Proceedings of the 1980 Winter Simulation Conference
T.I. Ören, C.M. Shub, P.F. Roth (eds.)

THE MISSION EFFECTIVENESS PROGRAM: AN ANALYST'S VIEWPOINT

Wilhelm H. Bortels, D.Sc.
Naval Underwater Systems Center
New London Laboratory
New London, Connecticut

The Mission Effectiveness Assessment Program simulation framework (MEAP) is being developed to provide Navy decision makers with a means to quantify the military value of naval combat units and combat unit elements in applications to realistic naval warfare engagements. This paper presents an overview of MEAP from a Naval Analyst (user) viewpoint.

Keywords:
Simulation
Naval Analysis
Mission Effectiveness Assessment
MEAP
Submarine Warfare
Naval Warfare

1. INTRODUCTION

1.1 Overview

At present most submarine mission effectiveness studies within the Navy are performed by using the SUBWEM, APSUB, and SIM-II simulation models. These are versatile analytical tools, which have been employed for various studies and are being used increasingly. The success of these models has led to more detailed, extensive, and complex questions being asked by OPNAV and DDR&E. The above models are limited by two main factors: the physical limitations in the computer program; and the time and cost needed to perform a given simulation run. These and other considerations have led to the requirement of a hardware and software framework which can handle significantly more complex scenarios and should be more efficient and responsive then existing simulation tools. This framework is to be known as the Mission Effectiveness Assessment Program (MEAP).

1.2 Scope of the Program

MEAP provides Navy acquisition managers with a systematic approach for assessing the military effectiveness of combat units and combat unit elements through determining the probability of mission success in realistic naval warfare engagements. In this context, combat units are the aircraft, surface ships, or submarines which individually or collectively are employed in the mission. Combat unit elements are the subsystems of the unit that contribute to the military value of the mission. Elements may be hardware systems such as sonar, fire control, weapons, communications, etc.; mission functions or doctrine such as command, control, or tactics; or separate systems such as SOSUS, which, although not physical elements of the combat unit, may contribute to the overall military effectiveness of the unit in a given mission scenario.

MEAP represents a systems approach to the measurement of mission effectiveness. It is concerned with all combat unit elements and functions in a given problem, such as platform dynamics, sensors, weapons, command, control, communications, tactics, environment, countermeasures, damage assessment, threat, operating procedures, etc.. The methodology used provides the means to approach the complex determination of mission effectiveness systematically. It places the Navy acquisition manager and the naval analyst in direct communication during assessment of the problem. The MEAP simulation framework is used to examine the problem, defined and bounded by the Navy acquisition manager and the naval analyst, and to account for all parameters of the engagement and for processing of the data to obtain quantitative results.

1.3 Software Control and Validation

Development of MEAP is conducted with rigid software control including configuration control and quality assurance of all software modules. Thus, by the time the naval analyst uses MEAP, coding errors are virtually eliminated and the documentation reflects the actual design. Validation of the initial version of MEAP was carried out by the analysts by 1) analytically determine the correctness of simulation outputs of scenarios specifically designed to test various key aspects, such as the final position of platforms after course and speed changes, 2) comparing the simulation outputs for a typical naval study scenario which was programmed both in MEAP and SIM II; the latter having been validated by its use over many years, and 3) simulating an at-sea submarine exercise and comparing the simulation results with the actual at-sea results. Preliminary results indicate an acceptable level of performance.

2. MEAP APPLICATION AND CAPABILITY

MEAP is designed to have the capability to serve a variety of users and to provide for program growth. Provisions are made to allow other computer models, such as models simulating the detailed performance of a radar or sonar model and, possibly, other hardware systems to provide input to or accept output from MEAP. An interactive mode, with a man-in-the-loop, is a desirable and realizable application.

Some examples of the MEAP capabilities include:

1. Quantifying the probability of mission success (military value) of combat units and combat unit elements in realistic naval warfare engagements.
2. Assessing the impact on mission success (military value) of changes in the performance characteristics of any part of the combat unit.
3. Determining the military effectiveness for alternative configurations of combat units and combat unit elements to aid acquisition managers in prioritizing new developments and Fleet introduction of new developments.
4. Assessing the effectiveness of standard tactics in alternative environments against current or projected threats.
5. Assessing proposed tactics to select those showing prospects of improving mission success for trial at sea.
6. Assessing defensive tactics to guide the development of those tactics that would minimize the probability of enemy mission success.
7. Simulating operational conditions and situations prohibited by peacetime safety rules.

