CONTEXT-BASED REPRESENTATION OF INTELLIGENT BEHAVIOR IN TRAINING SIMULATIONS

Avelino J. Gonzalez
Department of Electrical and Computer Engineering
University of Central Florida
PO Box 162450
Orlando, FL 32816-2450
ajg@ece.engr.ucf.edu

Robert Ahlers
Naval Air Warfare Center
Training Systems Division
12450 Research Parkway
Orlando, FL 32826

Abstract

This article presents, describes and evaluates a novel behavior representation paradigm that can effectively and efficiently be used to model the behavior of intelligent entities in a simulation. Called *Context-based Reasoning (CxBR)*, this paradigm is designed to be applicable whenever simulation of human behavior is required. However, it is especially well suited to representing tactical behavior of opponents and teammates in simulation-based tactical training systems. Representing human behavior in a simulation is a complex and difficult task that generally requires significant investment in human effort as well as in computing resources. Conciseness and simplicity of representation and efficiency of computation, therefore, are important issues when developing models of intelligent opponents. We believe that this paradigm is an improvement over the rule-based approach, currently a common technique used in representing human behavior.

We have preliminarily tested CxBR in two different prototype systems. Evaluation of the prototype shows that the context-based paradigm promises to meet the desired levels of simplicity, conciseness and efficiency required for the task.

1. INTRODUCTION

The use of autonomous and intelligent simulated platforms in a training simulation can provide a more realistic experience to the trainees than would be presented otherwise. In a military environment, these can be defined as instances of military platforms (i.e., submarine, destroyer, tank, helicopter, or fighter aircraft) which depict adversaries or allies. Alternatively, in game situations, these platforms could represent individual humans (e.g., a quarterback, a shortstop, a point guard) or groups of humans (e.g., a defensive unit in football, or a penalty-killing unit in hockey). Furthermore, in non-competitive situations, they could represent other intelligent platforms such as automobiles or commercial aircraft. Such intelligent simulated platforms will be referred to in this article as *Autonomous Intelligent Platforms* (or *AIP's*).

1.1 Tactical Knowledge

Tactical knowledge is required in order to endow AIP's with the ability to act, not only intelligently, but also realistically, in light of a trainee's actions. In general, tactical knowledge can be said to be 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].

In military confrontations, the selection of the appropriate plan of action to address the current situation is typically guided by military doctrine. Such doctrine is taught to enlisted personnel and officers and is described in several military publications (e.g., Field Manuals). These publications are typically classified or otherwise not generally available to the public. U. S. military doctrine, however, unlike that of some other countries, permits a significant amount of freedom of choice to the combatant, as it would be impossible for the doctrine to cover in detail all possible situations likely to be faced in combat. This unpublished knowledge is considered to be heuristic in nature, and is generally learned through an individual's own experience.

Humans in non-military settings also use tactical knowledge. Examples of this exist in team sports or other everyday tasks like driving an automobile or navigating a vessel. Such knowledge is also partly published (e.g., Rules of Baseball, Nautical Rules of the Road, Traffic Laws) and partly exists as heuristics learned through training or experience. Thus, appropriate tactical behavior in accordance with accepted practice is partly well defined (published rules or doctrine) and partly individual expertise based on heuristics. The basis of our approach to the problem of concisely and efficiently representing the knowledge of an AIP is called *Context-based Reasoning* (CxBR). CxBR considers both of these components in how it represents tactical knowledge.

CxBR is founded upon the following:

> 1) Tactical experts are proficient at their task by recognizing and treating only the key features of the situation. They then proceed to abstract these features for use as premises for general knowledge. Thus, they only use a small, but important portion of the available inputs. An example of this can be found in the tactical exercise of commanding a submarine in a wartime situation.
>
> The commanding officer (CO) of the submarine is generally bombarded with a multitude of inputs when performing his job. He receives audio inputs such as engine noise, electronic noise, and conversations with others around him. He likewise receives visual inputs such as the radar and sonar screens, possibly the periscope, etc., and tactile inputs such as vibrations of the submarine, etc. He is able to cognitively handle these inputs rather easily when they are all in the normal or expected range. However, if one of these should deviate from normal, such as abnormal noise and vibrations, the officer will immediately focus only on these inputs in order to recognize the present situation as, for instance, a potential grounding, collision or engine malfunction. All other inputs being received, meanwhile, are generally ignored during this crisis.

