A NOVEL PARADIGM FOR REPRESENTING TACTICAL KNOWLEDGE IN INTELLIGENT SIMULATED OPPONENTS

Avelino J. Gonzalez
Department of Electrical and Computer Engineering
University of Central Florida
Orlando, FL, 32816
(407) 823-5027
ajg@engr.ucf.edu

Robert H. Ahlers
Human, Systems Integration Division
Code 261
Naval Training Systems Center
Orlando, FL 32826
ahlers@ntsc-rd.navy.mil

ABSTRACT

The investigation described in this paper is an on-going effort aimed at developing a concise, yet rich representation paradigm that could effectively and efficiently be used to model the intelligent behavior of opponents in a simulation-based tactical training system. This feature would be quite useful in the training process for two reasons: 1) the trainee would face a realistic enemy who is knowledgeable about tactics in the domain of interest and, 2) the instructor would not have to be burdened with playing the part of the enemy in those training systems where this is commonly done.

The representation paradigm proposed is based on the idea that applicable tactical knowledge is highly dependent upon the situation being faced by the decision maker (i.e., the *context*). A combination of script-like structures and pattern-matching rules in an object-oriented environment could serve to hold all knowledge pertinent to the context present at the time. This paradigm was tested in a prototype system that incorporated the knowledge of a submarine tactical officer on a patrol mission. The prototype was implemented in CLIPS 5.1, a rule and object-based language developed by NASA. The results of tests with the prototype show that the context-based representation paradigm promises to meet the desired levels of conciseness and effectiveness described in the paper. Further research to be conducted in this area is also discussed.

1.0 INTRODUCTION

The use of autonomous and intelligent simulated adversaries in a training simulation can provide a more realistic experience to the trainees than would be presented otherwise. This would be especially true for tactical training simulators if an intelligent simulated adversary could be made to react and counter the trainee's tactics in a realistic fashion.

In many training simulators, the instructor typically controls the simulated adversaries, providing them with intelligent behavior. But this can be quite burdensome to the instructor. The presence of intelligent adversaries that can autonomously react to the trainees' action in a realistic fashion would not only serve to increase the effectiveness of the training process but also make it more efficient by off-loading some of the tasks from the instructor. Intelligent simulated adversaries will be henceforth referred to in this paper as *Autonomous Intelligent Platforms* (or *AIP's*).

Tactical knowledge is required in order to endow AIP's with the ability to act, not only intelligently, but realistically, in light of a trainee's actions. In general, tactical knowledge can be said to address time-stressed tasks which require 1) assessment of the situation at hand, 2) selection of a plan to most properly address the present situation, and 3) execution of that plan [Thorndike, 1984]. It is how this knowledge is represented and manipulated that is the focus of this on-going research project.

But representing tactical knowledge is a difficult task due to the many potential variations of any particular scenario and the multiple actions that can be taken for each. A rule-based paradigm appears to be a natural way to represent this knowledge, but the numerous conditions resulting from the many variations can translate into an explosive number of rules for even relatively simple tasks. Alternative representation and reasoning paradigms such as model-based, constraint-based, or case-based reasoning, although promising in some respects, (see Borning, [1977]; and Castillo, [1991]) are not "natural" for this form of knowledge since they do not easily capture the heuristics involved.

Significant research efforts exist in the development of AIP's. Most of it has centered around the concept of *Semi-Automated Forces (SAFOR)* applied to large-scale land battles and associated with the SIMNET and DIS efforts undertaken by DARPA and the U. S. Army. While no satisfactory definition has been agreed upon by the research community and its customer base, SAFOR, as per its name, attempts to provide a limited degree of autonomy to simulated friendly, as well as opposing, forces in order to reduce the manpower required to carry out a full scale simulated war game. Some implementations of SAFOR do provide full automation.

The various SAFOR projects surveyed [DMSO, 1993] present different means of representing intelligent behavior. Most of these approach the problem from a conventional software standpoint, and make use of procedures in an object-oriented environment to control the AIP's. One interesting variation [Smith, 1992], is the use of finite state machines to represent goals or desired states in the behavior of the AIP. These states are represented as C-Language functions.