Additional capabilities exist for supporting studies concerned with operating orders, war plan development, force multipliers, conceptual system assessment and system engineering.

3. MEAP SIMULATION TOOL STRUCTURE

3.1 Model Synthesis

The MEAP simulation model synthesis is depicted in Fig. 1. The synthesis begins with a task statement, a scenario and/or Letter of Intent (LOI) along with operation orders (OPORDERS). From this, the specific input data, output data (measures of effectiveness (MOEs)), combat system unit model, and command and control (C2) model requirements can be obtained.

As can be seen from the figure, MEAP's simulation tool is envisioned as a framework for constructing a task specific combat system unit model, a C2 model, and a data base.

The unit model and its modules may be thought of as being static in the sense that their basic structure does not generally change from simulation task to simulation task. The NA, however, can choose a module of specific characteristics, for a specific simulation task. Scenario dependent aspects are obtained through input data supplied to the data base. The C2 model, however, may be thought of as being dynamic in the sense that it must be specifically constructed to meet each simulation task objective. This is the model that the NA develops. The data base format remains fixed throughout the various simulation tasks but the input and output data are task dependent; the NA provides the task specific data. It should be noted that output data falls into two general catagories: a standard output, and a task specific one. The standard output can be collected routinely, and includes such data as a chronological log and a track log; the task specific output is specified by the NA in the C2 model, and is usually concerned with data needed for the evaluation of specific MOEs. In addition to these, there is also a simulation log, and various displays which can be used to observe the progress of the simulation during the simulation phase.

