          goto help3;
        end;
    end;
  end;
end;

```

```

procedure Nyranlys(Nyr, Avlin, Acqcst, Opcst, Utls,
                  Rd1, Rd2, Rd3, purlim: integer);

var
  X, Opscost, Acqbugt, Sysbght, wpnsbght: integer;
  Totopscst, Totacqcst, Totlutils, Totrd, Yrsutls: real;
  Sysbgt: real;
  sp: string[2];

begin
  wpnsbght := 0;
  Totopscst := 0.0;
  Totacqcst := 0.0;
  Totlutils := 0.0;
  for X := Avlin + 1 to Nyr do
    begin
      Opscost := Opcst * wpnsbght;
      Acqbugt := budget - Opscost;
      Sysbgt := Int(Acqbugt / Acqcst);
      Sysbght := Trunc(Sysbgt);
      if sysbght > purlim then sysbght := purlim;
      wpnsbght := wpnsbght + Sysbght;
      Yrsutls := int(wpnsbght) * int(Utls);
      Totopscst := Totopscst + Int(Opscost);
      Totacqcst := Totacqcst + Int(Sysbght * Acqcst);
      Totlutils := Totlutils + Yrsutls;
    end;
  Totrd := Int(Rd1 + Rd2 + Rd3);
  if (totrd + totacqcst + totopscst) = 0 then
    utlsperdol := 0.0
  else
    utlsperdol := Totlutils /
      (Totrd + Totacqcst + Totopscst);
end;

```

```
procedure compturn;
```

```
  label
```

```
    goond, goono, goons;
```

```
  var
```

```
    xx, yy : integer;
```

```
    bestbmbflag, bestdefflag, bestmslflag : integer;
```

```
    budgetleft : integer;
```

```
    ptr : integer;
```

```
    templet : letter;
```

```
    tempval : integer;
```

```
    numbuy, bstopbmb, bstopmsl, bstopftr, bstopabm : integer;
```

```
    ofopscost, optr, dptr : integer;
```

```
    bstbmbinv, bstmslinv, bstftrinv, bstabminv : integer;
```

```
begin
```

```
  clrscr;
```

```
  writeln('Please wait while the computer selects its',  
    ' moves.');
```

```
  for xx := 1 to 20 do
```

```
    if (systvals[xx,4] <= year) and (systvals[xx,1] <> 0)
```

```
    then
```

```
      begin
```

```
        Nyranlys(10, systvals[xx,5], systvals[xx,1],  
          systvals[xx,2], systvals[xx,3],  
          systvals[xx,6], systvals[xx,7],  
          systvals[xx,8], systvals[xx,9]);
```

```
        systutil[xx] := utlsperdol;
```

```
      end;
```

```
  bestbmb := 0;    bestmsl := 0;  bestdef := 0;
```

```
  for xx := 1 to 20 do
```

```
    if csys[xx,1] <> 0 then
```

```
      begin
```

```
        ptr := csys[xx,1];
```

```
        templet := copy(systname[ptr],1,1);
```

```
        if systutil[ptr] <> 0 then
```

```
          begin
```

```
            if (templet = 'B') then
```

```
              if bestbmb = 0 then bestbmb := ptr
```

```
                else if systutil[bestbmb] <
```

```
                  systutil[ptr] then
```

```
                    bestbmb := ptr;
```

```
            if (templet = 'M') then
```

```
              if bestmsl = 0 then bestmsl := ptr
```

```
                else if systutil[bestmsl] <
```

```
                  systutil[ptr] then
```

```
                    bestmsl := ptr;
```

AD-A172 782

ARTIFICIAL INTELLIGENCE CONCEPTS AND THE WAR GAMING

2/2

ENVIRONMENT: A CASE S. (U) AIR FORCE INST OF TECH

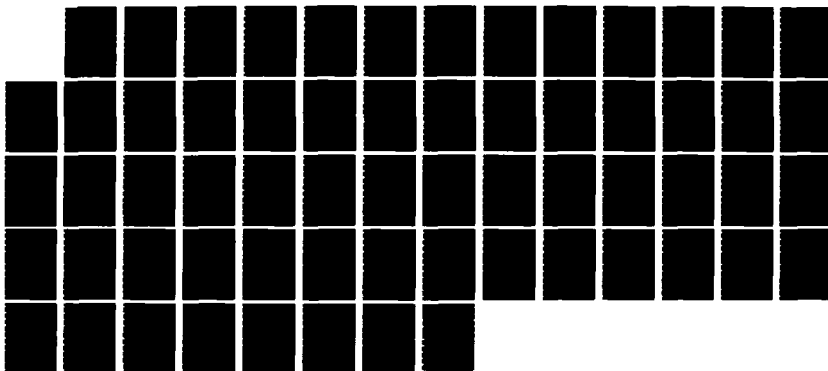
WRIGHT-PATTERSON AFB OH SCHOOL OF ENGI... G B WHITE

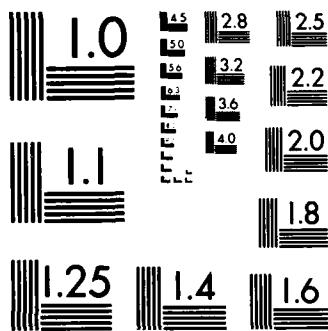
UNCLASSIFIED

MAR 86 AFIT/GCS/ENG/86M-1

F/G 15/7

NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

```

        if (templet = 'F') or (templet = 'A') then
            if bestdef = 0 then bestdef := ptr
            else if systutil[bestdef] <
                systutil[ptr] then
                bestdef := ptr;
        end;
    end;

for xx := 1 to 20 do
    if systvals[xx,4] = year then
        begin
            templet := copy(systname[xx],1,1);
            if systutil[xx] <> 0 then
                begin
                    if (templet = 'B') then
                        if systutil[bestbmb] < systutil[xx] then
                            bestbmb := xx;
                    if (templet = 'M') then
                        if systutil[bestmsl] < systutil[xx] then
                            bestmsl := xx;
                    if (templet = 'F') or (templet = 'A') then
                        if systutil[bestdef] < systutil[xx] then
                            bestdef := xx;
                end;
            end;

        budgetleft := budget;

{ ***** fund needed R&D ***** }
{   first, shelve any other R&D   }

        for xx:=1 to 20 do
            if (csys[xx,3] = 1) then
                if (csys[xx,1] <> bestbmb) and
                    (csys[xx,1] <> bestdef) and
                    (csys[xx,1] <> bestmsl) then csys[xx,3] := -year;

{   now, pay current R&D prices   }

        bestbmbflag := 0;   bestdefflag := 0;   bestmslflag :=0;
        for xx := 1 to 20 do
            begin
                if (csys[xx,1] = bestbmb) then
                    begin
                        bestbmbflag := 1;
                        tempval := (6 + systvals[bestbmb,5])-
                            (systvals[bestbmb,5]-
                                (year-systvals[bestbmb,4]));
                        budgetleft := budgetleft -
                            systvals[bestbmb,tempval];
                    end;
            end;

```

```

if (csys[xx,1] = bestdef) then
begin
    bestdefflag := 1;
    tempval := (6 + systvals[bestdef,5]) -
                (systvals[bestdef,5] -
                 (year-systvals[bestdef,4]));
    budgetleft := budgetleft -
                  systvals[bestdef,tempval];
end;
if (csys[xx,1] = bestmsl) then
begin
    bestmslflag := 1;
    tempval := (6 + systvals[bestmsl,5]) -
                (systvals[bestmsl,5] -
                 (year-systvals[bestmsl,4]));
    budgetleft := budgetleft -
                  systvals[bestmsl,tempval];
end;
end;

{ if flag = 0 then its a new R&D project and needs to }
{ be added to the array }

if bestdefflag = 0 then
for xx := 1 to 20 do
if csys[xx,1] = 0 then
begin
    csys[xx,1] := bestdef;
    csys[xx,2] := 0;
    csys[xx,3] := 1;
    budgetleft := budgetleft - systvals[bestdef,6];
    goto goond;
end;

goond: if bestbmbflag = 0 then
for xx := 1 to 20 do
if csys [xx,1] = 0 then
begin
    csys[xx,1] := bestbmb;
    csys[xx,2] := 0;
    csys[xx,3] := 1;
    budgetleft := budgetleft - systvals[bestbmb,6];
    goto goono;
end;

```

```

goono: if bestmslflag = 0 then
  for xx := 1 to 20 do
    if csys [xx,1] = 0 then
      begin
        csys[xx,1] := bestmsl;
        csys[xx,2] := 0;
        csys[xx,3] := 1;
        budgetleft := budgetleft - systvals[bestmsl,6];
        goto goons;
      end;

{***** if system finished R&D (paid last year),
      make it operational *****}

goons:  for xx := 1 to 20 do
  if (csys[xx,1] = bestbmb) or
    (csys[xx,1] = bestdef) or
    (csys[xx,1] = bestmsl) then
    begin
      ptr := csys[xx,1];
      tempval := systvals[ptr,4] + systvals[ptr,5] - 1;
      if tempval <= year then csys[xx,3] := 0;
    end;