2) There are only a limited number of things that can realistically take place in any situation. Using the example above, it would not be expected that a submarine be visually detected while it is underwater (at a depth of more than 100 feet). This can be used to advantage to prune the search space of the problem, since there is no need to be concerned with visual detection by an adversary when submerged below 100 feet or so.

3) The emergence of a new situation will generally require alteration of the present course of action. For example, the recognition of an impending collision may cause the CO to order the submarine to stop by reversing engine thrust or changing blade pitch angle. Thus, upon recognition of the potential for collision, the *active* context for the CO immediately changed from one of normal navigation, to one of potential-collision, with its attendant course of further action.

4) Tactical knowledge is quite general in nature. The inherent difficulty for non-experts is the classification of the situation in a form abstract enough for the general tactical knowledge to be applicable. One example of general tactical knowledge is:

> *When facing an inferior opponent, who has a tactical disadvantage, attack.*

While the rule is seemingly simple and straightforward, it may be quite difficult for a non-expert to identify what constitutes an "inferior opponent" or a "tactical disadvantage". An expert, on the other hand, would look for finer-grained features such as the opponent's numbers, their weaponry, their defenses, the surrounding terrain or ocean, the weather and his ability to fight in it, the element of surprise, etc. These would allow him to identify the opponent's strength and evaluate his advantage or disadvantage.

Most tactical tasks in the military consist of a pre-defined set of actions that are embarked upon after a certain situation has been recognized. The situation could be a mission, a set of orders, or merely a reflection of a specific set of battle conditions at the moment. The problem faced, therefore, is two-fold: 1) how to recognize the present situation (referred to as *situational awareness*), and 2) what to do when the situation is recognized (referred to as *implementation of actionable information*). Context-based Reasoning not only addresses the (concise) representation of the knowledge required to solve the above problems, but also defines how that knowledge is manipulated to achieve the desired results.

1.2 State of the Art in AIP's

Representing tactical knowledge is a difficult task due to the many potential variations of any particular scenario. A rule-based paradigm appears to be a natural way to represent tactical knowledge because it easily represents the heuristics involved in tactical behavior as IF-THEN rules. 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; Castillo, 1991; Catsimpoolas, 1992]) are not "natural" for this form of knowledge since they do not easily capture the heuristics involved in tactical behavior representation. CxBR, on the other hand, allows these heuristics to be represented through so-called *monitoring rules* (to be described later), but does not make them the focus of the knowledge representation.

Significant research efforts exist in the development of AIP's. Most of it has centered on the concept of *Semi-Automated Forces* (*SAFOR* or *SAF*) applied to large-scale land battles. These are associated with the SIMNET and DIS efforts undertaken by DARPA and the U. S. Army. SAFOR provides a limited degree of autonomy to simulated friendly and 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 for certain battlefield entity classes. However, no universally accepted definition of SAFOR has been agreed upon by the research community and its customer base. The more general name of *Computer Generated Forces (CGF)* has been recently adopted, presumably to address this lack of definition for SAFOR.

The various SAFOR projects [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 popular technique in SAFOR systems has been the use of Finite State Machines (FSM's) to implement goals or desired states in the behavior of the AIP [Dean, 1995]. These states are represented as C-Language functions. Three major SAFOR systems, the CCTT [Ourston, 1995], ModSAF [Calder, 1993], and IST-SAF [Smith, 1992] systems all employ FSM's as the representational paradigm. While similar to CxBR in that behaviors and tasks are also considered to be states, FSM’s differ in the fact that they do not necessarily aggregate all the related tasks, actions, and things to look out for in a self-contained module. CxBR represent an intuitive aggregation of all necessary knowledge for an AIP to display all the facets of behaviors during a specific mission. This has the added advantage that it permits for easy planning for a mission, as all the different contexts required for a mission can be predetermined [Grama, 1998]. Some FSM-based systems, on the other hand, allow for the control of one AIP by more than one FSM at the same time. This goes against the concept that all required knowledge is grouped together in a coherent fashion to control the AIP.