Another common approach to the AIP problem has been to use blackboard architectures to represent and organize the knowledge. One implementation [Chu, 1986] uses separate knowledge sources to carry out tasks such as situation assessment, planning, formulation of objectives, and execution of the plan. This system also implements adaptive training so that the AIP can modify its action according to the skills on which the trainee needs to concentrate. The system uses a modified version of Petri Nets to represent instructional knowledge.

A third approach has come from cognitive science researchers [Weiland, 1992; Zubritsky, 1989; Zachary, 1989]. These efforts do not directly address AIP's, but rather, the closely related problem of cognitive modeling of the human decision-making process. Their application of their research to tactical decision making in naval warfare environment makes it even more applicable to the goals of this investigation. Their efforts also make use of a blackboard architecture.

Work carried out in the form of maneuver decision aids at the Naval Undersea Warfare Center [Benjamin, 1993] has also employed a blackboard architecture. The objective of this research is to assist the submarine approach officer in determining the most appropriate maneuver to carry out to counter an existing threat, or to accomplish a mission.

All the systems surveyed appear to be quite large and complex in nature. While it is true that difficult problems require complex solutions, complexity per se is not a desirable characteristic of any solution. This investigation, therefore, has focused on a means of concisely representing tactical knowledge such as that required for AIP's, and its efficient manipulation. The first application chosen has been that of submarine tactics. While admittedly a less complicated domain than large scale land battles, the area of submarine tactical operations nevertheless provides a rich and worthy problem on which to test any novel approaches. The next section of the paper will discuss such an approach.

2.0 GENERAL DESCRIPTION OF CONTEXT-BASED REPRESENTATION PARADIGM

The approach described here is based on the following assumptions:

1) Life for an AIP is a continuous sequence of *contexts*, which change as the situation changes. A context can be likened to a situation that has been recognized, and which has a prescribed set of procedures that must be carried out, either sequentially or arbitrarily. The behavior of an AIP in the simulation is controlled by the context that is *active* for it at the time.

2) The active context may not be the same for all AIP's. This is reasonable to expect, since each may have a different mission, different sensor inputs, and different capabilities.

3) A specific context or group of contexts is always in effect. More than one context can be valid, but only one may be active. Furthermore, all valid contexts must be compatible. Thus, one context must be the central focus of attention. For example, using the case of traveling in an automobile, the driver may be cruising on an interstate normally at the speed limit, which can be characterized as a **normal-highway-driving** context. Moreover, he/she may also be hungry which can be labeled **driver-hungry**. If he/she is more anxious to arrive at the destination than hungry, then the first situation will dictate the action being undertaken and, therefore, be the active context (i.e., continue driving). Otherwise, the **driver-hungry** context will be the active one, and the action will shift towards finding a place to eat. Yet, both contexts may be valid simultaneously, and are clearly compatible.

4) Contexts are represented temporally as intervals of time rather than time points. Contexts can be considered to be transitions to reach a goal (look for a place to eat), or they can be a goal in themselves (eating).

5) Goals can be time points, but only to serve as transitions to other contexts.

6) Only a limited number of things can take place in any single context. A situation, therefore, by its very nature, will limit the number of other situations that can take place. Using the example of an automobile driver, it would not be normally expected that a tire blowout take place while waiting at a stop light. This can be used to advantage to prune the search space of the problem, since there is no need to consider a blowout while waiting at a stoplight. Getting rear-ended, on the other hand, is a much more likely proposition.

7) The presence of a new context will alter either the present course of action or the applicable expectations to some degree. For example, the recognition of a blowout at highway speeds will cause the driver to attempt to coast to a stop while maintaining a firm grip on the steering wheel, and directing the car towards the shoulder of the road. Thus, the context changed from one of **normal-highway-driving**, to one of **blowout**, with its attendant requisite action. This context remains in effect until the car comes to a complete stop (the goal), at which point another context will be recognized and acted upon (e.g., **get-out-of-car**, **inspect-tire**, **change-tire**).