{ **** now calculate best system of each weapon type **** }

  bstopbmb := 0;   bstopmsl := 0;   bstopftr := 0;
  bstopabm := 0;
  for xx := 1 to 20 do
    if (csys[xx,1] <> 0) and (csys[xx,3] = 0) then
      begin
        ptr := csys[xx,1];
        templet := copy(systname[ptr],1,1);
        if systutil[ptr] <> 0 then
          begin
            if (templet = 'B') then
              if bstopbmb = 0 then bstopbmb := xx
                else if systutil[bstopbmb] <
                  systutil[ptr] then
                  bstopbmb := xx;

            if (templet = 'M') then
              if bstopmsl = 0 then bstopmsl := xx
                else if systutil[bstopmsl] <
                  systutil[ptr] then
                  bstopmsl := xx;

            if (templet = 'F') then
              if bstopftr = 0 then bstopftr := xx
                else if systutil[bstopftr] <
                  systutil[ptr] then
                  bstopftr := xx;
          end;
      end;

```

```

        if (templet = 'A') then
            if bstopabm = 0 then bstopabm := xx
            else if systutil[bstopabm] <
                systutil[ptr] then
                    bstopabm := xx;
        end;
    end;

{ **** now determine best operational offensive
  wpns and pay ops costs ***** }

    optr := bstopbmb;  dptr := bstopmsl;
    if systutil[csys[bstopbmb,1]] <
        systutil[csys[bstopmsl,1]] then
        begin
            optr := bstopmsl;
            dptr := bstopbmb;
        end;
    if csys[dptr,2] > (csys[optr,2] / 1.7) then
        csys[dptr,2] := csys[dptr,2] - 12;
    if csys[dptr,2] < 0 then csys[dptr,2] := 0;
    if csys[optr,2] > (csys[dptr,2] * 2.01) then
        begin
            budgetleft := budgetleft -
                (5 * systvals[csys[dptr,1],2]);
            csys[dptr,2] := csys[dptr,2] + 5;
        end;
    ofopscost := csys[bstopmsl,2] *
        systvals[csys[bstopmsl,1],2] +
        csys[bstopbmb,2] *
        systvals[csys[bstopbmb,1],2];
    budgetleft := budgetleft - ofopscost;

{ *** now take out 1500 + from budget so that we are
  assured of some sort of balance in weapon systems ***}

    budgetleft := budgetleft - (1500 + year * 100);

{*** make sure not over budget now, if so, correct it***}

    while budgetleft < 0 do
        begin
            for xx := 1 to 20 do
                if (xx <> optr) and
                    (xx <> dptr) then csys[xx,2] := 0;
            csys[optr,2] := csys[optr,2] - 2;
            csys[dptr,2] := csys[dptr,2] - 1;
            budgetleft := budgetleft +
                (systvals[csys[optr,1],2] * 2) +
                systvals[csys[dptr,1],2];
        end;

```

```

{**** save inventory count of best weapons, scrap all
      the rest ***** }

bstbmbinv := csys[bstopbmb,2];
bstmslinv := csys[bstopmsl,2];
bstftrinv := csys[bstopftr,2];
bstabminv := csys[bstopabm,2];
for xx := 1 to 20 do
  csys[xx,2] := 0;

{ *** now buy the best offensive systems *** }
{ note:--- maintain a 2 to 1 ration of best to 2nd best }

numbuy := trunc(budgetleft /
  ((2 * systvals[csys[optr,1],1]) +
   ( systvals[csys[dptr,1],1])) );

{ ** make sure not > purch lim ** }

if (2 * numbuy) > systvals[csys[optr,1],9] then
  numbuy := trunc(systvals[csys[optr,1],9] / 2);

{ ** now add to inventory & calculate budgetleft ** }

budgetleft := budgetleft -
  ((2 * numbuy * systvals[csys[optr,1],1]) +
   ( numbuy * systvals[csys[dptr,1],1]));
if optr = bstopbmb then
  begin
    bstbmbinv := 2 * numbuy + bstbmbinv;
    bstmslinv := numbuy + bstmslinv;
  end
  else
  begin
    bstmslinv := 2 * numbuy + bstmslinv;
    bstbmbinv := numbuy + bstbmbinv;
  end;

{ **** now fill in inventory slots **** }

csys[bstopbmb,2] := bstbmbinv;
csys[bstopmsl,2] := bstmslinv;
csys[bstopftr,2] := bstftrinv;
csys[bstopabm,2] := bstabminv;

{ ***** now redo sequence using defensive systems ***** }

optr := bstopftr; dptr := bstopabm;
if systutil[csys[bstopftr,1]] <
  systutil[csys[bstopabm,1]] then
  begin
    optr := bstopabm;

```

```

    dptr := bstopftr;
end;
if csys[dptr,2] > (csys[optr,2] / 1.7) then
    csys[dptr,2] := csys[dptr,2] - 7;
if csys[dptr,2] < 0 then csys[dptr,2] := 0;
if csys[optr,2] > (csys[dptr,2] * 2.01) then
    csys[optr,2] := csys[optr,2] - 2;

{   add back in 1500+ we took away earlier   }

    budgetleft := budgetleft + (1500 + year * 100);

{   first pay ops cost for those in inventory   }

ofopscost := csys[optr,2] * systvals[csys[optr,1],2] +
             csys[dptr,2] * systvals[csys[dptr,1],2];
budgetleft := budgetleft - ofopscost;

while budgetleft < 0 do
begin
    xx := 0;
    if csys[optr,2] > 0 then
        if csys[optr,2] = 1 then
            begin
                xx := 1;
                csys[optr,2] := 0;
            end
        else
            begin
                xx := 2;
                csys[optr,2] := csys[optr,2]
                               - 2;
            end;
        end;
    yy := 0;
    if csys[dptr,2] > 0 then
        begin
            yy := 1;
            csys[dptr,2] := csys[dptr,2] - 1;
        end;
    budgetleft := budgetleft +
        (systvals[csys[optr,1],2] * xx) +
        (systvals[csys[dptr,1],2] * yy);
end;

{ **** if still can buy some more, do so **** }

numbuy := trunc(budgetleft /
    ((2 * systvals[csys[optr,1],1]) +
     (systvals[csys[dptr,1],1])) );
if (2 * numbuy) > systvals[csys[optr,1],9] then
    numbuy := trunc(systvals[csys[optr,1],9] / 2);

```

```

if numbuy > 0 then
  begin
    budgetleft := budgetleft -
      ((2 * numbuy *
        systvals[csys[optr,1],1]) +
      (
        numbuy *
        systvals[csys[dptr,1],1]));
    csys[optr,2] := csys[optr,2] + (2 * numbuy);
    csys[dptr,2] := csys[dptr,2] + numbuy;
  end;

```

```

{ ** can we squeeze in just one more wpn ? **}

if budgetleft > systvals[csys[optr,1],1] then
  if (2 * numbuy) <> systvals[csys[optr,1],9] then
    begin
      budgetleft := budgetleft -
        systvals[csys[optr,1],1];
      csys[optr,2] := csys[optr,2] + 1;
    end;
  if budgetleft > systvals[csys[dptr,1],1] then
    if numbuy <> systvals[csys[dptr,1],9] then
      begin
        budgetleft := budgetleft -
          systvals[csys[dptr,1],1];
        csys[dptr,2] := csys[dptr,2] + 1;
      end;

```

```

end;

```

```

procedure displaymenu;
begin
  clrscr;
  writeln;
  writeln('                Player Function Menu');
  writeln;
  writeln('1.  Reprint the reports. ');
  writeln('2.  Enter this years orders. ');
  writeln('3.  Perform weapon system analysis. ');
  writeln('4.  Finished, execute orders and proceed to ',
        'next year. ');
  writeln;
  writeln('Please enter the number of the option you wish ',
        'to perform. ');
  writeln('(or type H for help) ');
  writeln;
end;

```

```

procedure orders;

label
  redord, pl, pl5, p2, p3, tryagn, modlim, scrpng, prchlm;

var
  xx, ptr, flag : integer;
  answer : name;
  templet : letter;

begin
  for xx := 1 to 20 do
    begin
      tmparay[xx,1] := -1;
      modify[xx] := 0;
    end;
  tryagn: for xx := 1 to 20 do
    if (psys[xx,1] <> 0) and (psys[xx,3] = 0) then
      begin
        clrscr;
        ptr := psys[xx,1];
        write('For the ', systname[ptr],
              ' weapon system, you currently ');
        writeln('have ', psys[xx,2],
                ' in your inventory. ');
        prchlm: write('How many do you wish to BUY? ');
        readintnum(tmparay[xx,1]);
        if tmparay[xx,1] > systvals[ptr,9] then
          begin
            write('Your buy order exceeds the purchase ',
                  'limit for this ');
            writeln('system of ', systvals[ptr,9]);
            goto prchlm;
          end;
        writeln;
        scrpng: write('How many do you wish to SCRAP? ');
        readintnum(tmparay[xx,2]);
        if tmparay[xx,2] < 0 then
          begin
            writeln('You cannot have a negative ',
                    'SCRAP value. ');
            goto scrpng;
          end;
        if (systvals[psys[xx,1],11] > 0) and
           (psys[xx,2] > 0) then
          begin
            writeln;
            modlim: write('How many do you wish to be ',
                          'MODIFICATIONS? ');
            readintnum(modify[xx]);

```