Various other SAFOR systems are described in [Danisas, 1990]. However, they are basically large and complex land battle wargaming systems whose simulated agents exhibit only a modest level of autonomy.

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.

Architectures such as SOAR [Laird, 1987; Tambe, 1995] take a goal-oriented approach in which goals and sub-goals are generated and plans to reach them are formulated and executed. These plans are in effect until the goals are reached, at which point they are replaced by new goals that address the situation. However, SOAR is based on the rule-based paradigm, which, as mentioned before, although very intuitive, has many disadvantages.

Another approach has come from cognitive science researchers [Wieland, 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 efforts also make use of a blackboard architecture. The COGNET representation language [Zachary, 1992] uses a task-based approach based on the GOMS concept [Card, 1983; Olsen, 1990], in an opportunistic reasoning system. In this approach, all actions are defined as tasks to be performed by the AIP. The definition of each task includes a trigger condition that indicates the situation that must be present for that task to compete for activation with other similarly triggered tasks. The use of blackboard system, however, introduces a high overhead and much added complexity.

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.

Golovcsenko [1987] discusses some Air Force prototype systems that exhibit certain amount of autonomy in the simulated agents. One in particular, a special version of the Air Force's TEMPO force planning war game system, uses artificial intelligence techniques to replace one of the human teams involved in the game. In general, however, these are not considered to be AIP's as defined in this article.

All the systems surveyed are quite large and complex in nature. While it is true that the simulation of intelligent behavior is a difficult problem, difficult problems do not always require complex solutions. The context-based approach described here focuses on <u>simply</u> and <u>concisely</u> representing tactical knowledge such as that required for AIP's, and its <u>efficient</u> manipulation. The first application chosen has been that of submarine tactics. Submarine tactical operations provide a rich, real-world problem on which to test any novel approaches.

2. DESCRIPTION OF CONTEXT-BASED REPRESENTATION PARADIGM

Context-based Reasoning is based on the ideas that 1) any recognized situation inherently defines a set of actions or procedures that properly address the situation, and 2) identification of a future situation can be simplified if all things that are likely to happen while under the current situation are limited by the current situation itself. CxBR encapsulates knowledge about appropriate actions and/or procedures as well as possible new situations into *contexts*. By associating the potential future situations and their corresponding actions to specific situations, identification of a new situation can be simplified because only a subset of all possible situations is applicable under the current situation.

Our context-based approach to implementing the concepts described above borrows from the idea of *Scripts*. A script is a knowledge representation paradigm developed by Schank [1977] that attempts to encapsulate the actions, persons, and things 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, paying the bill, etc. A restaurant script also contains *props* (objects typical to a restaurant scene from the customer's standpoint such as tables, chairs, menus, food, eating utensils, etc.) as well as *actors* (i.e., waiters, hostesses, chefs). The actions involved are only those typical of the restaurant experience. It would not be normally expected, for example, that one would go to a restaurant to wash her car. Scripts were originally developed for use in natural language processing (NLP). By inherently associating several actions, actors and props with a concept, a NLP system could make inferences not explicitly stated in the dialogue being interpreted. For example, it would be understood by the NLP system that "... going to the restaurant .." would imply eating there, and not washing one's car

Scripts, however, have not been notably used to represent tactical behavior in AIP's. The concept of scripts, nevertheless, can be easily extended to military tactics, where a script-like structure can be used to hold the set of steps that are necessary to carry out the action required by the present situation. The contexts are such a structure, which when active, can control the behavior of an AIP in a simulation.