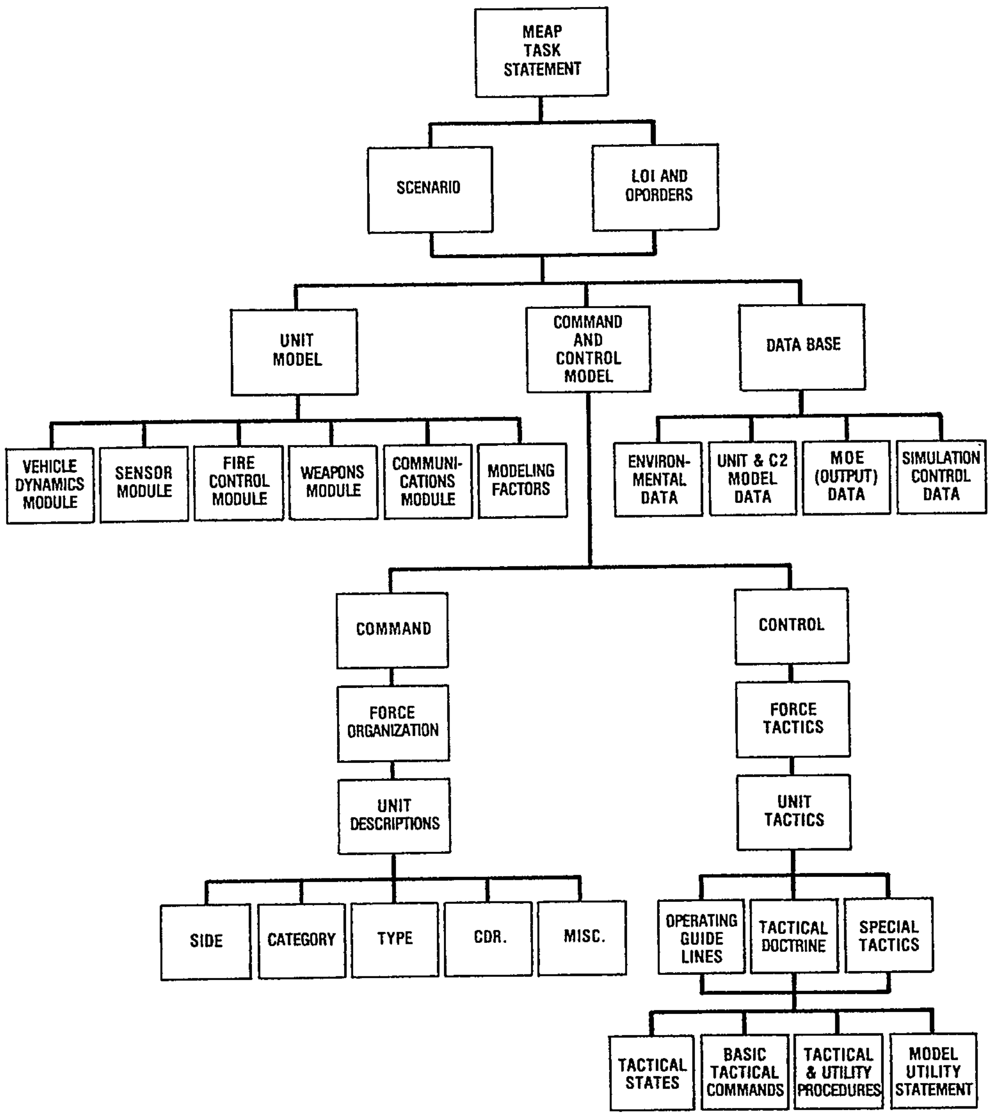


Figure 1. MEAP Model Synthesis

The combat system unit model consists of several modules, sometimes referred to as functional models, which describe the functional characteristics and behavior of the vehicle, sensors, fire control mechanism, weapons, and communications equipment, and provides other factors which may be necessary to construct the MEAP simulation task.

The C2 model, sometimes referred to as the tactical model, or just tactics, describes the command and control aspects of the simulation task. The command structure describes the force organization and its individual units - their side (blue/orange), category (submarine, surface ship, aircraft), type (SSN, SSBN, etc.) the force command level, and other unit specific parameters. The control portion describes the overall force tactics and the tactics for each individual unit. Unit tactics is based on operating guidelines, tactical doctrines, or is designed specifically for the simulation task as directed by the OPORDER or scenario. Unit tactics will be composed of tactical states, basic tactical commands, tactical and utility procedures, and model utility statements.

3.2 Concepts for Modeling Unit C2

As previously described, unit C2 modeling is dictated by the specific OPORDERS or scenario defined assignments for that unit.

The modeling of each platform-threat encounter (Fig. 2) is approached from the point of view of an observer (e.g., the commanding officer) on that platform. The platform is then envisioned as being the focus of the threat environment. This environment may change to reflect accompanying friendly forces and other platforms. Threat information passes to the observer only through the means available on that platform, such as the sensors (including visual), communications, and other reporting channels.

