Original

D. R. S. Michalski

The Engineering Society
For Advancing Mobility
Land Sea Air and Space®

400 COMMONWEALTH DRIVE WARRENDALE, PA 15096

SAE Technical Paper Series

891053

Real-Time Decision Making for Autonomous Flight Control

Sulyuth Swangwanna
Computer Science Department
Wichita State University
Wichita, KS

Jan M. Zytkow
Department of Computer Science
George Mason University
Fairfax, VA

General Aviation Aircraft
Meeting & Exposition
Wichita, Kansas
April 11-13, 1989

The appearance of the code at the bottom of the first page of this paper indicates SAE's consent that copies of the paper may be made for personal or internal use, or for the personal or internal use of specific clients. This consent is given on the condition, however, that the copier pay the stated per article copy fee through the Copyright Clearance Center, Inc., Operations Center, P.O. Box 765, Schenectady, N.Y. 12301, for copying beyond that permitted by Sections 107 or 108 of the U.S. Copyright Law. This consent does not extend to other kinds of copying such as copying for general distribution, for advertising or promotional purposes, for creating new collective works, or for resale.

Papers published prior to 1978 may also be copied at a per paper fee of $2.50 under the above stated conditions.

SAE routinely stocks printed papers for a period of three years following date of publication. Direct your orders to SAE Order Department.

To obtain quantity reprint rates, permission to reprint a technical paper or permission to use copyrighted SAE publications in other works, contact the SAE Publications Division.

All SAE papers are abstracted and indexed in the SAE Global Mobility Database

No part of this publication may be reproduced in any form, in an electronic retrieval system or otherwise, without the prior written permission of the publisher.

ISSN 0148 - 7191

Copyright © 1989 Society of Automotive Engineers,Inc.

Positions and opinions advanced in this paper are those of the author(s) and not necessarily those of SAE. The author is solely responsible for the content of the paper. A process is available by which discussions will be printed with the paper if it is published in SAE Transactions. For permission to publish this paper in full or in part, contact the SAE Publications Division.

Persons wishing to submit papers to be considered for presentation or publication through SAE should send the manuscript or a 300 word abstract of a proposed manuscript to: Secretary, Engineering Activity Board, SAE.

Printed in U.S.A.

891053

Real-Time Decision Making for Autonomous Flight Control

Sulyuth Swangwanna
Computer Science Department
Wichita State University
Wichita, KS

Jan M. Zytkow
George Mason University

ABSTRACT

This paper describes a software system that makes real time decisions for an autonomous airplane in a simulated combat environment. It utilizes a network of plans and goals and selects them dynamically, according to the changing circumstances and to the mission goals. Architecture of the system allows for quick, hierarchical selection of plans and flexible execution by adjusting plan instantiation to the varying situation. The system monitors its own performance, checking for the safety, resources, and effectiveness of currently executed plans and dropping those that become dangerous or counterproductive. The architecture of the system allows for a prompt response to dangerous situations by suspending the currently executed plan and invoking an emergency plan. The system can follow commands from the formation leader or ground control. Commands may be very specific, but the system can also follow commands on higher levels of abstraction, deciding autonomously on necessary details.

A RELIABLE AUTOMATED PILOT has become one of the focal points of applied artificial intelligence (1,2,3,4). The advantages are obvious, especially for dangerous airplane missions. But the knowledge requirements and decision making ability required of an automated pilot are tremendous, and certainly some time will pass before a trustworthy automated pilot will seize the cockpit. An intermediate form of automated control, the so called "Pilot's Associate", is also being investigated. The most sophisticated, central module of Pilot's Associate, aiding the pilot in his decision making process, is called the "Tactical Manager" (5). An autonomous decision making system, which can be viewed as a tactical manager expert system, is the subject of this paper. We describe the system architecture which includes the basics of pilot's decision making process. Our Tactical Manager plays the role of an automated pilot rather than a pilot's associate. It applies in a simulated combat environment. We start with an analysis of requirements, then describe the design and implementation of the system as well as its performance in simulation based tests. Airplanes can "see" each other by mutual access to the locations and velocities records. This way we are abstracting from sensor data analysis. Since our primary task is an architecture for decision making we consider 2D movements only.