Control of an AIP through CxBR assumes the following:

1) Life for an AIP is a continuous and dynamic decision making process. Decisions are heavily influenced by a never-ending sequence of *contexts,* each of which, when *active*, controls the behavior of the AIP and provides an expectation for the future. Active contexts change not only in response to external events or circumstances, but also as a result of actions taken by the AIP itself. One example of a context would be navigating a submarine to a specified sector of ocean. The behavior of an AIP in that situation is controlled by an active context that describes the situation (i.e., **Transit-To-Sector**).

2) The active context may not be the same for all AIP's at the same time. This is reasonable to expect, since each may have a different situation (i.e., a different mission, different sensor inputs, different capabilities, or a different physical location).

3) At least one specific context is always active for each AIP. More than one context can be valid, but only one may be *active,* making it the sole controller of the AIP. For

example, while transiting to a designated sector, a submarine may experience an attack by an opposing warship. Since the crew wants to survive the attack more than they want to reach the sector, the active context will immediately change to one called **Under-Attack**. Upon termination of the threat, **Transit-To-Sector** may once again be set to the active context in order to resume the travel to the designated sector.

4) Contexts are represented temporally as occupying intervals of time rather than being time points. Contexts can be considered to be a sequence of states or tasks required to reach a goal (navigate to the sector).

5) Goals can also be time points, but only to serve as transitions to other contexts. For example, arrival at a designated sector can be defined as a goal of **Transit-To-Sector** and can be represented as a time point. But this is not a context in its own right; only a transition to another context (e.g., **Sector-Search**). This process may go on until the mission ends.

6) A situation, by its very nature, will limit the number of other situations that can realistically follow. Therefore, only a limited number of things can be expected to take place within any one context. Using as an example the domain of submarine warfare, it would not be expected that the submarine would be attacked in its own homeport. This can be used to advantage in pruning the search space of the problem, since there is no need to monitor the simulation for a torpedo attack while waiting to be resupplied at port. If unexpected situations do take place, that introduces the element of surprise into the AIP's behavior, which is highly consistent with the real world.

7) Certain cues exist which indicate that a transition to another context is desirable. This makes use of the hypothesis that experts look for a certain few specific inputs that indicate to them the emergence of a new situation (e.g., the sound of a torpedo on sonar.)

8) The presence of a new context alters the present course of action and/or the applicable expectations. For example, the recognition of an attack typically causes a submarine to attempt an evasion maneuver to trick the incoming torpedo into veering away from it. Thus, the context changed from **Sector-Search** (for example), to one of **Under-Attack**, with its attendant requisite action. This context remains in effect until the danger passes and a counter-attack can be mounted (e.g., **Counter-Attack**).

This is in some ways similar to a system proposed by Thorndike [1984] called "Context Template-driven SAFOR". However, the latter appears to implement AIP's that are intelligent in a much higher level in that they can only respond to orders from higher command, or requests from subordinates. They do not appear to be able to perceive the environment independently. Moreover, the transition from one context to another appears to be significantly more rigid that what is possible in CxBR, which may lead to unintelligent decisions.

Dreyfus and Dreyfus [Dreyfus, 1986] take exception to the idea of using contexts to simulate human intelligence in computers. They correctly point out that there can exist many contexts in the course of human life, and to attempt to account for all of them is a hopeless task. Nevertheless, their argument arises from the standpoint of refuting the claims that computers can be intelligent in the same way as humans, in all aspects of human intelligence. The objective of this work is less ambitious in scope, since the AIP's do not have to have the breadth of knowledge possessed by humans in order to appear intelligent in a training simulation. Rather, they only must appear to behave as a reasonable human opponent would in a very specific and narrow domain (i.e., submarine warfare, driving an automobile). In fact, this may mean that they should not display optimal behavior, as that is not always typical under wartime stress. It is our belief that applying Context-based Reasoning as described below presents a highly effective and efficient methodology for imparting sufficient intelligence to AIP's so as to achieve their objective in a training simulator.

CxBR has some similarity in concept to the frame-based adaptive behavior mechanism described by Morgan [1993]. However, CxBR is significantly simpler and more flexible to implement than is the latter, keeping with our goal of simplicity and conciseness.