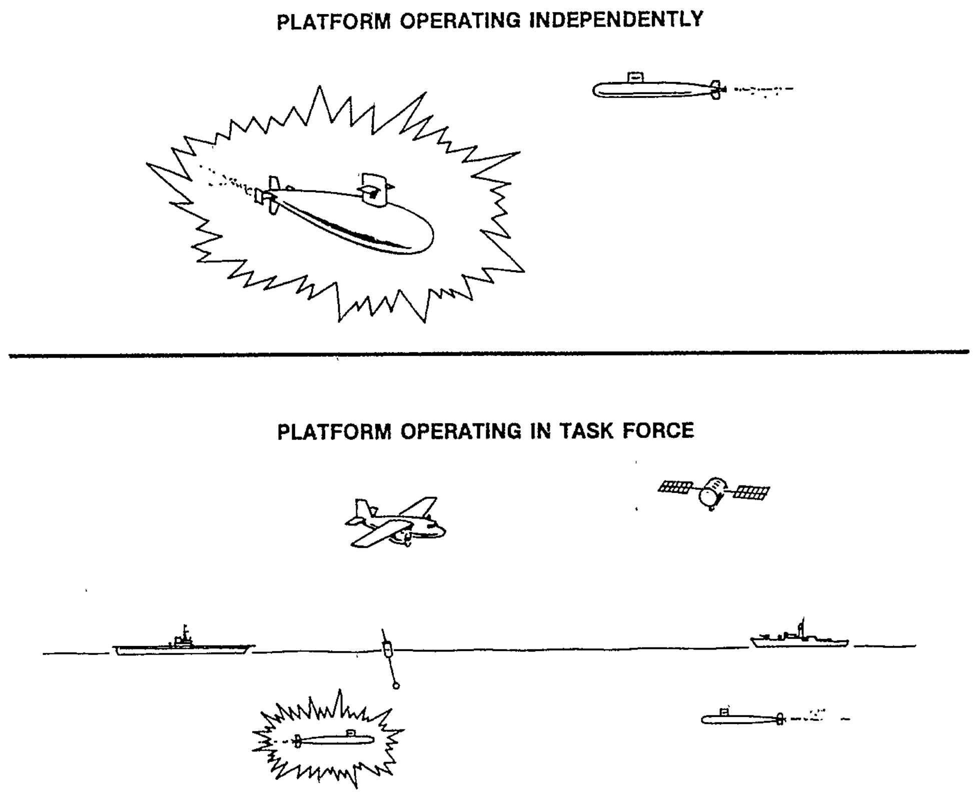

PLATFORM OPERATING INDEPENDENTLY
PLATFORM OPERATING IN TASK FORCE

Figure 2. Modeling Concept of Individual Platform Tactics

An advantage of this approach is that it allows the simulation to be straightforward yet realistic for even very complex scenarios, since the NA need specify only the tactical responses to the information received. Naturally, a more complex threat environment may produce more information to be processed and, perhaps, elicit additional tactical logic to cope with the situation.

3.3 Functional Flow of C2 Model

At the beginning of a simulation run, an individual platform will be in its initial state (Fig. 3). A set of tactics defined by the OPORDER is associated with this initial state. A submarine conducting a barrier patrol or a high-value unit in transit are examples of initial states. The platform will remain in its initial state unless and until some interaction or event occurs that causes a change of state. The tactics executed in the new state are generally dictated by the scenario or the mission objective of that platform. For example, an event CONTACT may cause the platform to change from its initial state to a state in which the platform executes tactics in an attempt to classify the contact. In general, there may be any number of events which cause a platform to change states.

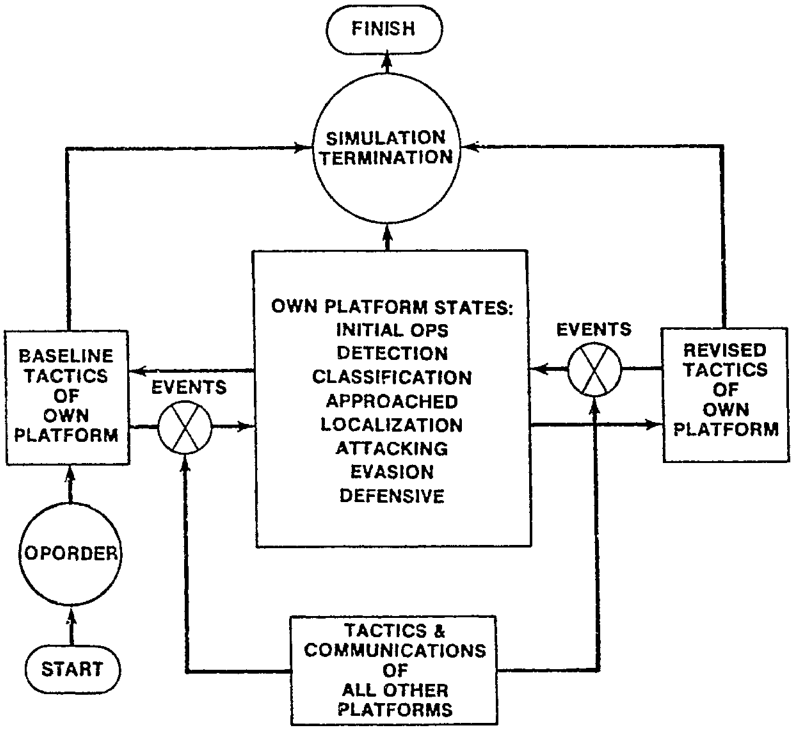


Figure 3. Functional Flow of Tactics Model

The simulation may be terminated if the initial tactics produce no interactions within some specified time, if the tactics of a new state produce no further interaction, or if the state has been selected as terminal (i.e., the mission objective has been achieved).