REQUIREMENTS: TASK PERSPECTIVE

A variety of operational requirements must be satisfied. They may be divided into several broad categories. An autonomous pilot must:

1. Work in real time, frequently updating its view of the external situation and respond quickly, preferably before the next input of data. The frequency of the recognition-action cycle may be on the order of one second, although it should eventually be shorter for a real-life implementation.
2. Utilize large quantities of knowledge. The system's knowledge should be easy to understand and structured in a way compatible with human knowledge, since it may then be evaluated more easily, and may be augmented with additional tactical knowledge of human pilots.
3. Apply towards a broad variety of goals. The system needs to understand its top level mission goals in the context of various positional situations, available weapons, maneuvering capability, etc., and select a response appropriate to the situation.
4. Respond non-schematically, because predictable tactics may be defeated. Be able to choose between various responses.
5. Apply cooperative tactics. It should be able to cooperate with a similar system

0148-7191/89/0411-1053$02.50
Copyright 1989 Society of Automotive Engineers, Inc.

or with a human pilot. According to common wisdom "Under most combat conditions the advantages of mutual support will outweigh the advantages of single fighters." (6). The Tactical Manager should provide for and utilize such tactics.

6. Prioritize among a number of conflicting goals, in particular between following a command, avoiding a direct threat, and following its own mission goal.

7. Act flexibly, staying on a particular goal for a sufficient period but dropping a goal if it cannot be achieved.

8. Switch immediately from one goal to another if the situation requires.

9. Act reliably in a broad range of situations.

10. Choose solutions that defeat the enemy. In the simulation environment the enemy is represented by a different copy of the Tactical Manager Expert System.

REQUIREMENTS ANALYSIS: AI PERSPECTIVE

To meet the requirement of real-time performance our Tactical Manager must operate in a one second recognition-action cycle, including its decision making process. The system must apply large quantities of knowledge, but avoid the combinatorial explosion in planning. This suggests a number of design considerations concerning the requirements 1 and 2, above.

First, the system should use preprogrammed plans, as opposed to doing its own planning, whenever possible. Autonomous planning, or plan generation, involves extensive search, while selection of an existing plan is much quicker if the knowledge base is appropriately organized. Findings of knowledge engineering sessions with pilots further support this solution. Most pilots' knowledge is organized in plans that have operational conditions of application and a clear sequencing of execution. Modern day combat pilots are subject to extensive training which provides them with a large number of clearly defined procedures and a deep conviction that the procedures should be followed because they are more effective than ad hoc solutions. This is especially true in emergency situations or in cooperative tactics which require mutual understanding between allies, since any deviation from accepted plans may be damaging.

Second, the knowledge base should be distributed in a context-dependent manner rather than being stored in a single large database operating under a production system inference engine. The final system may utilize hundreds of plans, each of them with a variety of potential versions valid under different circumstances. The distributed solution was successfully used in MYCIN (7), and our system follows that example.

Third, the system should apply hierarchical planning, starting from the high level abstract plan which is filled with increasingly detailed subplans at lower levels. Associated with hierarchical planning is the idea of instantiation of each plan at the latest possible moment, just before performance of an action is required.

Plans are subordinate to goals. The system should be able to decide on its current goal and then select among the best plans for achieving that goal. An oriented network of plans and goals satisfies requirements 3 and 4, and fits the postulate of context-dependent distribution of knowledge-base. Goals are OR nodes, each connected to a number of plans. Selection of a few plans for each goal and each situation allows for varied response. Each plan is either a leaf node or is an AND node in the network, connected to a number of subgoals and subplans. Network structure allows for easy addition of new plans and goals.

Cooperative tactics involve a choice between conflicting decisions and change of the goals. A rapid change in the mode of operation may be also necessary in response to a sudden immediate danger. To satisfy requirements 5 nad 6, a wingman has to prioritize among a response to a direct threat, a new decision of his leader, an old command that has not yet been completed, and his own judgement about satisfying the mission goals. The Tactical Manager for the leader airplane requires similar capability of choice and swift transition from one goal to another. Such a capability should be available at at each cycle, even if the actual conflicts are rather infrequent.

Results of hierarchical selection of plans and goals can be implemented on a stack. The mission goal is positioned at the bottom of the stack, whereas the currently executed, most detailed plan, stays at the top and is removed after being completed. The stack of goals and plans that is under the control of the system satisfies a number of needs, including requirements 7 and 8. While the regular mode of operation on the stack is analogous to the lisp interpreter operation on the lisp stack of function calls, an immediate transition from one goal to another that was postulated in the former paragraph can be implemented by simple manipulations on the stack. The stack can be kept in memory, allowing continuous work on a given combination of goals and plans for as many cycles as required. Tactical Manager's own performance can be monitored and evaluated by a self-awareness module that, at each cycle, compares the stack to the external situation and to the airplane status.