By associating the potential contexts and corresponding actions to specific situations, the identification of a situation can be simplified because only a subset of all possible situations is applicable under the active context. This context-based approach also easily addresses what actionable information to use when a situation is recognized.

One approach to implementing the method described above lies partly in the use of a script-like concept. A *script* is a knowledge representation paradigm developed by Schank [Schank, 1977] which attempts to capture the actions, objects, persons, and concepts that may be related within a given context. For example, a restaurant script will be composed of all the actions which are typically part of going to a restaurant, such as reading the menu, ordering the meal, eating it, and paying the bill. A restaurant script also contains props, objects which are typical to a restaurant scene from the customer's standpoint, (e.g., tables, chairs, menus, food, eating utensils, napkins, salad bars) as well as actors (e.g., waiters, hostesses, chefs, busboys). The actions involved are only those typical of the restaurant experience. It would not be normally expected, therefore, that a customer wash the car at a restaurant.

This concept can be easily extended to military tactics. A script can be used in this application to express the set of steps (at either a high or low level) that are necessary to carry out the action required by the present situation. Within the context of a mission, there is a limited number of things that are generally expected in terms of actions to carry out and the expectations in regards to the possible situations. It would be quite difficult to represent all this knowledge using rules alone. Thus, the basis for the work described here is the use of a script-like structure combined with a minimal number of rules as the knowledge representation paradigm for a set of AIP's. For lack of a better name, this representation and reasoning paradigm will be referred to as a *context-based representation*. The next section describes in greater detail how context-based representation can be implemented to achieve the objectives generally set for AIP's in a simulation.

2.1 Control of AIP Through Contexts

The AIP is controlled by a *General Context (GC)*, a high level context that defines the overall mission to be undertaken. The GC is an instance of a class representing the mission to be executed, and it is composed of *Acts*, intermediate-level contexts that

define certain maneuvers, situations, or tactics relevant to that mission. Acts can be "installed" in a sequential nature or in response to a developing situation. The Acts, in turn, can have *Sub-acts* which are contexts describing lower-level tactics or maneuvers to be undertaken within the Acts.

Each context (GC, Act, or Sub-act) will have a set of rules and/or procedures attached which will implement some action and/or detect when a transition to another context is called for. The rules, in particular, form the basis of an AIP's situation awareness. If a change in the parameters of the simulation calls for a change in the situation, the rules will recognize the change and execute the appropriate change of context. The use of contexts, in addition to prescribing actionable information as a block of procedures, can help in the situation awareness process by limiting the types of situations that can be expected. For example, in a context of peacetime and within territorial waters, a submarine would not expect to be attacked by an enemy.

Explaining the concept of context-based reasoning would be easier if the explanation is tied to an example. Therefore, the representation of submarine tactical knowledge during a routine patrol mission will be used as an illustration. It is also the basis for the prototype described in section 3.0 below.

The AIP which is the recipient of the tactical knowledge will be referred to as ownsub. While this might cause confusion with the traditional naval custom of using this nomenclature to identify the student's own platform, this AIP is the focus of this investigation and should be recognized as such.

Ownsub is represented as an object (instance of class SUBMARINE) in an object-oriented environment. Its static slots (defined as those slots whose values will not change during the simulation) define its capabilities such as its maximum speed, quiet speed, maximum depth, periscope depth, weapon systems, (e.g., number and ranges of torpedoes, missiles), sensors (e.g., range and types of passive sonar, active sonar, towed arrays), and Electronic Warfare capabilities (e.g., sonar decoys). Additionally, ownsub's dynamic slots (those whose values will be updated at least once during the simulation) describe its actual position (i.e., x-coordinate and y-coordinate), depth, heading, and speed, in the course of the simulation, as well as whether the sensing equipment is on or off. Damage assessment as well as the conditions of the stores (e.g., weapons, food supply) is also contained in dynamic slots.