4. MEAP TACTICS LANGUAGE

4.1 Meap Tactics Language Philosophy

The tactics language provides the means for the analyst to translate a scenario into detailed tactics for platforms in the MEAP simulation. This language not only permits a degree of realism closely approximating the real-life situation, but is efficient for modeling tactics in a complex scenario.

The tactics language consists of a minimal set of functions which can be chained to form more complex constructs. The language at the most basic level consists of statements or routines which specify that some action be taken (or not taken) by a specific sensor, weapon, platform, communications device, etc. i.e., energize radar, launch weapon, change speed, send message, etc.. These functions represent the end points, or smallest branches in any given set of tactics. With this minimal set of commands, all command and control decision processes or tactics for all the elements of the simulation are executed. Although such commands are sufficient to "program" the tactics for any platform ease of implementation suffers unless higher order tactics terms or constructs are available to the analyst.

Although tactical decision processes can be implemented by a basic set of instruction commands as described, it is a tedious job at best to account for all the elemental tactics statements on every platform for each situation in every simulation. Fortunately, complex C2 processes are a pyramiding set of elementary constructs that result in successively more complex C2 procedures or tactics. Also, many complex tactics represent basic U.S. forces tactical doctrines for given circumstances. Once constructed for one scenario, many complex tactics are usable, with minor changes, in many other scenarios. For example, the tactic SPRINT and DRIFT always implies the same kind of action by a platform even though the values of some variables (i.e., sprint speed) may change. Thus, to profit from previous MEAP simulation runs, and to prevent the requirement to program at the most basic level for each simulation, it is important to have the capability to store complex tactics for future use. Thus, as the number of complete simulation runs grows, a valuable library of tactical routines also grows. The analyst can then select proven tactics routines to cut down on the development time required for future simulation runs.

Modeling with the tactics language relies on the definition of "states" and "procedures". The states are intended to serve as situation indicators, and form the basis for selecting specific tactical procedure or set of procedures. The procedures are of varying levels of complexity.

States are defined as being true when some precondition or conditions are met. Thus, a state can be conceived of as a threshold (positive or negative) that is exceeded when certain criteria are met. By defining a basic set of states to be mutually exclusive, the procedures to be executed whenever a particular state exists can be made independent of the procedure invoked whenever another state exists.

The concept of states and procedures allows a shorthand notation in which cause and effect are programmed with a minimal amount of effort. When this is combined with the previously described complex tactics library, a complex maneuver, search plan, etc., may be implemented with a minimal amount of user setup time.