Testing and development can be best conducted in a simulated combat environment, which allows the Tactical Manager to function in a mode close to its destination, while providing the programmer with a large selection of tools for system evaluation. Video output allows for visual inspection and intuitive judgement about performance. The consequences of each decision can be traced and evaluated by experimentation. Different capabilities can be turned on and

off selectively, allowing to compare their intended impact with the actual one. Simulation provides the opportunity to confront the Tactical Manager with a variety of possible enemy tactics.

The space of all solutions is too large and underdefined to be systematically studied in search for optimality. The notion of optimal solution may have no operational meaning. Therefore we are only looking for a satisficing solution, acceptable because it defeats given class of enemy tactics, as determined by extensive testing under a broad range of situations.

Learning is not considered in this paper, although our Tactical Manager can learn from simulation-based experience (4).

IMPLEMENTATION METHODOLOGY

A symbol oriented language such as LISP appears to be the best programming tool for the development of the Tactical Manager. Symbol processing is vital for complex decision making while numerical calculations can be also efficiently executed in LISP. Sophisticated programming environments are available on artificial intelligence workstations. We have been using a Symbolics 3670 Lisp Machine and CommonLisp in our implementation.

For a project of this complexity there is virtually no choice other than to start from functionally simple system applied to a single tactical mission and to develop an initial prototype that handles such a limited scope in a simulation. Our initial prototype, based on a fighter sweep mission, was tested, revised, and augmented many times, resulting in the system which is described in this paper.

EXAMPLE OF A MISSION: FIGHTER SWEEP

Consider the task of maintaining control of an airspace. A patrol of two friendly aircraft fly a prescribed pattern while attempting to locate enemy aircraft. Any enemy aircraft located is then attacked.

A minimal scenario for this mission includes three aircraft: an allied patrol that consists of a leader and a wingman, and an enemy airplane entering the patrolled area. To execute a simulation for such a scenario requires three versions of the Tactical Manager, one for each airplane.

The enemy airplane is approached at a certain distance and attacked by firing a missile towards it. The firing distance is chosen so that the target plane is within the missile launch envelope of the firing plane, while keeping the firing plane out of