```

        if modify[xx] > systvals[psys[xx,1],9] then
            begin
                write('Your modify order exceeds the',
                    ' limit for this ');
                writeln('system of ',systvals[psys[xx,1],9]);
                goto modlim;
            end;
        end;
    end;
    clrscr;
    writeln('Please review the following to see ',
        'if it is correct');
    writeln;
    writeln(' Weapon system      BUY      SCRAP',
        '      MODIFY      TOTAL');
    for xx := 1 to 20 do
        if tmparray[xx,1] <> -1 then
            begin
                write('      ',systname[psys[xx,1]],
                    tmparray[xx,1]:13, tmparray[xx,2]:15);
                write(modify[xx]:13);
                writeln((psys[xx,2] + modify[xx] + tmparray[xx,1]
                    - tmparray[xx,2]):13);
            end;
        writeln;
        writeln('Is this correct (Y/N)?');
        read(answer);
        if (copy(answer,1,1) <> 'Y') and
            (copy(answer,1,1) <> 'y') then
            begin
                for xx := 1 to 20 do
                    tmparray[xx,1] := -1;
                writeln;
                writeln('Do you want to try again (Y/N)?');
                read(answer);
                if (copy(answer,1,1) = 'Y') or
                    (copy(answer,1,1) = 'y') then goto tryagn;
            end;

    {***** now check for R&D *****}

    redord: for xx := 1 to 20 do
        begin
            rndarray[xx,1] := 0;
            rndarray[xx,2] := 0;
            rndarray[xx,3] := 0;
        end;

```

```

flag := 0;
clrscr;
for xx := 1 to 20 do
  if systvals[xx,4] = year then
    begin
      flag := 1;
      write('Do you want to fund research for ',
        systname[xx], ' ?');
      write(' (Y/N) ');
      readln(answer);

      if ((copy(answer,1,1) = 'Y') or
        (copy(answer,1,1) = 'y')) then
        begin
          rndaray[xx,1] := 1;
          if systvals[xx,5] = 1 then
            begin
              pl:
                write('How many ',systname[xx],
                  ' systems do you want to buy? ');
                readintnum(rndaray[xx,2]);
                if rndaray[xx,2] > systvals[xx,9] then
                  begin
                    write('Your buy order exceeds ',
                      'the purchase limit for this ');
                    writeln('system of ',systvals[xx,9]);
                    goto pl;
                  end;
                if systvals[xx,11] <> 0 then
                  begin
                    pl5:
                      write('How many ',systname[xx],
                        ' systems do you want',
                        ' to modify? ');
                      readintnum(rndaray[xx,3]);
                      if rndaray[xx,3] > systvals[xx,9] then
                        begin
                          write('Your modify order',
                            ' exceeds the limit',
                            ' for this ');
                          writeln('system of ',
                            systvals[xx,9]);
                          goto pl5;
                        end;
                      end;
                    end;
                  end
                else rndaray[xx,2] := 0;
            end;
          end;
        end;
      for xx := 1 to 20 do
        if (psys[xx,3] = 1) or (psys[xx,3] = 999) then
          begin
            flag := 1;
            write('Do you want to continue research for ');
            write(systname[psys[xx,1]], ' ? (Y/N) ');

```

```

readln(answer);
if ((copy(answer,1,1) = 'Y') or
    (copy(answer,1,1) = 'y')) then
begin
    rndaray[psys[xx,1],1] := 1;
    if ((systvals[psys[xx,1],5] +
        systvals[psys[xx,1],4] - 1)=year)
        or (psys[xx,3] = 999) then
        begin
p2:      write('How many ',systname[psys[xx,1]],
              ' systems do you want to buy? ');
          readintnum(rndaray[psys[xx,1],2]);

          if rndaray[psys[xx,1],2] >
             systvals[psys[xx,1],9] then
            begin
                write('Your buy order exceeds ',
                      'the purchase limit for this ');
                writeln('system of ',
                        systvals[psys[xx,1],9]);
                goto p2;
            end;
        end
    else
        rndaray[psys[xx,1],2] := 0;
    end;
end;
end;
for xx := 1 to 20 do
    if psys[xx,3] < 0 then
        begin
            flag := 1;
            write('Do you want to resume research',
                  ' for the shelved ');
            write(systname[psys[xx,1]], ' ? (Y/N) ');
            readln(answer);
            if ((copy(answer,1,1) = 'Y') or
                (copy(answer,1,1) = 'y')) then
                begin
                    rndaray[psys[xx,1],1] := 1;
                    if (-psys[xx,3] >=(systvals[psys[xx,1],4]-1)+
                        systvals[psys[xx,1],5]) then
p3:      begin
                            write('How many ',systname[psys[xx,1]],
                                  ' systems do you want to buy? ');
                            readintnum(rndaray[psys[xx,1],2]);
                            if rndaray[psys[xx,1],2] >
                               systvals[psys[xx,1],9] then
                                begin
                                    write('Your buy order exceeds the',
                                          ' purchase limit for this ');
                                    writeln('system of ',
                                            systvals[psys[xx,1],9]);
                                    goto p3;
                                end
                            end
                        end
                    end
                end
            end
        end
    end
end

```

```

                                end;
                                end
                                else
                                rndaray[psys[xx,1],2] := 0;
                                end;
                                end;

                                if flag = 1 then
                                begin
                                writeln;
                                write('Please review this information.',
                                ' Is it correct? (Y/N) ');
                                readln(answer);

                                if ((copy(answer,1,1) <> 'Y') and
                                (copy(answer,1,1) <> 'y')) then
                                goto redord;
                                end;

                                { ***** now check to see about intel reports wanted ***** }

                                for xx := 1 to 4 do
                                tintel[xx] := 0;
                                clrscr;
                                write('Do you want any intelligence reports? (Y/N) ');
                                readln(templet);
                                if ((copy(templet,1,1) = 'Y') or
                                (copy(templet,1,1) = 'y')) then
                                begin
                                writeln;
                                writeln('At 100 $ per intelligence report, ',
                                'do you want a report on');

                                write ('      Offensive systems ? (Y/N) ');
                                readln(templet);
                                if ((copy(templet,1,1) = 'Y') or
                                (copy(templet,1,1) = 'y')) then
                                tintel[1] := 1;

                                write ('      Defensive systems ?      ');
                                readln(templet);
                                if ((copy(templet,1,1) = 'Y') or
                                (copy(templet,1,1) = 'y')) then
                                tintel[2] := 1;

                                write ('      Offensive R & D ? ');
                                readln(templet);
                                if ((copy(templet,1,1) = 'Y') or
                                (copy(templet,1,1) = 'y')) then
                                tintel[3] := 1;

```

```
write ('      Defensive R & D ? ');
readln(templet);
if ((copy(templet,1,1) = 'Y') or
    (copy(templet,1,1) = 'y')) then
    tintel[4] := 1;
end;
```

```
end;
```

```

procedure wpanalys;

  label
    agoon, ogoon;

  var
    templet : letter;
    nyrs, rndlft, acost, ocost, utlvl: integer;
    rnd1, rnd2, rnd3, prlm, tempval : integer;
    wpnname : name;

  begin
    clrscr;
    writeln;
    write('Enter the name of the weapon system. ');
    readln(wpnname);

    write('Enter the number of years ',
          'to run the analysis. ');
    readintnum(nyrs);

    write('Enter the number of years of R&D left. ');
    readintnum(rndlft);

    agoon: write('Enter the system acquisition cost. ');
    readintnum(acost);
    if acost <= 0 then
      begin
        writeln('ERROR---Cannot have a negative or ',
                'zero acquisition cost');
        goto agoon;
      end;

    ogoon: write('Enter the system yearly operation cost. ');
    readintnum(ocost);
    if ocost <= 0 then
      begin
        writeln('ERROR---Cannot have a negative or ',
                'zero operation cost');
        goto ogoon;
      end;

    write('Enter the system util value. ');
    readintnum(utlvl);

    write('Enter the 1st year R&D cost ',
          '(enter a 0 if none). ');
    readintnum(rnd1);

    write('Enter the 2nd year R&D cost. ');
    readintnum(rnd2);

```

```

write('Enter the 3rd year R&D cost. ');
readintnum(rnd3);

write('Enter the purchase limit. ');
readintnum(prlm);

nyranlys(nyrs, rndlft, acost, ocost,
         utlvl, rnd1, rnd2, rnd3, prlm);
tempval := trunc(utlsperdol * 100);

writeln;
writeln('The utils/$ percentage for the ',wpnname,
        ' is ',tempval,' %');
writeln;
write('press return to continue');
readln(templet);

end;

```

```

procedure exorders;

label
  goon, goon2 ;

var
  xx, yy : integer;
  tempval : integer;
  templet : letter;

begin
  pbudgetleft := pbudget;

  { **** first take care of scraps, then ops costs, **** }
  then buys

  for xx := 1 to 20 do
    if (tmparray[xx,1] <> -1) and (tmparray[xx,2] <> 0) then
      begin
        psys[xx,2] := psys[xx,2] - tmparray[xx,2];
        if psys[xx,2] < 0 then psys[xx,2] := 0;
      end;
  for xx := 1 to 20 do
    if (psys[xx,2] <> 0) then
      pbudgetleft := pbudgetleft -
        (psys[xx,2] * systvals[psys[xx,1],2]);
  for xx := 1 to 20 do
    if (tmparray[xx,1] <> -1) and (tmparray[xx,1] <> 0) then
      begin
        psys[xx,2] := psys[xx,2] + tmparray[xx,1];
        pbudgetleft := pbudgetleft -
          (tmparray[xx,1] *
            systvals[psys[xx,1],1]);
      end;

  { **** now take care of ongoing rnd projects **** }

  for xx := 1 to 20 do
    if (psys[xx,1] <> 0) and (psys[xx,3] <> 0) then
      begin
        if psys[xx,3] = 999 then
          begin
            {***** r&D had been scratched and later continued *****)
            if rndarray[psys[xx,1],1] = 1 then
              begin { * and want to keep it going *}
                psys[xx,3] := 777;
                psys[xx,2] := rndarray[psys[xx,1],2];
                pbudgetleft := pbudgetleft -
                  systvals[psys[xx,1],8];
                pbudgetleft := pbudgetleft -
                  (psys[xx,2] * systvals[psys[xx,1],1]);
              end
            else { * want to scratch it once more *}

```