A local database containing all of the external information that is relevant to the mission is also necessary. This includes all mission-dependent static inputs which define the task to be undertaken, as well as the environment that will be experienced by ownsub. Some of these are the 1) mission, 2) geographical information, 3) presence of friendly forces in the sector or in the route, 4) basic assumptions about the state of affairs (e.g., peacetime, hot war, cold war, tensions), as well as others, such as coordination with other friendly forces.

This local database should also contain all the situation-dependent dynamic inputs which ownsub would see from its sensors. Such information partially consists of sonar input and communications with the command or with other friendly ships/submarines.

This type of information should be placed on the database by a central supervisory function that can access all the data in the simulation, yet knows what data is to be made visible to ownsub. While a blackboard architecture seems to be well suited to this approach, its high degree of complexity leads us to discourage its use. The central supervisory function would instead be a global data structure and a set of associated functions which would pass on to the individual AIP's any inputs that each has a "right" to see. That right would be gained through the activation of certain sensors and their constraints. This central supervisory function also plays the role of an omniscient entity by determining which weapons launched are successfully targeted and which are not.

The general description of a context (without the inputs) could be stored in memory or on disk. A context can be "installed" onto an AIP object as may be called for by the situation, overwriting the old one when the change represents mutually-exclusive contexts. If the new context is not mutually-exclusive with the existing one, then it needs to be overlaid onto the old context so that at its completion the original one regains control.

Each context contains procedural attachments to implement procedural control over the simulation, such as dictating the speed, depth and course of ownsub. Moreover, the context would also contain a set of rules together with its own "mini" inference engine consisting of a pattern matcher, a Rete net and

There are basically three types of rules involved in the intelligent decision making: *Sentinel rules* continually monitor the simulation data in order to recognize the factors that can lead to a change in situation. These rules are typically tied into a particular context. Sentinel rules form the basis of situation awareness, since they infer the new situation and, therefore, the context, from raw data.

Transition rules, on the other hand, react to the situation identified by the sentinel rules and determine which context should be activated as a response to the changing situation.

Internal rules are used only to make decisions within the active context. Such decisions do not cause a transition to other contexts. One example is that when in a baffle-clearing sub-act, the course of the submarine must be changed to x degrees for 5 minutes. When the five minutes elapse, the course must be changed to y degrees.

An example of a sentinel rule would be, when the TRANSIT-TO-SECTOR (full-speed travel to the sector to be patrolled) context is active, a rule belonging to that context will monitor the position of ownsub so that arrival at the designated sector is recognized. This rule requires the fact

(active-context TRANSIT-TO-SECTOR)

be present in the factbase. The action of this rule may designate the situation to be

(situation arrived-in-sector ownsub).

A transition rule will recognize this posted fact, and react by activating the SECTOR-SEARCH context, and initializing all appropriate elements. Once the new context is activated, the sentinel rule mentioned above is no longer applicable, since its related context has been deactivated.

Some sentinel rules, however, will always be "active" regardless of which General-Context/Act/Sub-act is active. These are general rules that oversee the entire simulation and are not tied to any one context. For example, a rule that searches for enemy torpedoes needs to be always in an active status whenever ownsub is out of port in a wartime context, whether there are enemy submarines detected or not. If an under-attack situation is identified, then a transition rule will immediately activate the UNDER-ATTACK context. Such rules do not require any active-context facts to be present in the factbase in order to execute.

The above section generally describes how the context-based representation should be implemented. It does not, however, describe exactly how it was actually implemented in the prototype described in Section 3.0 of this report. The prototype represents some simplifications of the description contained above.

3.0 IMPLEMENTATION AND VERIFICATION OF CONTEXT-BASED PARADIGM

The context-based paradigm was implemented in a prototype in order to verify that 1) it can be used to suitably represent tactical knowledge, and that 2) it can do so concisely. The more specific objective of the prototype was to implement a simple mission of searching a pre-determined sector for the presence of enemy submarines, and to track one when found. Such a mission was labeled *SEARCH-AND-TRACK*. The knowledge implemented in the prototype is described in detail in Gonzalez [1992].