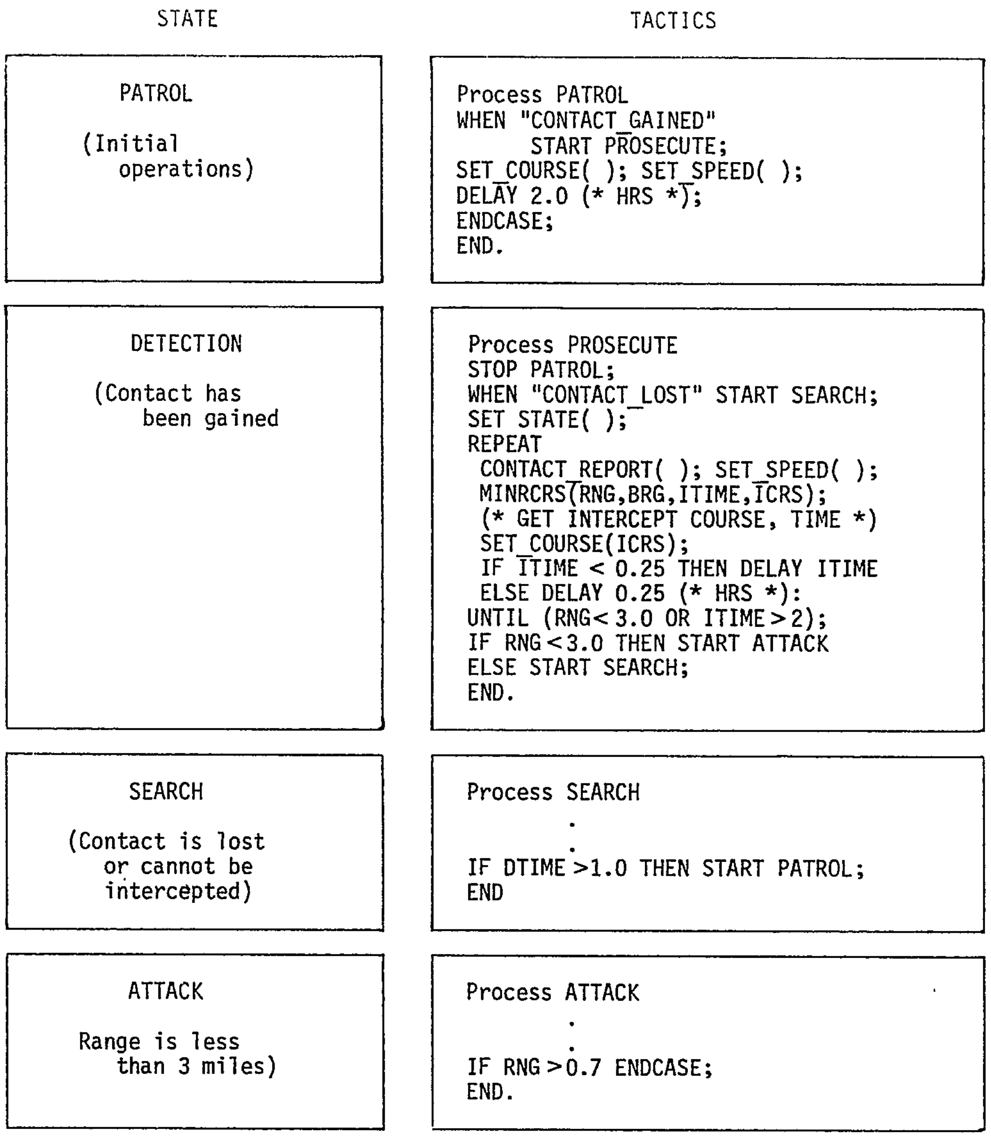
The concept of states is also valuable at postsimulation time when statistics are gathered. Since the criteria necessary to achieve each given state (for each simulation element) are known to the analyst, a shorthand notation of the simulation results can be gathered. Thus, the statistics might be more easily displayed and understood by the nonanalyst. Outputs will be provided in such a form that a sponsor interested in such statistics such as the percentage of time spent in searching, attacking, transiting, etc., or the elapsed time between detection and kill of a target, can obtain this information in a minimum amount of time.

Procedures are designed as modules. An important benefit of this design is the ease with which modules may be replaced, added to or deleted from a set of platform tactics.

As an example, consider a ship patrolling an area while searching for hostile ships (Fig. 4). The friendly ship's orders detail certain actions to be taken when any other units are encountered. These actions are detailed in a procedure called PROSECUTE. The orders call for the friendly unit to execute the orders contained in PROSECUTE each time contact is gained on another ship. A tactical DETECTION STATE is then defined to be true every time the ship (1) gains as ESM contact, (2) gains a radar contact, or (3) gains a sonar contact, etc.. The analyst can then invoke the execution of the procedure PROSECUTE every time DETECTION STATE is true. This procedure contains the vehicle dynamics, sensor employment scheme, communication plan, etc., for the platform.

An additional advantage of the states and procedures is that the analyst can implement a tactics model at a level of detail consistent with the scenario. This is especially useful in the initial stages of developing the model logic. In addition, procedures are executed only after control is transferred to them as a result of some criteria, not because they are arranged in some specific order. This makes it easier for the analyst to choose a procedure from the tactics library and insert it into the tactic being constructed for a given platform.