```

        psys[xx,3] := -year;
    end; { * end psys[xx,3] := 999 *}
    if (psys[xx,3] = 1) then
        { ** then research paid for last year **}

        if rndaray[psys[xx,1],1] = 1 then
            begin
                {*** and want to continue this year ***}

                if (systvals[psys[xx,1],4] +
                    systvals[psys[xx,1],5] - 1) <= year then
                    begin

                        {**** system gone operational, count new systems ****}

                        psys[xx,2] := rndaray[psys[xx,1],2];
                        psys[xx,3] := 0;
                        pbudgetleft := pbudgetleft -
                                      (psys[xx,2] *
                                       systvals[psys[xx,1],1]);

                        end;
                        tempval := 6 +
                                   (year- systvals[psys[xx,1],4]);
                        pbudgetleft := pbudgetleft -
                                      systvals[psys[xx,1],tempval];
                    end
                else
                    {*** don't want to continue ***}
                    psys[xx,3] := -year
                    else
                        { ** else, research for this had been scratched **}
                        if (psys[xx,3] <> 777) then
                            begin
                                if rndaray[psys[xx,1],1] = 1 then
                                    begin
                                        if (systvals[psys[xx,1],5] = 2) then
                                            begin

                                                { ** shelved had 2 yr r&d span, thus now op **}

                                                psys[xx,3] := 0;
                                                psys[xx,2] := rndaray[psys[xx,1],2];
                                                pbudgetleft := pbudgetleft -
                                                                (psys[xx,2] *
                                                                 systvals[psys[xx,1],1]);
                                                tempval := trunc(1.5 *
                                                                systvals[psys[xx,1],7]);

                                                end
                                            else
                                                { ** shelved had 3 year r&d span ** }
                                                if (-psys[xx,3] >=

```

```

                                (systvals[psys[xx,1],4]-1)+
                                systvals[psys[xx,1],5]) then
begin

(* shelved with one yr to go, thus now op *)

    psys[xx,3] := 0;
    psys[xx,2] :=
        rndaray[psys[xx,1],2];
    pbudgetleft := pbudgetleft -
        (psys[xx,2] *
         systvals[psys[xx,1],1]);
    tempval := trunc(1.5 *
                     systvals[psys[xx,1],8]);
end
else
begin

(* shelved with two yrs to go*)

    psys[xx,3] := 999;
    tempval := trunc(1.5 *
                     systvals[psys[xx,1],7]);
end;
    pbudgetleft := pbudgetleft - tempval;
end;
end;
    if psys[xx,3] = 777 then psys[xx,3] := 0;
end;

( **** now handle new rnd projects **** )

for xx := 1 to 20 do
begin
    if (rndaray[xx,1] <> 0) and
        (systvals[xx,4] = year) then
        for yy := 1 to 20 do
            if psys[yy,1] = 0 then
            begin
                psys[yy,1] := xx;
                psys[yy,2] := rndaray[xx,2];
                if systvals[xx,5] = 1 then
                begin
                    { *** system operational *** }
                    psys[yy,3] := 0;
                    psys[yy,2] := rndaray[xx,2];
                    pbudgetleft := pbudgetleft -
                        (psys[yy,2] *
                         systvals[xx,1]);
                    if systvals[xx,11] <> 0 then
                    begin
                        systvals[xx,11] :=
                            -systvals[xx,11];
                    end
                end
            end
        end
    end
end

```

```

systvals[systvals[xx,11],11] :=
    - systvals[systvals[xx,1],11];

if rndarray[xx,3] >
    psys[systvals[xx,11],2] then
    rndarray[xx,3] :=
        psys[systvals[xx,11],2];
    psys[systvals[xx,11],2] :=
        psys[systvals[xx,11],2] -
            rndarray[xx,3];
    psys[yy,2] := psys[yy,2] +
        rndarray[xx,3];
    pbudgetleft := pbudgetleft -
        (rndarray[xx,3] *
            systvals[systvals[xx,11],10]);
    end;
    end
    else psys[yy,3] := 1;
    pbudgetleft := pbudgetleft -
        (systvals[psys[yy,1],6]);
    goto goon;
end;
goon: end;

{ *** handle modify for systems which had
    r&d done in a previous yr *** }

for xx := 1 to 20 do
    if modify[xx] > 0 then
        begin
            if modify[xx] > psys[systvals[psys[xx,1],11],2] then
                modify[xx] := psys[systvals[psys[xx,1],11],2];
            psys[systvals[psys[xx,1],11],2] :=
                psys[systvals[psys[xx,1],11],2] - modify[xx];
            psys[xx,2] := psys[xx,2] + modify[xx];
            pbudgetleft := pbudgetleft -
                (modify[xx] *
                    systvals[systvals[psys[xx,1],11],10]);
        end;

{ **** now account for intel reports costs **** }

for xx := 1 to 4 do
    begin
        if tintel[xx] <> 0 then pbudgetleft := pbudgetleft - 100;
        intel[xx] := tintel[xx];
    end;

writeln('you spent this much this year ',
    pbudget-pbudgetleft);
read(templet);

```

```

overspent := 0;

{ *** check to see if overspent budget,
  if so, penalize by amount          ***** }

if pbudgetleft < 0 then
begin
  pbudgetny := pbudget + pbudgetleft;
  pbudget := pbudget + pbudgetleft;
  writeln(' *****   You overspent your budget ',
    'for this year. *****');
  writeln('This will result in a reduction in your ',
    'next years budget. ');
  writeln('In addition, current weapon systems had ',
    'to be scratched in ');
  writeln('order to get your budget in line. ');
  write('      press return to continue ');
  read(templet);
  overspentcnt := overspentcnt + 1;
  overspent := overspentcnt;

  {*** scrap systems, starting with newest in inventory ***}

  while pbudgetleft < 0 do
  begin
    for xx := 20 downto 0 do
    begin
      if xx = 0 then
      begin
        pbudgetleft := 0;
        goto goon2;
      end
      else
      if psys[xx,2] > 0 then
      begin
        psys[xx,2] := psys[xx,2] - 1;
        pbudgetleft := pbudgetleft +
          systvals[psys[xx,1],2];
        goto goon2;
      end;
    end;
  end;
goon2:  end;
end;

end;

```

```

procedure playerturn;

var
  answer : letter;
  num : integer;

begin
  answer := ' ';
  while answer <> '4' do
    begin
      displaymenu;
      read(answer);
      if length(answer) <> 1 then answer := ' ';
      case answer of
        '1': report;
        '2': orders;
        '3': wpnanalys;
        '4': exorders;
        'h','H': begin
                    num := 4;
                    helpscr(num);
                  end;
      else
        writeln('Improper response, hit ',
              'return to continue');
        read(answer);
      end; {end case}
    end;
  end;
end;

```

```
procedure endofyear;
```

```
var
```

```
  ptr, xx : integer;  
  templet : letter;  
  cofftot, pofftot: integer;  
  t1, t2 : integer;
```

```
begin
```

```
  cbmbtot := 0;  cbmbcnt := 0;  
  cmsltot := 0;  cmslcnt := 0;  
  cfttot := 0;  cftrent := 0;  
  cabmtot := 0;  cabmcnt := 0;
```

```
  for xx := 1 to 20 do
```

```
    if (csys[xx,1] <> 0) and (csys[xx,3] = 0) then
```

```
      begin
```

```
        ptr := csys[xx,1];
```

```
        templet := copy(systname[ptr],1,1);
```

```
        if (templet = 'B') then
```

```
          begin
```

```
            cbmbcnt := cbmbcnt + csys[xx,2];
```

```
            cbmbtot := cbmbtot +  
              (systvals[ptr,3] * csys[xx,2]);
```

```
          end;
```

```
        if (templet = 'M') then
```

```
          begin
```

```
            cmslcnt := cmslcnt + csys[xx,2];
```

```
            cmsltot := cmsltot +  
              (systvals[ptr,3] * csys[xx,2]);
```

```
          end;
```

```
        if (templet = 'F') then
```

```
          begin
```

```
            cftrent := cftrent + csys[xx,2];
```

```
            cfttot := cfttot +  
              (systvals[ptr,3] * csys[xx,2]);
```

```
          end;
```

```
        if (templet = 'A') then
```

```
          begin
```

```
            cabmcnt := cabmcnt + csys[xx,2];
```

```
            cabmtot := cabmtot +  
              (systvals[ptr,3] * csys[xx,2]);
```

```
          end;
```

```
      end;
```

```
  pbmbtot := 0;  pbmbcnt := 0;  
  pmsltot := 0;  pmslcnt := 0;  
  pfttot := 0;  pftrent := 0;  
  pabmtot := 0;  pabmcnt := 0;
```

```

for xx := 1 to 20 do
  if (psys[xx,1] <> 0) and (psys[xx,3] = 0) then
    begin
      ptr := psys[xx,1];
      templet := copy(systname[ptr],1,1);
      if (templet = 'B') then
        begin
          pbmbcnt := pbmbcnt + psys[xx,2];
          pbmbtot := pbmbtot +
            (systvals[ptr,3] * psys[xx,2]);
        end;
      if (templet = 'M') then
        begin
          pmslcnt := pmslcnt + psys[xx,2];
          pmsltot := pmsltot +
            (systvals[ptr,3] * psys[xx,2]);
        end;
      if (templet = 'F') then
        begin
          pfttrcnt := pfttrcnt + psys[xx,2];
          pfttrtot := pfttrtot +
            (systvals[ptr,3] * psys[xx,2]);
        end;
      if (templet = 'A') then
        begin
          pabmcnt := pabmcnt + psys[xx,2];
          pabmtot := pabmtot +
            (systvals[ptr,3] * psys[xx,2]);
        end;
    end;
end;

```