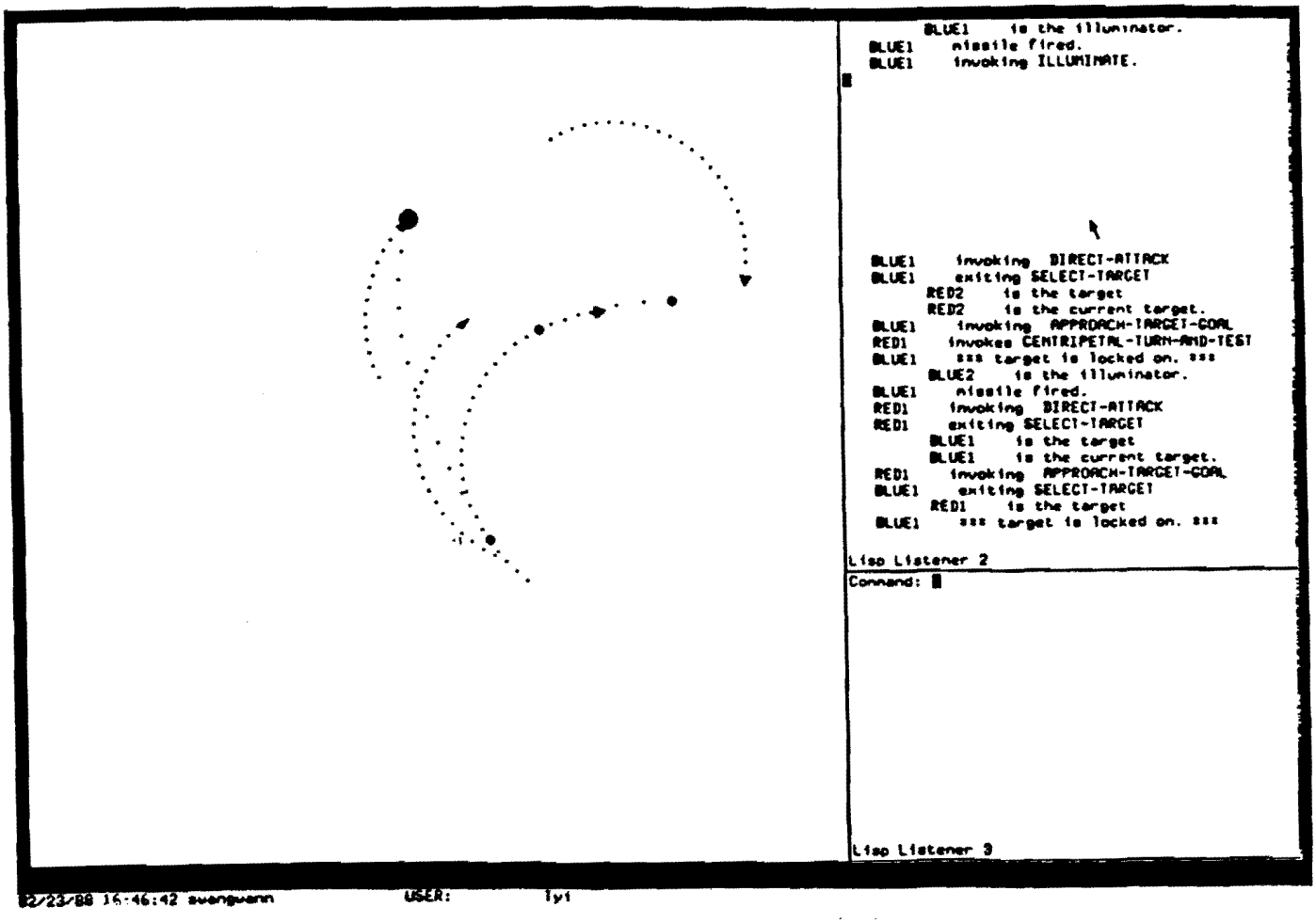

Figure 1. An example of fighter combat: 2 blue against 2 red aircraft

the enemy's missile launch envelope, if possible. The second allied plane then 'illuminates' the target with a radar signal, thus guiding the missile. This allows the first allied plane to escape to safety, while the second airplane may illuminate from a safe distance.

SIMULATION CYCLE

Simulated air combat can include any number of airplanes and missiles. At the top level the system executes in a continuous simulation cycle, each cycle being composed of three steps:

1. For each airplane: a recognition-action cycle which provides decisions concerning new velocities, missile firing, etc.
2. For each airplane and missile: updating the positions and velocities to their newly computed values,
3. (optional) Displaying the new positions of airplanes and missiles on the screen.

Decision making is performed in the first step. Reassignment of values that is done in the second step models the physical movement, changing the situation in the simulated world and providing fresh data for the next recognition-action cycle for each airplane. In our simulation, each plane can read the position and velocity of every other plane. A sample of a simulation in progress is provided in Figure 1.

RECOGNITION-ACTION CYCLE

The tactical planning and decision making for each airplane is performed in the recognition-action cycle. The cycle for a single airplane includes:

1. Deciding whether the current situation requires immediate, extraordinary response by switching to other goals and plans, and if not:
2. Deciding whether the current plan should continue, and if not:
3. Selection of an appropriate plan,
4. Execution of an action that is provided by the plan that is currently on the top of the stack. This may result in velocity and heading changes, missiles being fired, etc.

If the current plan has not yet been completed and is still effective, safe to continue, and it satisfies resources requirements at a given cycle, it will be continued. The stack of plans and goals is remembered and will be used at the next cycle for the same aircraft.

KNOWLEDGE REPRESENTATION

The Tactical Manager's knowledge is organized primarily within a network of goals and plans. A part of such a network is reproduced in Figure 2. Goals are OR nodes, while plans are AND nodes in this network. Plans provide executable actions for every cycle until their goal is reached, or