In summary, the tactics language is the means to implement complex command and control procedures for platform, sensor, and weapon employment and tactics. Tactics performance, as well as communication, weapon, sensor, and platform performance may be evaluated. The MEAP tactics language provides the analyst with great flexibility in describing tactics, and with freedom from much repetition and tedium in setting up and implementing those tactics.



```
Process PATROL
WHEN "CONTACT_GAINED"
      START PROSECUTE;
SET_COURSE( ); SET_SPEED( );
DELAY 2.0 (* HRS *);
ENDCASE;
END.
```

```
Process PROSECUTE
STOP PATROL;
WHEN "CONTACT_LOST" START SEARCH;
SET_STATE( );
REPEAT
 CONTACT_REPORT( ); SET_SPEED( );
 MINRCRS(RNG,BRG,ITIME,ICRS);
  (* GET INTERCEPT COURSE, TIME *)
 SET_COURSE(ICRS);
 IF ITIME < 0.25 THEN DELAY ITIME
 ELSE DELAY 0.25 (* HRS *):
UNTIL (RNG< 3.0 OR ITIME>2);
IF RNG<3.0 THEN START ATTACK
ELSE START SEARCH;
END.
```

```
Process SEARCH
        .
        .
IF DTIME >1.0 THEN START PATROL;
END
```

```
Process ATTACK
        .
        .
IF RNG >0.7 ENDCASE;
END.
```

Figure 4. Example of States and Related Tactics Process

4.2 Tactics Language Constructs

The MEAP tactics language consists of a high level general purpose computer language and specially formulated tactics or C2 model constructs. The general purpose computer language has the capability to: (1) perform arithmetic operations, (2) make logical comparisons, (3) have assignment statements, (4) interpret character strings and (5) permit the output of selected information.

The C2 model for each platform is created by the use of four sets or types of constructs. These are: tactical states, basic tactical commands, tactical and utility procedures, and model utility statements.

Tactical States

Tactical states are a conceptual set of all possible situations that may occur in a commander's mission, as described in the OPORDER. In addition to the initial state, other specific states may result from having made a detection; classified a contact; completed an approach to a point, area, or contact; localized a contact; achieved an attack position; received a message with information or orders; successfully evaded an opponent; been attacked; been damaged; completed the mission objective or any other NA-specified situation. The particular set of states for each platform in the scenario is derived manually from the OPORDER by an NA. The events that invoke a particular state are associated with that state for each state in the set. The states and related events are used to modify the tactical logic flow used in the simulation. Control of the actions of a platform while it is in a given state is maintained by the use of basic tactical commands and/or basic tactical procedures.

Tactical states describe a platform's current situation. They are the result of one or more significant events having occurred. Platforms reach a given state whenever they have reached an assigned position, made a classification, received a threat, etc., and require a decision to be made. States are thus the "nodes" in the tactical decision tree.

Events are made up of two types, SYSTEM EVENTS and LOCAL EVENTS. The former is often an input to the latter, with the latter being the event or events causing the platform to enter a new state.

System Events

System events are declared by the functional model of MEAP. The criteria for these events to occur are evaluated in each functional module. The occurrence of these events may or may not cause a particular platform to enter a specific state. The determination as to whether it does is made after evaluating each event against a number of factors defined in the C2 model, based on the scenario being modeled. Basic events, with typical modifiers given in brackets, include the following:
DETECTION (gained, held, lost)
CLASSIFICATION (category, side, type)
MESSAGE (intercepted, received)
LOCATION (reached position, area, range)
DAMAGE (report, (to) component)
WEAPON LAUNCH (detected preparation, actual)
ELECTRONIC WARFARE (threat warning, interference, deception)

Local Events

Local events are declared by the C2 model of MEAP and are local to a specific platform. The criteria for these events to occur is governed by the particular scenario being modeled. Simulation MOE outputs are collected based on these events. An example of a local event might be TARGET ACQUISITION. This event causes the platform to enter an associated state whenever the target has been acquired within weapon acquisition parameters. Another example of a local event might be GAINED CONTACT. The conditions for this event to occur depend, first, on the fact that the system event DETECTION has occurred and, second, that it has met additional criteria, such as the platform's signal return exceeded a specified level over a specified time.

Basic Tactical Commands

Basic tactical commands are a set of: (a) maneuvering commands, such as SET COURSE and SET SPEED; (b) operational commands such as ACTIVATE or DEACTIVATE a sensor; and (c) status commands which are utilized to obtain current data on own platform's sensor contact and own platform's current operational parameters, such as speed, course, etc..

The following set of tactical commands is an initial set of commands. As MEAP evolves it is conceivable that this set may be expanded to encompass other more specialized or detailed operations.