```

{ ***** }
{ *** apply penalties for unbalanced forces *** }
{ unbalanced means that have less than half the number of }
{ one system than another. For example, if we have }
{ 100 bmbrs and 75 msls we have a balanced force. If, }
{ however, we had 100 bmbrs but only 40 msls this is }
{ unbalanced. The penalty in this case is }
{ utils * ((2.01 * 40)/100) or approx 8/10 of the }
{ util values }
{ ***** }

```

```

if cbmbcnt > cmslcnt then {more bombers}
  if (trunc(2.01 * cmslcnt)) < cbmbcnt then
    cbmbtot := trunc(((2.01 * cmslcnt) /
      cbmbcnt) * cbmbtot);

if cmslcnt > cbmbcnt then {more missiles}
  if (trunc(2.01 * cbmbcnt)) < cmslcnt then
    cmsltot := trunc(((2.01 * cbmbcnt) /
      cmslcnt) * cmsltot);

```

```

if cftrent > cabmcnt then {more fighters}
  if (trunc(2.01 * cabmcnt)) < cftrent then
    cfttot := trunc(((2.01 * cabmcnt) /
                     cftrent) * cfttot);

if cabmcnt > cftrent then { more abm's }
  if (trunc(2.01 * cftrent)) < cabmcnt then
    cabmtot := trunc(((2.01 * cftrent) /
                     cabmcnt) * cabmtot);

if pbmbcnt > pmslcnt then {more bombers}
  if (trunc(2.01 * pmslcnt)) < pbmbcnt then
    pbmbtot := trunc(((2.01 * pmslcnt) /
                     pbmbcnt) * pbmbtot);

if pmslcnt > pbmbcnt then {more missiles}
  if (trunc(2.01 * pbmbcnt)) < pmslcnt then
    pmsltot := trunc(((2.01 * pbmbcnt) /
                     pmslcnt) * pmsltot);

if pftrent > pabmcnt then {more fighters}
  if (trunc(2.01 * pabmcnt)) < pftrent then
    pfttot := trunc(((2.01 * pabmcnt) /
                     pftrent) * pfttot);

if pabmcnt > pftrent then { more abm's }
  if (trunc(2.01 * pftrent)) < pabmcnt then
    pabmtot := trunc(((2.01 * pftrent) /
                     pabmcnt) * pabmtot);

if cbmbtot < 0 then cbmbtot := 32700;
if cmsltot < 0 then cmsltot := 32700;
if pfttot < 0 then pfttot := 32700;
if pabmtot < 0 then pabmtot := 32700;
if pbmbtot < 0 then pbmbtot := 32700;
if pmsltot < 0 then pmsltot := 32700;
if cfttot < 0 then cfttot := 32700;
if cabmtot < 0 then cabmtot := 32700;

t1 := cbmbtot - pfttot;
t2 := cmsltot - pabmtot;
if ((t1 < 0) or (t2 < 0)) then overdefend := 1
  else overdefend := 0;

if t1 < 0 then t1 := 0;
if t2 < 0 then t2 := 0;
cofftot := t1 + t2;
t1 := pbmbtot - cfttot;
t2 := pmsltot - cabmtot;
if t1 < 0 then t1 := 0;
if t2 < 0 then t2 := 0;

```

```

pofftot := t1 + t2;
clrscr;
if cofftot < 0 then cofftot := 32700;
if pofftot < 0 then pofftot := 32700;

utilaray[year,1] := trunc(cofftot/10) * 10;
utilaray[year,2] := trunc(pofftot/10) * 10;

writeln('End of year intelligence reports',
        'indicated that');
if cofftot > pofftot then
    writeln('you were behind in the "Arms Race" ',
            'for this year. ');
if pofftot > cofftot then
    writeln('you were ahead in the "Arms Race" ',
            'for this year. ');
if pofftot = cofftot then
    writeln('both sides were at about the same ',
            'level in the "Arms Race". ');

writeln;
writeln('hit a return to continue');
read(templet);

```

```

{ ***** }
{ ***** possibly change values of some of the r&d systems }
{ this is to reflect changes in economy and bad estimates }
{ from competing defense contractors }
{ ***** }

```

```

for xx := 1 to 20 do
begin
    if (systvals[xx,4] = year) and
        (systvals[xx,5] = 2) then
        begin
            systvals[xx,1] :=
                trunc((systvals[xx,1]/1.5 +
                    (systvals[xx,1] * random))/10.0) * 10;
            if systvals[xx,2] > systvals[xx,1] then
                systvals[xx,2] := systvals[xx,1] - 10;
            systvals[xx,7] := trunc((systvals[xx,7]/1.6 +
                (systvals[xx,7] * random))/10.0) * 10;
            systvals[xx,3] := trunc((systvals[xx,3]/2.1 +
                (systvals[xx,3] * random))/10.0) * 10;
        end;
end;

```

```

if (systvals[xx,4]+1 = year) and
  (systvals[xx,5] = 3) then
  begin
    systvals[xx,2] := trunc((systvals[xx,2]/1.8 +
      (systvals[xx,2] * random))/10.0) * 10;
    systvals[xx,8] := trunc((systvals[xx,8]/1.7 +
      (systvals[xx,8] * random))/10.0) * 10;
    systvals[xx,3] := trunc((systvals[xx,3]/2.2 +
      (systvals[xx,3] * random))/10.0) * 10;
  end;
end;

```

```

{ ***** calculate if there is a war or not ***** }

```

```

if chancewar > random then
  begin
    clrscr;
    writeln;
    writeln('
      W      W      W',
      A      RRRRRRR');
    writeln('
      W      W W      W',
      A A      R      R');
    writeln('
      W      W      W      W',
      AAAAAA      RRRRRRR');
    writeln('
      W W      W W      W',
      A      A      R      RR');
    writeln('
      W      W      W',
      A      A      R      RR');
    writeln;
    writeln('Due to worldwide tensions, a war broke out',
      'this year. This brief war ended with');
  end;

```

```

if cofftot < pofftot then
  begin
    budget := budget - 800;
    pbudget := pbudget - 200;
    pbudgetny := poudgetny - 200;
    if utilaray[year,1] = 0 then utilaray[year,1] := -1
    else {flag this as a war year}
      utilaray[year,1] := -utilaray[year,1];
    writeln('you as the VICTOR. Your budget, however,',
      'was still reduced by $200. ');
  end;

```

```

if cofftot > pofftot then
  begin
    budget := budget - 200;
    pbudget := pbudget - 800;
    pbudgetny := pbudgetny - 800;
    if utilaray[year,2] = 0 then utilaray[year,2] := -1
    else

```

```

        utilaray[year,2] := -utilaray[year,2];
        writeln('you as the LOSER. Your budget,',
                ' consequently, has been reduced by $800.');
```

end;

```

if cofftot = pofftot then
begin
    budget := budget - 400;
    pbudget := pbudget - 400;
    if utilaray[year,1] = 0 then utilaray[year,1] := -1
    else
        utilaray[year,1] := -utilaray[year,1];
    if utilaray[year,2] = 0 then utilaray[year,2] := -1
    else
        utilaray[year,2] := -utilaray[year,2];

    writeln('a truce between forces.',
            ' Your budget has been reduced by $400.');
```

end;

```

{ ***** make it harder to have another war ***** }
```

```

    warcount := warcount + 2.5;
    writeln;
    write('press return to continue');
    readln(templet);
end;
```

```

{ **** calculate chance of war next year **** }
```

```

    chancewar := 0.0;
    if cofftot < pofftot then
        chancewar := (pofftot-cofftot) / (2000 + (500*year));
    if pofftot < cofftot then
        chancewar := (cofftot-pofftot) / (2000 + (500*year));
    chancewar := chancewar / (warcount + 1.0);
    if chancewar < 0.15 then chancewar := 0.15;
```

end;

```

procedure endofgame;

var
  xx : integer;
  comptot, playtot : real;

begin
  clrscr;
  writeln;
  writeln(' The game is over. The yearly util ',
           ' totals are as follows: ');
  writeln;
  writeln('Year          Computers total          ',
           ' Players total');
  comptot := 0.0; playtot := 0.0;
  for xx := 1 to 8 do
    begin
      if (utilaray[xx,1] >= 0) and
         (utilaray[xx,2] >= 0) then
        writeln(' ',xx, utilaray[xx,1]:20,
                 utilaray[xx,2]:26)
      else
        begin
          utilaray[xx,1] := abs(utilaray[xx,1]);
          utilaray[xx,2] := abs(utilaray[xx,2]);
          writeln(' ',xx, utilaray[xx,1]:20,
                   utilaray[xx,2]:26,
                   ' war year');
        end;
      comptot := comptot + int(utilaray[xx,1]);
      playtot := playtot + int(utilaray[xx,2]);
    end;

  writeln;
  writeln('Computer grand total = ',
           trunc(comptot/10),'0');
  writeln('Players grand total = ',
           trunc(playtot/10),'0');
  writeln;
  write('Based on the above data, ');
  if comptot > playtot then
    write('the COMPUTER has');
  if comptot < playtot then write('YOU have');
  if comptot = playtot then
    write('NOBODY (a draw) has');
  writeln(' been declared the winner. ');
end;

```