Object-oriented languages excel in applications to simulations. Objects can be defined and assigned a certain behavior, which can be controlled by sending messages to it. Since the prototype is, in its most basic terms, a simulation, an object-oriented language was chosen as the implementation tool. Since the endowment of intelligence to the object ownsub was the primary goal of the prototype, an expert system tool which could manipulate rules was also desirable. Many commercially-available expert system shells incorporate these features. CLIPS 5.1 was chosen for the prototype due to its powerful pattern-matching capabilities, its newly-available Clips Object-Oriented Language (COOL), its procedural capability, its availability in a PC platform, and its low cost.

The basis of this prototype is the instantiation of the class SUBMARINE, called "ownsub," which represents the AIP. Ownsub traverses a Cartesian coordinate plane of unlimited size in three dimensions (x-coordinate, y-coordinate and depth). It's actions are controlled by a set of contexts that are based on the General Context SEARCH-AND-TRACK. The Acts that compose the SEARCH-AND-TRACK GC are called:

- TRANSIT-TO-SECTOR

- SECTOR-SEARCH

- COVERT-TRACKING

- TRANSIT-HOME

These four Acts are mutually exclusive, and are described in detail in Gonzalez [1992]. Whenever any one of them is installed, a message is sent to ownsub to initialize its parameters for operation under this context, e.g., setting its speed, heading and depth, activating or deactivating certain sensors, deploying its weapons. A context is installed by asserting a fact to the factbase that advertises that context as active. These facts were discussed in Section 2.0 above, and are of the form:

(active-context <context>)

The prototype is centered around the actions of ownsub. Objects representing up to five enemy submarines can be created, and they are referred to as *opsub1*, *opsub2*, etc., through *opsub5*. Additionally, other objects in the simulation can be created to represent torpedoes (up to three at one time) and sonar decoys (up to five). Only the ownsub and opsubn objects contained any attributes that changed during the course of the simulation (except for the x-y position, which also changed for the torpedoes and other objects.

The performance objective of the prototype was to indirectly control the actions of ownsub by directly controlling those of opsub. The only exception to these was when specific orders were given to ownsub (by fleet command).

Situation awareness can be thought of as the reasoning necessary in order determine when to effect a transition to another context. The sentinel rules described in section 2.0 form the basis of this process. For example, the situation where an enemy is detected is determined by a calculation of the distance between the positions of ownsub and that of all existing opsubs. If the active sonars of all submarines are off, then ownsub's passive sonar will detect the presence of the enemy at 3000 meters. If ownsub's active sonar is on, then that distance increases to 4000 meters. Likewise, if the enemy's active sonar is on, not only is the range then 5000 meters, but it is assumed that the enemy is aware of ownsub's presence, something that was not assumed in the previous two cases. Losing track of an enemy happens when the distance is greater than 500 meters above the applicable range. The range of the torpedoes is roughly 7 Km, which is implemented as a run duration of four minutes at a speed of 100 Km/hr.

It should be noted here that while the above is admittedly a mis-representation of the capabilities of submarines, their weapons, and their sensing equipment (the ranges and speeds used in the prototype are hypothetical), the relatively slow speeds of the submarines involved require that short distances be used in order to limit the duration of the simulation to a reasonable one, while at the same time being able to exhibit the many features of the prototype.

A sentinel rule also searches for the presence of torpedoes, and recognizes them they come within the appropriate range as described above. Other sentinel rules monitor the simulation for the end of an enemy attack, check to see whether a torpedo has hit the target, and determine when to end an evasion maneuver, among many other things.

In some cases, rules were written such that they skipped the intermediate step of explicitly asserting the situation and integrated the functions of the sentinel and the transition rules into one. While this has the effect of making the system more concise, which was one of the objectives of this prototype, it tends to de-modularize the knowledge, something that could be disadvantageous when carrying out the knowledge engineering for a significantly larger system.