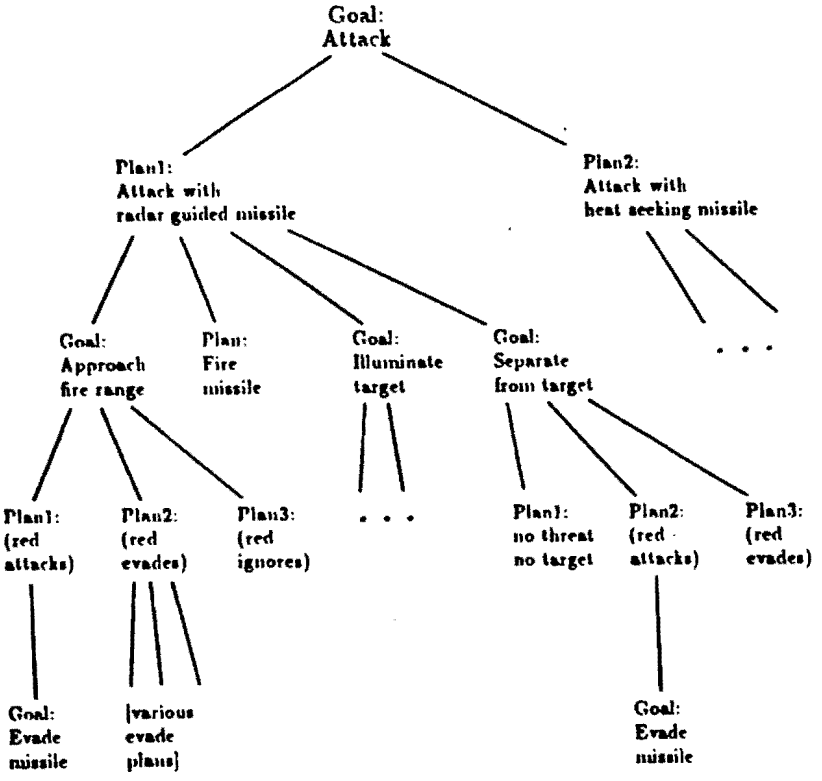

Figure 2. A part of the network of plans and goals

another termination condition is met. After a plan is selected, some of its parameters may be instantiated at each cycle, in response to the details of the situation. It is not enough to simply select the 'Evade' plan and hope for a swift turn. The direction of the turn must be specified, the angular acceleration, and possibly the duration. The values of these parameters may vary from cycle to cycle.

In addition to plans and goals, part of knowledge is implemented as procedures, including computational geometry, kinematics, tests for goal satisfaction, tests for plan effectiveness, and so forth. Other parts of knowledge take the form of data structures, such as aircraft types and missile types. The knowledge about missiles is useful to the Tactical Managers, but it is also used by the missile drivers.

Instances of records in the categories of airplane types and missile types store knowledge that is permanent, while instances in the categories of airplanes and missiles store information related to particular objects in particular scenarios, varying during simulation. This information, referred to as data, is described in the next section. Airplane and missile types describe physical capabilities and limitations of aircraft and missiles, and therefore rarely, if ever, change. They form the knowledge base of possible types of aircraft which may be simulated and missiles that can be applied as weapons.

DATA REPRESENTATION

Data consist of information specific to a particular simulation. By giving initial values to this set of data structures, a scenario is created and can be executed in the simulation. Data can be divided into two categories, based upon the permanence of their values in a scenario.

Part of data describe information which is mission dependent, but which remains constant throughout the mission. These include structures to describe a formation, attack information (friends, foes, etc.), as well as instances of these structures which are created for each aircraft in the scenario.

Some data items will change constantly throughout the simulation of a particular scenario. These include some values within each aircraft's data structure, the values within each missile's data structure, as well as the instantiation of the missile data structures themselves. Examples are such data as velocities, headings, current targets, known enemies within range, etc. By considering their values at a particular instant we may obtain a snapshot of the simulation at that time.

PLAN INTERPRETER AND PLAN EXECUTION

The Plan Interpreter can be seen as a level of program abstraction. It functions as a translator for plans, maintaining the stack of plans and executing the appropriate plans for each aircraft. The current mission status for each airplane can be represented by a number of plans arranged in a stack. The plan on the top of the stack is currently being executed, while other plans in the stack are suspended in various stages of their execution.