```
procedure prntrprt;
```

```
var
```

```
  xx : integer;  
  tempstr, tempstrg : name;
```

```
begin
```

```
  tempstr := 'YEAR.';  
  str(year:1,tempstrg);  
  tempstr := concat(tempstr,tempstrg);
```

```
  assign(sysrprt,tempstr);  
  rewrite(sysrprt);
```

```
  with lineprt do  
    begin
```

```
      linechr := '  
      write(sysrprt,lineprt);
```

```
      linechr := 'pbmbcnt = '  
      str(pbmbcnt:3,tempstr);  
      linechr := concat(linechr,tempstr, '. ');  
      write(sysrprt,lineprt);
```

```
      linechr := 'pmslcnt = '  
      str(pmslcnt:3,tempstr);  
      linechr := concat(linechr,tempstr, '. ');  
      write(sysrprt,lineprt);
```

```
      linechr := 'pfttrcnt = '  
      str(pfttrcnt:3,tempstr);  
      linechr := concat(linechr,tempstr, '. ');  
      write(sysrprt,lineprt);
```

```
      linechr := 'pabmcnt = '  
      str(pabmcnt:3,tempstr);  
      linechr := concat(linechr,tempstr, '. ');  
      write(sysrprt,lineprt);
```

```
      linechr := 'bestbmb = '  
      str(systutil[bestbmb]:3,tempstr);  
      linechr := concat(linechr,tempstr, '. ');  
      write(sysrprt,lineprt);
```

```
      linechr := 'bestmsl = '  
      str(systutil[bestmsl]:3,tempstr);  
      linechr := concat(linechr,tempstr, '. ');  
      write(sysrprt,lineprt);
```

```

if (utilaray[year,1] >=0) and
  (utilaray[year,2] >=0) then
  linechr := 'waryear = 0.
else
  if (utilaray[year,1] < 0) and
    (utilaray[year,2] < 0) then
    linechr := 'waryear = 3.           {tie in war}
  else
    if utilaray[year,1] < 0 then
      linechr := 'waryear = 2.       {player won}
    else
      linechr := 'waryear = 1.       ;{computer }
write(sysrprt,lineprt);

linechr := 'overspent = ';
str(overspent:1,tempstr);
linechr := concat(linechr,tempstr, '. ');
write(sysrprt,lineprt);

if pbudgetleft > 100 then
  linechr := 'budgetleft = 1.
else
  linechr := 'budgetleft = 0.
write(sysrprt,lineprt);

linechr := 'overdefend = ';
str(overdefend:1, tempstr);
linechr := concat(linechr,tempstr, '. ');
write(sysrprt,lineprt);

linechr := '
write(sysrprt,lineprt);

end;
close(sysrprt);
end;

```

```

{***** main routine *****}

begin
  init;
  for year := 1 to 8 do
    begin
      budgetny := budget +
        (Trunc ((700.0 * Random) / 10.0) * 10);
      if budget = pbudget then pbudgetny := budgetny
      else pbudgetny := pbudget +
        (Trunc ((700.0 * random) / 10.0) * 10);

      report;
      playerturn;
      compturn;
      endofyear;
      tempbudget := Trunc((500.0 * Random) / 10.0);
      budget := budget + (tempbudget * 10);
      if budgetny = pbudgetny then pbudget := budget
      else pbudget := pbudget + (tempbudget * 10);
      if pbudget < 1500 then pbudget := 1500;

      prntrprt;

    end;
  endofgame;

end.

```

```
{***** program to initialize the weapon system file *****}
```

```
program initwsys;
```

```
type
```

```
  name = string[6];
```

```
  fileline = record
```

```
    sysname : name;
```

```
    acqcost,
```

```
    opscost,
```

```
    utilval,
```

```
    yrforrd,
```

```
    availin,
```

```
    rndyr1,
```

```
    rndyr2,
```

```
    rndyr3,
```

```
    purchlm,
```

```
    modcost,
```

```
    modflag : integer;
```

```
  end;
```

```
var
```

```
  sysdata : file of fileline;
```

```
  tempdata : fileline;
```

```
begin
```

```
  assign(sysdata, 'WPNSYS.DAT');
```

```
  rewrite(sysdata);
```

```
  with tempdata do
```

```
    begin
```

```
      sysname := 'B-1  ';
```

```
      acqcost := 50;
```

```
      opscost := 30;
```

```
      utilval := 20;
```

```
      yrforrd := 0;
```

```
      availin := 0;
```

```
      rndyr1 := 0;
```

```
      rndyr2 := 0;
```

```
      rndyr3 := 0;
```

```
      purchlm := 30;
```

```
      modcost := 0;
```

```
      modflag := 0;
```

```
      write(sysdata, tempdata);
```

```
      sysname := 'M-1  ';
```

```
      acqcost := 70;
```

```
      opscost := 50;
```

```
      utilval := 40;
```

```
      yrforrd := 0;
```

```
      availin := 0;
```

```
      rndyr1 := 0;
```

```
      rndyr2 := 0;
```

```

rndyr3 := 0;
purchlm := 20;
modcost := 50;
modflag := -10;
    write(sysdata, tempdata);
sysname := 'F-1';
acqcost := 100;
opscost := 60;
utilval := 50;
yrforrd := 0;
availin := 0;
rndyr1 := 0;
rndyr2 := 0;
rndyr3 := 0;
purchlm := 30;
modcost := 70;
modflag := -7;
    write(sysdata, tempdata);
sysname := 'ABM-1';
acqcost := 40;
opscost := 20;
utilval := 20;
yrforrd := 0;
availin := 0;
rndyr1 := 0;
rndyr2 := 0;
rndyr3 := 0;
purchlm := 30;
modcost := 0;
modflag := 0;
    write(sysdata, tempdata);
sysname := 'B-2';
acqcost := 100;
opscost := 50;
utilval := 200;
yrforrd := 1;
availin := 3;
rndyr1 := 400;
rndyr2 := 700;
rndyr3 := 500;
purchlm := 20;
modcost := 0;
modflag := 0;
    write(sysdata, tempdata);
sysname := 'M-2';
acqcost := 100;
opscost := 90;
utilval := 150;
yrforrd := 2;
availin := 3;
rndyr1 := 300;
rndyr2 := 600;
rndyr3 := 300;

```

```

purchlm := 20;
modcost := 0;
modflag := 0;
    write(sysdata, tempdata);
sysname := 'F-1A';
acqcost := 150;
opscost := 80;
utilval := 80;
yrforrd := 1;
availin := 1;
rndyr1 := 200;
rndyr2 := 0;
rndyr3 := 0;
purchlm := 20;
modcost := 0;
modflag := -3;
    write(sysdata, tempdata);
sysname := 'ABM-2';
acqcost := 100;
opscost := 50;
utilval := 100;
yrforrd := 1;
availin := 2;
rndyr1 := 400;
rndyr2 := 700;
rndyr3 := 0;
purchlm := 20;
modcost := 0;
modflag := 0;
    write(sysdata, tempdata);
sysname := 'B-3';
acqcost := 200;
opscost := 80;
utilval := 200;
yrforrd := 2;
availin := 3;
rndyr1 := 800;
rndyr2 := 800;
rndyr3 := 800;
purchlm := 20;
modcost := 0;
modflag := 0;
    write(sysdata, tempdata);
sysname := 'M-1A';
acqcost := 100;
opscost := 60;
utilval := 70;
yrforrd := 1;
availin := 1;
rndyr1 := 100;
rndyr2 := 0;
rndyr3 := 0;
purchlm := 25;

```

```

modcost := 0;
modflag := -2;
  write(sysdata, tempdata);
sysname := 'M-3';
acqcost := 150;
opscost := 100;
utilval := 280;
yrforrd := 2;
availin := 3;
rndyr1 := 300;
rndyr2 := 600;
rndyr3 := 800;
purchlm := 20;
modcost := 0;
modflag := 0;
  write(sysdata, tempdata);
sysname := 'M-4';
acqcost := 100;
opscost := 80;
utilval := 160;
yrforrd := 2;
availin := 2;
rndyr1 := 800;
rndyr2 := 600;
rndyr3 := 0;
purchlm := 20;
modcost := 0;
modflag := 0;
  write(sysdata, tempdata);
sysname := 'F-2';
acqcost := 160;
opscost := 140;
utilval := 300;
yrforrd := 2;
availin := 3;
rndyr1 := 300;
rndyr2 := 1000;
rndyr3 := 1000;
purchlm := 25;
modcost := 0;
modflag := 0;
  write(sysdata, tempdata);
sysname := 'ABM-3';
acqcost := 100;
opscost := 50;
utilval := 200;
yrforrd := 2;
availin := 1;
rndyr1 := 1000;
rndyr2 := 0;
rndyr3 := 0;
purchlm := 30;
modcost := 0;

```

```
modflag := 0;  
  write(sysdata, tempdata);  
  sysname := 'F-3';  
  acqcost := 200;  
  opscost := 60;  
  utilval := 200;  
  yrforrd := 3;  
  availin := 2;  
  rndyr1 := 1000;  
  rndyr2 := 1800;  
  rndyr3 := 0;  
  purchlm := 20;  
  modcost := 0;  
  modflag := 0;  
  write(sysdata, tempdata);
```