	Command	Purpose
1.	SET COURSE	Set Course
2.	SET SPEED	Set Speed
3.	SET DEPTH	Set Depth
4.	SET ALTITUDE	Set Altitude
5.	ACTIVATE	Activates sensors and devices
6.	DEACTIVATE	Deactivates sensors and devices
7.	LAUNCH	Launches platform or device
8.	RECOVER	Recover platform or device
9.	FIRE	Fire aimed (dumb) weapons
10.	SEND MESSAGE	Send message
11.	RECEIVE MESSAGE	Receive message
12.	SET STATE	Set a tactical state
13.	GET STATUS	Get status of any platform parameter, including the tactical state
14.	REPORT	Obtain contact, damage, ESM report
15.	DELAY	Delay execution
16.	SET THRESHOLD	Set the threshold value for which a system event will be declared
17.	DEFINE LOCAL EVENT	Defines the criteria for a local (platform) event

Tactical and Utility Procedures

Tactical procedures are built around several basic tactical commands. They form the building blocks of more complex tactical models. Examples of these tactical procedures could be procedures such as CPA, EKELUND, MINRANGECRS, MAXRANGECRS, LAG and LEAD. In general, basic tactical procedures can be used in any number of tactical decision processes and are similar to subroutines or procedures in higher level computer languages. Utility procedures are also used in the formulation of tactical models but do not represent tactical routines; rather, they provide the NA with tools to affect random distributions, perturb true values to simulate real world observations, and perform other frequently used analytical functions.

Tactical and utility procedures will be used in conjunction with basic tactical commands and model utility statements to design more complex C2 models. Generally, this will result in a given platform reaching a more advanced state, as for example from detection to classification. The combined tactics of all individual platforms describe the overall mission model. The procedures are developed as part of the evolving MEAP simulation tool. Other examples of the set are listed below:

Name	Purpose
COLLISION COURSE	Derive intercept course and time
CLOSE	Derive minimum speed intercept parameters
SPIRAL	Derive course, speed, and time for expanding search
TURN	Derive course, speed, and time for search to and around datum
DATUMXY	Find x,y coordinates of datum
DATUMRB	Find range, bearing to datum
SORT	Sort (target) list
EVALFN	Evaluate function
EXPON	Provide random number from exponential distribution
UNIFORM	Provide random number from uniform distribution
NORM	Provide random number from normal distribution
EKELUND	Provide TMA solution
ASPECT	Compute aspect of target
RELBRG	Compute relative BRG of target
RN1-RN6	Provide uniform random number generators

<u>Model Utility Statements</u>
The model utility statements are used (1) to aid in the construction of the tactics model, (2) to model certain tactical functions, operations, intelligence, etc., and (3) to identify and/or specify specific data to be collected during the simulation in a form suitable for efficient analysis of the simulation results.

Model utility statements are also used to aid in construction of tactical procedures. Although the basic tactical commands and procedures closely resemble the actual commands and procedures of a real situation, the model utility statements are needed to provide the necessary communication between the modeler and the computer. Model utility statements consist of statements that (a) manipulate processes, such as start a process when an "event" occurs or delay a process until a given simulation time is reached, and (b) specify particular output data collection requirements.

The following set of statements is provided to be used in addition to those available in the underlying language.

Statement	Purpose
END_CASE	Terminates execution of case
END_PLATFORM	Terminates platform
START_CONCURRENT	Indicates the beginning of a set of commands whose execution is to be started simultaneously
STOP_CONCURRENT	Indicates the end of the set of above commands
TRUE_DATA	Gets the true data about another platform
GAMTIM(days,hours, minutes,seconds)	Gets simulation clock time in days, hours, minutes, and seconds
NOTE	Place MOE entry in the output data base
WHEN "event" START process	Starts the specified process when the "event" occurs
WHEN "event" STOP process	Stops process when the "event" occurs
WHEN TIME=time START process	Starts process when "time" is reached
WHEN TIME=time STOP process	Stops process when "time" is reached
DEFINE EVENT="event"	Defines a local event
DECLARE local-platform-event	Declares a local event, i.e., a tactician defined event
WAIT_FOR event	Waits for the occurrence of the "event" before proceeding
WAIT_UNTIL days:hours: minutes:seconds	Waits until indicated time is reached before proceeding
DELAY days:hours:seconds	Delays the execution of subsequent commands or statements until specified simulation time has elapsed