4.0 EVALUATION AND DISCUSSION OF THE CONTEXT-BASED PARADIGM AND PROTOTYPE

The prototype was executed numerous times during the course of its development as well as thereafter. Ownsub was able to react autonomously to the actions of the opsub entity as well as to independently initiate maneuvers such as baffle clearing, much the same way as would a submarine piloted by a human. It was able to recognize the presence of an enemy submarine and act in accordance with the mission being carried out. Since its mission was to patrol a sector and track any enemy activity, it searched for the presence of an enemy at quiet speeds, and at a depth which maximized stealth. Once contact with an enemy had been established, ownsub followed the enemy while trying to hide behind its baffles. If it

perceived that it had been detected, ownsub would immediately take evasive action. If ordered by fleet command to attack, it undertook the attack. Success of failure of the attack was based on an external probabilistic function. Likewise, when attacked, it took evasive action by releasing countermeasures before returning the fire.

While ownsub's behavior certainly cannot be considered to be as expert as an experienced submarine captain, it does provide a robust simulation of intelligent behavior to a student who is in a simulated tactical encounter with it. This are in fact the objectives of this project. Thus, for these reasons, it was determined that from a qualitative standpoint, the performance of the prototype was considered to be successful.

One of the features of the prototype was that the simulation of the tactical environment was also done in CLIPS 5.1. While this was an advantage in the sense that the prototype is self contained, the simulation was quite inefficient, since CLIPS is not a robust simulation language. Many of the simulation functions were done with rules, which is not the most efficient manner available. The CLIPS version used would continually run out of memory after medium length runs of the simulation. This placed some limits in the kind of features that could be implemented in the prototype. We hope that this problem is resolved with the new version of CLIPS which makes use of Windows (CLIPS 6.0).

Since one of the major objectives of the context-based paradigm is to simulate intelligent behavior in simulated agents in a concise and efficient manner, the issue of concise representation is a critical one which merits careful evaluation. One simple, albeit crude, way to measure conciseness is to simply count all the CLIPS elements used in the prototype. However, a distinction must be made between elements used for the purpose of carrying out the simulation, and those used for intelligent decision making. Therefore, all the elements for each of the prototypes will be counted, but only those used for decision making were used in the analysis of conciseness. The prototype used the following types and numbers of CLIPS elements in the decision making: 6 classes (SUBMARINE, SECTOR, GENERAL-CONTEXT, SEARCH-AND-TRACK, TORPEDO, and SONAR-DECOY), 15 global variables, 13 time-related functions, 11 general functions, 2 main functions, 18 message handlers, and 34 rules.

Ideally, it would be of great benefit to develop a purely rule-based version of the knowledge and capabilities exhibited by the prototype, so that an objective comparison of efficiency and conciseness could be made. Nevertheless, a rough comparison can be made with another situational awareness knowledge-based system prototype developed at the first author's institution. The latter prototype is a performance monitoring system that evaluates the actions of a student in an automobile driving simulator. Both systems are similar in that they represent situation awareness knowledge and both were developed with CLIPS, but the performance monitoring prototype represents its knowledge in rules alone. While the submarine AIP prototype employing context-based reasoning required 34 rules and 18 message handlers, the automobile driver performance monitor required nearly 50 rules and numerous other functions to perform a much simpler mission. This comparison, rough as it may be, supports the hypothesis that the context-based paradigm is capable of representing the knowledge involved in the SEARCH-AND-TRACK mission in a concise manner. It is also likely that additional refinements in the system could lead to an even more succinct representation.

5.0 FURTHER RESEARCH

The results of the research described indicate that the context-based paradigm can be used to concisely represent the knowledge required of an AIP in a simple training simulation. Nevertheless, significant work remains in order to make the concept a user-hardened technology.