```
end;
```

```
close(sysdata);
```

```
end.
```

```

/***** M.1 TEMPO ICAI Code *****/
/*****      TMPOICAI      *****/

```

```

prefix cycle.
prefix the.
prefix varclear.

```

```

infix for.
infix balanced.
infix war.
infix overspent.
infix overdefend.
infix budgetleft.

```

```

postfix offense.
postfix defense.
postfix force.
postfix check.
postfix paused.

```

```

/* no goal in this one goal = done. */

```

```

initialdata = [the consultation is over].

```

```

if do(loadcache year.X) and
    bestbmb = B and
    bestmsl = M and
    B > M and
    display(['The best offensive system this ',
             'year was a bomber.',nl])
then the answer for cycle X =
    'the best offensive system is a bomber'.

```

```

if bestbmb = B and
    bestmsl = M and
    M > B and
    display(['The best offensive system this ',
             'year was a missile.',nl])
then the answer for cycle X =
    'the best offensive system is a missile'.

```

```

if bestbmb = B and
    bestmsl = M and
    M = B and
    display(['The offensive systems this ',
             'year were equally effective.',nl])
then the answer for cycle X =
    'the offensive systems were equal'.

```

```

if the answer for cycle X is known and
   cycle X balanced force is complete and
   varclear X is complete
then cycle X is complete.

```

```

if cycle 1 is complete and
   cycle 2 is complete and
   cycle 3 is complete and
   cycle 4 is complete and
   cycle 5 is complete and
   cycle 6 is complete and
   cycle 7 is complete and
   cycle 8 is complete
then consultation is complete.

```

```

if consultation is complete and
   display(['The consultation is over.',nl])
then the consultation is over.

```

```

if cycle X balanced offense is complete and
   cycle X balanced defense is complete and
   cycle X war check is complete and
   cycle X overspent check is complete and
   cycle X overdefend check is complete and
   cycle X budgetleft check is complete and
   cycle X paused is known
then cycle X balanced force is complete.

```

```

if pbmbcnt = PB and
   pmslcnt = PM and
   PB >= PM and
   PB <= (2.01 * PM) and
   display(['Your offensive systems for year ',
           X, ' were balanced.',nl])
then cycle X balanced offense is complete.

```

```

if pbmbcnt = PB and
   pmslcnt = PM and
   PM >= PB and
   PM <= (2.01 * PB) and
   display(['Your offensive systems for year ',
           X, ' were balanced.',nl])
then cycle X balanced offense is complete.

```

```

if pbmbcnt = PB and
   pmslcnt = PM and
   PM > PB and
   PM > (2.01 * PB) and
   display(['Your offensive systems for year ',
           X, ' were NOT balanced.',nl,
           'This results in a penalty applied ',

```

```

        'to your total utils which',nl,
        'means that your systems were less',
        'effective than you anticipated',nl,nl))
then cycle X balanced offense is complete.

if pbmbcnt = PB and
  pmslcnt = PM and
  PB > PM and
  PB > (2.01 * PM) and
  display(['Your offensive systems for year ',
    X,' were NOT balanced.',nl,
    'This results in a penalty applied',
    'to your total utils which',nl,
    'means that your systems were less ',
    'effective than you anticipated',nl,nl))
then cycle X balanced offense is complete.

if pfttrcnt = PF and
  pabmcnt = PA and
  PF > PA and
  PF > (2.01 * PA) and
  display(['Your defensive systems for year ',
    X,' were NOT balanced.',nl,
    'This results in a penalty applied ',
    'to your total utils which',nl,
    'means that your systems were less ',
    'effective than you anticipated',nl,nl))
then cycle X balanced defense is complete.

if pfttrcnt = PF and
  patmcnt = PA and
  PF >= PA and
  PF <= (2.01 * PA) and
  display(['Your defensive systems for year ',
    X,' were balanced.',nl])
then cycle X balanced defense is complete.

if pfttrcnt = PF and
  patmcnt = PA and
  PA > PF and
  PA > (2.01 * PF) and
  display(['Your defensive systems for year ',
    X,' were NOT balanced.',nl,
    'This results in a penalty applied to ',
    'your total utils which',nl,
    'means that your systems were less ',
    'effective than you anticipated',nl,nl))
then cycle X balanced defense is complete.

if pfttrcnt = PF and
  pabmcnt = PA and
  PA >= PF and
  PA <= (2.01 * PF) and

```

```

        display(['Your defensive systems for year ',
                X,' were balanced.',nl])
    then cycle X balanced defense is complete.

if overspent = OV and
    OV = 0
    then cycle X overspent check is complete.

if overspent = OV and
    OV = 1 and
        display(['You overspent your budget this year. ',
                'This is a critical',nl,
                'mistake as it will result in a reduced ',
                'budget the next',nl,
                'year as well as causing you to lose ',
                'some of your systems.',nl,
                'The systems lost may not be ones you ',
                'would have wished to',nl,
                'lose, they are picked by the ',
                'computer at random.',nl,nl])
    then cycle X overspent check is complete.

if overspent = OV and
    OV = 2 and
        display(['You overspent your budget again. ',
                'This is a critical',nl,
                'error and it is extremely important ',
                'that you ensure that',nl,
                'you remain within budget each year.',nl,nl])
    then cycle X overspent check is complete.

if overspent = OV and
    OV = 3 and
        display(['This is the third year that you ',
                'overspent your budget.',nl,
                'This could indicate that you do not ',
                'quite understand all',nl,
                'of the variables that go into the ',
                'calculating of the budget',nl,
                'spent each year. One of the most ',
                'often forgotten items is',nl,
                'the active forces operations costs. ',
                'If you do not scrap',nl,
                'a system the computer will deduct ',
                'operations cost for it.',nl,
                'You must also remember to add in any ',
                'R&D costs as well as',nl,
                'the cost of purchasing any new systems. ',
                'Finally, remember',nl,
                'that each type of intel report
                costs you $100.',nl,nl]) and

```

```

    ovspent is complete
then cycle X overspent check is complete.

```

```

if overspent = OV and
  OV = 4 and
  display(['You have once again overspent your budget.',
    '  This could',nl,
    'undoubtedly be a deciding factor ',
    'in the game.',nl,nl])
then cycle X overspent check is complete.

```

```

if overspent = OV and
  OV > 4 and
  display(['You have once again overspent ',
    'your budget.',nl,nl])
then cycle X overspent check is complete.

```

```

if ovready = yes and
  display(['If you have the following, what ',
    'is your budget cost?',nl,nl,
    'B1 --> Acq cost = 100,  Ops cost = 50',nl,
    'F1 --> Acq cost = 50,   Ops cost = 25',nl,nl,
    'Your current inventory contains ',
    '5 Bls and 2 Fls.',nl,nl,
    'You wish to purchase an additional',
    '5 Bls and 4 Fls.',nl,nl,
    'You want an intelligence report ',
    'on offensive R&D.',nl,nl,
    'You wish to fund R&D on a B2, cost 200 ',
    'for 1st year.',nl,nl]) and
  ovcost = 1300 and
  display(['Correct',nl])
then ovspent is complete.

```

```

if ovready = yes and
  display(['Incorrect',nl,
    'The correct answer is 1300 which is ',
    'obtained as follows:',nl,nl,
    'Current Ops costs:',nl,
    '  5 Bls @ 50 = 250',nl,
    '  2 Fls @ 25 = 50',nl,
    'Acquisition costs:',nl,
    '  5 Bls @ 100 = 500',nl,
    '  4 Fls @ 50 = 200',nl,
    'Intel report: 1 @ 100 = 100',nl,
    'R&D:          1 @ 200 = 200',nl,
    '-----',nl,
    'TOTAL COST          1300',nl])
then ovspent is complete.

```

```

if waryear = WY and
  WY = 0
  then cycle X war check is complete.

if waryear = WY and
  WY = 1 and
  display(['There was a war this year which you LOST.',nl,
    'If you recall, losing a war will ',
    'reduce your budget by $800',nl,
    'while only reducing you opponents by 200. ',
    'This means that',nl,
    'your opponent will have more money ',
    'to work with next year',nl,
    'and thus have a better chance of ',
    'maintaining an advantage.',nl,nl])
  then cycle X war check is complete.

if waryear = WY and
  WY = 2 and
  display(['There was a war this year which you WON.',nl,
    'This means that your opponent had ',
    'its budget reduced',nl,
    'by $800 while yours was only reduced ',
    'by $200. This',nl,
    'gives you a chance to maintain your
    advantage, provided',nl,
    'you make no mistakes in the future.',nl,nl])
  then cycle X war check is complete.

if waryear = WY and
  WY = 3 and
  display(['There was a war this year which ',
    'ended in a draw.',nl,
    'While this meant your budget was ',
    'reduced by $400',nl,
    'it should not have adversely affected',
    'you since your',nl,
    'opponents budget was reduced by ',
    'the same amount.',nl,nl])
  then cycle X war check is complete.

if overdefend = OD and
  OD = 0
  then cycle X overdefend check is complete.

if overdefend = OD and
  OD = 1 and
  display(['You overdefended this year. ',
    'This means that you spent',nl,
    'more money on defense than you ',
    'needed to. Money spent',nl,
    'in this manner is referred to as ',

```

```

        "Futile utils" since you',nl,
        'receive no benefits from any extra ',
        'defensive utils (they',nl,
        'are wasted.',nl,nl))
    then cycle X overdefend check is complete.

if budgetleft = BL and
    BL = 0
    then cycle X budgetleft check is complete.