Some plans are performed within one cycle, while others last for many cycles. At the end of each cycle the current stack of plans is converted into an instantaneous description that 'freezes' the stage of execution of each plan in the stack. At the beginning of the next cycle the stack is retrieved by the Plan Interpreter, and after checking and setting the priorities, the execution of the top level plan is continued. The plan is then advanced one cycle, packed again, and so on. There are several reasons for utilizing such an interpreter:

1. Plans can share a uniform format, simplifying the design and coding.
2. The stack is directly accessible to the actions included in plans and can be easily modified. The lisp stack maintained by the lisp interpreter is not so flexible.
3. Flexibility is important when dealing with situations which may require stack manipulations. In real-life situations, some dangers may initiate a complete turnabout in plans, therefore the Tactical Manager must also have this capability.
4. Simulation includes action of several Tactical Managers, each having its own stack of plans. Only one stack is active at one time, whereas the instantaneous descriptions of the stacks for other airplanes are put into storage.

A plan is halted by the plan interpreter after its goal has been achieved, or it may be exited earlier if it is no longer appropriate or if its continuation is counterproductive. When a plan is halted, the next plan on the stack, usually corresponding to a higher level goal which had been suspended, is continued.

HIERARCHICAL CALLS TO PLANS

Plans and goals can call other plans and goals. The called plans and goals are placed on top of their calling plans and goals in the stack. When the called plan has been terminated, it is removed from the stack, and the execution of the calling plan continues. For instance, if the sequence of actions includes: (i) approach attack point, (ii) fire a missile, (iii) evade, then the plans for a wingman who fires a missile and leaves its illumination to his leader may be executed in such a manner:

```
Follow-leader
  Co-op-attack-with-missiles
    Check-and-choose-role
    Approach-attack-point
    Fire-missile
    Disengage
  (End Co-op-attack)
(Resume Follow-leader),
```

where indentations indicate position on the stack.

PRIORITIES

At the beginning of each cycle for any aircraft, an extraordinary, emergency plan may be invoked. The plan interpreter checks such a possibility once every cycle, allowing for an immediate response to any emergency situation. It need not be done more frequently, because new data is available only once every cycle. Selection of an extraordinary plan is based upon multiple factors: (i) new commands (from the leader aircraft); (ii) old commands; (iii) threats; (iv) default procedures. A priority level must be given to each of these factors. Each factor is tested in the priority order to see if any action is required. If so, an appropriate plan or goal is selected and deposited on the top of the stack. If no action is needed, the next factor is checked. After that, the plan interpreter starts working on the top item in the stack. When a top priority goal was deposited on the stack in result of a command or a threat, such a goal is usually specified at a high level, for instance: 'evade airplane A', or 'attack airplane B'. In such a case the plan interpreter will make selection of more concrete, lower level plans and goals, until a directly executable plan is reached. The network of plans and goals is guiding this selection process, exactly in the same way as in regular, non-emergency situations.

META-LEVEL REASONING ABOUT OWN ACTIONS

Tactical Manager can reason about its own actions. Whenever the plan interpreter considers a new item at the top of the stack, it answers a number of questions: (1) Is the current plan still valid in the new situation? (2) If I continue this plan, am I going to get into a dangerous situation? (3) Am I making enough progress towards my current goal? (4) Do I have resources adequate for the current plan? (5) What is the next goal? Can I progress towards that goal while working on the current goal at the same time?

After questions 1-4 are answered positively, the execution of the current plan continues, otherwise the top level plan is removed. The top level plan is not changed in result of question 5 is kept, but it may be instantiated differently. If the next goal has been found, the top level plan can be instantiated differently, so that progress is made both on the current goal and the next goal at the same time. The lookahead procedure which allows for finding the next goal in question (5) is guided by the tree of goals and plans (Figure 2), and by projecting the current situation to the time when the current goal should be completed.

TESTING AND DEVELOPMENT THROUGH SIMULATION

In order to execute a simulation, it is necessary to define the scenario, including the initial conditions and the mission (goals) for each aircraft. Defining a scenario requires creating several instances of data structures:

1. one for each aircraft in the simulation, specifying the initial location, velocity, flight role, etc.,
2. one for each formation, specifying each airplane in the formation and its relative location to the leader,
3. describing the attack information for each side, specifying friends, foes, firing range, and so forth.
4. describing the mission for each aircraft by specifying priorities, goals and default actions to be taken. Mission will generally differ for different aircraft taking part in a scenario.

The creation of these structures and their values completely describes a scenario and enables execution of the simulation. To facilitate generation of scenarios, various formation and attack information structures have been predefined.

In testing the Tactical Manager we followed two approaches. The first is based on generating scenarios which, we believe, are particularly interesting and diagnostic. We run simulations of these cases, and by visual inspection of the oucomes we decide whether the result is satisfactory. This way we can detect gross weaknesses and malfunctions. By making visual comparisons, we can also find out whether a particular addition to the system removes a particular weakness.