Most importantly, a formalization of the context-based reasoning concept is necessary. The resulting formal knowledge representation paradigm should incorporate features that address temporal reasoning as well as dealing with uncertainties. Additionally, guidelines should be developed to assist a user in creating a knowledge base from human tactical knowledge. These guidelines could be incorporated in a highly interactive graphical authoring system with a look and feel similar to that of the VISTA system [Ahlers, 1990]. Finally, the burden of carrying out the simulation should be placed on a more generalized simulation environment, with the CLIPS prototype merely providing the intelligence and the control of the simulated agents. This would make the system more widely applicable and possibly capable of being retrofitted to existing simulators.

Additional testing in a more complex threat environment is clearly warranted so as to confirm or deny its general applicability to the problem. An implementation in an existing training simulator should be undertaken to determine the feasibility of retrofitting the latter with AIP's.

6.0 ACKNOWLEDGEMENTS

The investigation described in this paper was carried out by the first author under the sponsorship of the Naval Air Warfare Center, Training Systems Division, Human, Systems Integration Division, Orlando, FL, and under the auspices of the ASEE Summer Faculty Program, in the Summer of 1992.

7.0 REFERENCES

[Ahlers, 1990] Ahlers, R. and Schnitzius, M., "Human Engineering the Knowledge Acquisition Interface: The Evolution of VISTA," Proceedings of the The Third Annual Florida Artificial Intelligence Research Symposium, Cocoa Beach, FL, April, 1990.

[Benjamin, 1993] Benjamin, M., Viana, T., Corbett, K. and Silva, A., "The Maneuver Decision Aid: A Knowledge Based Object Oriented Decision Aid," Proceedings of the Sixth Florida Artificial Intelligence Research Symposium, April, 1993, pp 57-61.

[Borning, 1977] Borning, A., ThingLab - An Object-oriented System for Building Simulations Using Constraints," ACM Transactions on Programming, Languages and Systems, 3 (4) October, 1977, pp 353 - 387.

[Castillo, 1991] Castillo, D., "Toward a Paradigm for Modelling and Simulating Intelligent Agents," Doctoral Dissertation, University of Central Florida, December, 1991.

[Chu, 1986] Chu, Y. Y. and Shane, D., "Intelligent Opponent Simulation for Tactical Decision Making," Report PFTR-1120-11/86, sponsored by the Office of Naval Research and the Naval Training Systems Center, 1986.

[DMSO, 1993] DMSO Survey of Semi-Automated Forces, March 15, 1993.

[Gonzalez, 1992] Gonzalez, A. J., "Concise Representation of Autonomous Intelligent Platforms in a Simulation through the Use of Scripts," Technical Report TR92-019, Naval Training Systems Center, Orlando, FL, (in preparation).

[Schank, 1977] Schank, R. C., and Abelson, R. P., 1977, "Scripts, Plans, Goals and Understanding," Erlbaum, Hillsdale, NJ, 1977.

[Smith, 1992] Smith, S. and Petty, M., "Controlling Autonomous Behavior in Real-Time Simulation," Proceedings of the Southeastern Simulation Conference, Pensacola, FL 1992, pp 27-40.

[Thorndike, 1984] Thorndike, P. W., and Wescourt, K. T., "Modeling Time-stressed Situation Assessment and Planning for Intelligent Opponent Simulation," Final Technical Report PPAFTR-1124-84-1, sponsored by the Office of Naval research, July, 1984.

[Weiland, 1992] Weiland, M. Z., Cooke, B., and Peterson, B., "Designing and Implementing Decision Aids for a Complex Environment Using Goal Hierarchies," Proceedings of the Human Factors Society 36rd Annual Meeting, 1992.

[Zachary, 1989] Zachary, W. W., "A Context-based Model of Attention Switching in Computer Human Interaction Domain," Proceedings of the Human Factors Society 33rd Annual Meeting, 1989, pp 286-290.

[Zubritsky, 1989] Zubritsky, M. C., Zachary, W. W., and Ryder, J. M., "Constructing and Applying Cognitive Models to Mission Management Problems in Air Anti-Submarine Warfare," Proceedings of the Human Factors Society 33rd Annual Meeting, 1989, pp 129-133.