if budgetleft = BL and
    BL = 1 and
    display(['You had more than $100 dollars left ',
            'in your budget at the',nl,
            'end of this year. You should always ',
            'try to spend your entire',nl,
            'budget even if it means funding ',
            'a lesser system.',nl,nl])
    then cycle X budgetleft check is complete.

if do(reset)
    then varclear X is complete.

question(cycle X paused) = [nl,
    'Hit any character, a period, and the enter',
    ' key when ready to continue.',nl].

question(ovready) = ['Lets run through an example. ',
    'Type yes when ready.',nl].

question(ovcost) = ['What is the years ',
    'total budget cost?',nl].
    legalvals(ovcost) = integer.

```

Appendix C
(Sample game turn)

Start of Year 1
Your Budget is \$ 9840
The estimated budget for next year is \$ 9940

SYSTEM	INVEN	PER UNIT				TOTAL			PURCH LIMIT	MODI COST
		AQCOST	OPCOST	UTILS	OPCOST	UTILS	OPCOST	UTILS		
B-1	40	40	20	20	800	800	30	0		
N-1	60	100	30	30	1800	1800	20	30		
	Offensive Total				2600					
F-1	20	100	40	70	800	1400	30	60		
ARM-1	100	30	10	20	1000	2000	30	0		
	Defensive Total				1800					
Total present operating cost					4400					

(enter an H for help or press return to continue)

Proposed R&D Programs

All of the following are expected values except year 1 R&D cost
PER UNIT

SYSTEM	AVAILABLE IN	AQCOST	OPCOST	UTILS	YEAR 1	YEAR 2	YEAR 3	PURCH LIMIT
B-2	3yrs	120	60	100	490	830	770	20
F-1A	1yrs	160	80	100	270	0	0	20
ABM-2	2yrs	60	50	70	380	950	0	20
M-1A	1yrs	70	60	70	70	0	0	25

REMEMBER---- Systems may be bought the same year final R&D costs are paid
(enter an H for help or press a return to continue)

----- Intelligence report -----

Intelligence reports indicate chance of war this year = 14 %

enter an H for help or press return to continue

Player Function Menu

1. Reprint the reports.
2. Enter this years orders.
3. Perform weapon system analysis.
4. Finished, execute orders and proceed to next year.

Please enter the number of the option you wish to perform.
(or type H for help)

3

Enter the name of the weapon system. B-1
Enter the number of years to run the analysis. 10
Enter the number of years of R&D left. 0
Enter the system acquisition cost. 40
Enter the system yearly operation cost. 20
Enter the system util value. 20
Enter the 1st year R&D cost (enter a 0 if none). 0
Enter the 2nd year R&D cost. 0
Enter the 3rd year R&D cost. 0
Enter the purchase limit. 30

The utils/\$ percentage for the B-1 is 84 %

press return to continue

Player Function Menu

1. Reprint the reports.
2. Enter this years orders.
3. Perform weapon system analysis.
4. Finished, execute orders and proceed to next year.

Please enter the number of the option you wish to perform.
(or type H for help)

3

Enter the name of the weapon system. B-2
Enter the number of years to run the analysis. 10
Enter the number of years of R&D left. 3
Enter the system acquisition cost. 120
Enter the system yearly operation cost. 60
Enter the system util value. 100
Enter the 1st year R&D cost (enter a 0 if none). 490
Enter the 2nd year R&D cost. 830
Enter the 3rd year R&D cost. 770
Enter the purchase limit. 20

The utils/\$ percentage for the B-2 is 127 %

press return to continue

For the B-1 weapon system, you currently have 40 in your inventory.
How many do you wish to BUY? 10

How many do you wish to SCRAP? 0

For the M-1 weapon system, you currently have 60 in your inventory.
How many do you wish to BUY? 0

How many do you wish to SCRAP? 0

For the F-1 weapon system, you currently have 20 in your inventory.
How many do you wish to BUY? 10

How many do you wish to SCRAP? 0

For the ABM-1 weapon system, you currently have 100 in your inventory.
How many do you wish to BUY? 0

How many do you wish to SCRAP? 10

Please review the following to see if it is correct

Weapon system	BUY	SCRAP	MODIFY	TOTAL
B-1	10	0	0	50
M-1	0	0	0	60
F-1	10	0	0	30
ABM-1	0	10	0	90

Is this correct (Y/N)?

y

Do you want to fund research for B-2 ? (Y/N) y
 Do you want to fund research for F-1A ? (Y/N) n
 Do you want to fund research for ABM-2 ? (Y/N) n
 Do you want to fund research for M-1A ? (Y/N) y
 How many M-1A systems do you want to buy? 2
 How many M-1A systems do you want to modify? 0

Please review this information. Is it correct? (Y/N) y

Do you want any intelligence reports? (Y/N) y

At 100 \$ per intelligence report, do you want a report on

Offensive systems ? (Y/N) y

Defensive systems ? y

Offensive R & D ? y

Defensive R & D ? y

Player Function Menu

1. Reprint the reports.
2. Enter this years orders.
3. Perform weapon system analysis.
4. Finished, execute orders and proceed to next year.

Please enter the number of the option you wish to perform.
(or type H for help)

4new research budget = 490
new research budget = 70
you spent this much this year 6800

End of year intelligence reports indicated that
you were ahead in the "Arms Race" for this year.

hit a return to continue

VITA

Captain Gregory B. White was born January 29, 1958 in Washington D.C. Upon graduation in 1975 from Oxon Hill High School in Oxon Hill, Maryland he entered the Brigham Young University. He graduated with a Bachelor of Science degree in Computer Science in 1980 and was commissioned a Reserve Second Lieutenant in the United States Air Force.

After graduation, Captain White was assigned to the 3900th Computer Services Squadron, Offutt AFB, Nebraska where he worked as a Systems Analyst for the Joint Chiefs of Staff War Plans Programming Directorate. While at Offutt, he received his Regular Commission and was sent to Squadron Officers School from whence he graduated in March of 1984. In May of 1984 Captain White entered the Masters program at the Air Force Institute of Technology, Wright Patterson AFB, Ohio.

Captain White is married to the former Charlan G. Cook of Moriarity, New Mexico. They have three children: Sandra, Heather, and Gregory M. White. Captain White is a member of Tau Beta Pi.

Permanent Address: 4216 Dorris Rd
Irving, TX 75038

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

AD-A172-782

REPORT DOCUMENTATION PAGE

1a REPORT SECURITY CLASSIFICATION UNCLASSIFIED			1b RESTRICTIVE MARKINGS		
2a SECURITY CLASSIFICATION AUTHORITY			3 DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited.		
2b DECLASSIFICATION DOWNGRADING SCHEDULE					
4 PERFORMING ORGANIZATION REPORT NUMBER(S) AFIT/GCS/ENG/86M-1			5. MONITORING ORGANIZATION REPORT NUMBER(S)		
6a NAME OF PERFORMING ORGANIZATION School of Engineering		6b. OFFICE SYMBOL (If applicable) AFIT/ENG	7a. NAME OF MONITORING ORGANIZATION		
6c ADDRESS (City, State and ZIP Code) Air Force Institute of Technology Wright-Patterson AFB, Ohio 45433			7b. ADDRESS (City, State and ZIP Code)		
8a NAME OF FUNDING/SPONSORING ORGANIZATION Squadron Officers School		8b OFFICE SYMBOL (If applicable) SOS/EDCX	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER		
8c ADDRESS (City, State and ZIP Code) SOS/EDCX Maxwell AFB, AL 36112-5582			10 SOURCE OF FUNDING NOS.		
11 TITLE (Include Security Classification) See Box 19			PROGRAM ELEMENT NO.	PROJECT NO	TASK NO
12 PERSONAL AUTHOR(S) Gregory B. White, B.S., Capt, USAF			WORK UNIT NO		
13a TYPE OF REPORT MS Thesis		13b TIME COVERED FROM _____ TO _____	14. DATE OF REPORT (Yr., Mo., Day) 1986 March		15 PAGE COUNT 154
16 SUPPLEMENTARY NOTATION					
17 COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)		
FIELD	GROUP	SUB GR	Artificial Intelligence, Expert System, Military Science, War Gaming		
06	04				
15	07				
19 ABSTRACT (Continue on reverse if necessary and identify by block number)					
Title: Artificial Intelligence Concepts and the War Gaming Environment: A Case Study Using the TEMPO War Game					
Thesis Chairman: Steve Cross, Major, USAF Assistant Professor of Electrical Engineering					
Approved for public release; LAW, AFR 150-1, BY <i>[Signature]</i> 9 May 86 STEVE E. WOLVER Dean for Research and Professional Development Air Force Institute of Technology (AFIT) Wright-Patterson AFB OH 45433					
20 DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT ☐ DTIC USERS ☐			21 ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED		
22a NAME OF RESPONSIBLE INDIVIDUAL Stephen E. Cross, Major, USAF			22b TELEPHONE NUMBER (Include Area Code) 513-255-3576		22c OFFICE SYMBOL AFIT/ENG

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

war gaming has long been an accepted practice in the mental preparation of officers for war. With the introduction of computers, the games have become increasingly sophisticated yet most current war games are either too slow, not realistic, or use the computer as a referee only and not as a player. An approach is discussed in the context of TEMPO, a force planning war game currently used by the Air Force at its Squadron Officers School. This thesis involved the development of a version of TEMPO in which a computer expert system takes the place of one of the players, and an intelligent computer aided instruction system that takes the place of the section leader. The system is implemented on a microcomputer allowing its use in professional military education seminar courses.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

END

11-86

DTIC