After several test cases, we become convinced that a particular addition brings a positive change in Tactical Manager's performance. Then we apply the second approach. It consists in systematic testing and comparison of performance before and after a particular change has been made. By this approach we may determine whether the improvement is universal, or whether there are some areas of weakness of the new facility. We can also produce statistics of various performance measurements (8). Tests in the second category cover systematically a cartesian product of sets of values of parameters important for a particular plan or goal. Testing is exponentially explosive when the number of parameters increases, so that systematic testing may be replaced by statistical sampling. So far we conducted

systematic testing using increments of values that produce not more than 5000 tests in a particular space (8).

Different facilities of the Tactical manager can be enabled or disabled for the purpose of testing, so that any subset of features can be isolated. Advantages of each feature discussed in this paper can be demonstrated both by systematic testing and on demonstration runs.

In all tests our Tactical Management System performs in real time, although we did not yet consider architectural solution to enforce real time behavior (3).

The nature of the Tactical Manager implies that it will not be supported by a clear and decidable underlying theory. But then, no existing approach to tactics generation throughout history has been, nor has any major engineering achievement been proven to work before it was demonstrated. Still, the tactics generated by and performance of the Tactical Manager may be subject to evaluation by comparison with human pilot performance or with another Tactical Manager.

SUMMARY

Although far from a complete Tactical Manager System, our implementation:
1. Covers the most characteristic aspects of an autonomous pilot;
2. Simulates a real-time system and performs in real time;
3. Allows for varied responses of the system, depending on commands received, external situation recognition, and default plans;
4. Allows for switching from normal plans to emergency plans at any time;
5. Allows for simple and natural implementation of goals and plans;
6. Allows for monitoring the performance of the currently executing plans, removing plans that are counterproductive.

ACKNOWLEDGEMENT

We would like to thank Michael Erickson and Prakash Govindarajulu for their help and numerous discussions. This work was supported by the Army Research Office under grant #DAAL 03-87-G-0003, and in part by the Advanced Research Projects Agency under a grant #NOOO1487-K-0874, administered by the Office of Naval Research.

REFERENCES

1. Azarewicz, J., Fala G., Fink, R. and Heithecker, C., Plan recognition for airborne tactical decision making. AAAI-86, Morgan Kaufmann Publ. Los Altos CA. 1986, p.805-811.

2. Zytkow, J.M. and Erickson, M., Tactical Manager in a simulated environment. In: Ras, Z. and Zemankowa, M. eds. Methodologies for Intelligent Systems, Elsevier, 1987, p.139-147.

3. Pettersson, G. and Stromberg D., System architecture for a real planning agent in an autonomous air craft. In: Ras Z. and Saitta L. eds. Methodologies for Intelligent Systems 3, North-Holland, 1988, p.132-139.

4. Erickson, M. and Zytkow, J.M., Utilizing experience for improving the Tactical Manager. in: Laird, J. ed. Proceedings of the Fifth International Conference on Machine Learning, Morgan Kaufmann Publ. San Mateo CA. 1988, p.444-450.

5. Eisenhardt, R.G., "Pilot's Associate Definition Study", Final Report for Period 7/84 - 5/85. Perceptronics, Inc., Ann Arbor, Michigan, 1985.

6. Shaw, R.L. , Fighter Combat: Tactics and Maneuvering. Naval Institute Press, Annapolis, MD. 1985.

7. Buchanan, B.G., and Shortliffe, E.H., Rule-Based Expert Systems: The MYCIN Experiment of the Stanford Heuristic Programming Project. Addison-Wesley Publ. Reading, MA. 1984

8. Swangwanna, S., Squeeze: an algorithm which combines two plans together. M.S. Thesis, Computer Science Department, Wichita State University, 1988.

This paper is subject to revision. Statements and opinions advanced in papers or discussion are the author's and are his responsibility, not SAE's; however, the paper has been edited by SAE for uniform styling and format. Discussion will be printed with the paper if it is published in SAE Transactions. For permission to publish this paper in full or in part, contact the SAE Publications Division.

Persons wishing to submit papers to be considered for presentation or publication through SAE should send the manuscript or a 300 word abstract of a proposed manuscript to: Secretary, Engineering Activity Board, SAE.

Printed in U.S.A.