2.1 Representation of Tactics through Contexts

In CxBR, contexts are the most important representational item. Much knowledge about how the AIP should behave, as well as to what other contexts it can transition is stored in the contexts themselves. Contexts are represented as classes which encapsulate functions to execute certain behaviors or actions typical of such contexts, and a definition of what to expect when in that context.

But tactical knowledge is complex as well as voluminous. This can make it difficult to manage efficiently. Furthermore, tactical knowledge can be segregated into several levels, corresponding to its generality and to its importance. Thus, the general idea of contexts is sub-divided hierarchically into three types: These are 1) the *Mission Context*, 2) the *Major-Contexts* and 3) the *Sub-contexts*. These will be described below.

2.1.1 Mission Contexts

A Mission Context (simply referred to as a *Mission*) is an overall definition of the objectives of the scenario. It defines the objectives as well as the constraints of the operation. The Mission can also define the things to avoid during the mission being undertaken. Examples of Missions in the domain of submarine warfare are **SEARCH-AND-DESTROY, MINING** a harbor or choke point, **GATE-KEEPING, BATTLE-GROUP-ESCORT, ANTI-SURFACE-OPERATION, SPECIAL-OPERATIONS**, and others. A Mission can define the types of lower level contexts that may be necessary during its execution. A Mission Context can also describe the political environment under which the mission is to be carried out. For example, if a shooting conflict is in effect, then the rules of engagement will surely be different than if in a cold war. This can influence which contexts are employed, as well as possibly how they are employed. No more than one Mission will be active at any one time, and Missions are mutually exclusive. So, a new Mission would have to bump an existing Mission from active status. While it is conceivable that in real life a submarine may be charged with simultaneous multiple missions, in practice, however, there would be little need to change Mission Contexts during the course of a training session.

The Mission can be used to define several factors associated with the execution of the mission, such as the constraints faced by the AIP and the things for it to avoid. These are described in the following attributes:

> **Constraints:** This attribute lists all the constraints that are imposed on the AIP during this mission. Some of these could be the weapon status (hold, tight and free) as well as any limitations placed upon the submarine's performance characteristics, because of, for example, damage.
>
> **Avoid:** This attribute describes anything that must be avoided at all times throughout the training scenario. One obvious one is destruction-of-self, but there may be others such as avoid counter-detection at all costs.

2.1.2 Major-Contexts

Major-Contexts are the main focus of the Context-based reasoning and representational paradigm. They contain all the necessary information to operate the AIP under major situations. They also contain the knowledge required to recognize when that Major-Context should be deactivated and another one put in its place.

A Major-Context represents a tactical operation undertaken as part of the Mission. Such tactical operations are designed to assist in achieving the goals set forth in the Mission. In this way, they can be used to plan the actions of the AIP during the Mission. For example, in a **SEARCH-AND-TRACK** Mission, consisting of patrolling a designated sector of ocean, the plan would be to sequentially activate the following Major-Contexts: 1) **Transit-to-sector** to reach the sector, 2) **Sector-Search** to patrol the sector until a contact is made with an opposing platform, 3) **Target-Track** to track the opposing platform until told to break contact, and 4) **Transit-Home** to navigate to home base upon completion of mission.

One Major-Context is always in control of the AIP. Major-Contexts are, also by definition, mutually exclusive of each other. Unlike the Mission, however, Major-Contexts would normally activated and deactivated many times throughout the course of a training session. A Major-Context is deactivated by retracting a flag that identifies it as the active Major-Context from a global data structure (often referred to in knowledge-based systems as a *fact base*). Another flag indicating the new active Major-Context is then posted to the global data structure, and sets into active status all actions associated with the new active Major-Context. This will permit the continuous monitoring of the simulation for indications of the need to transition to another Major-Context. Furthermore, the initialization message will also modify any parameters that requires modification, such as possibly the AIP's heading, speed, depth, etc.

Although some Major-Contexts may become invalid through the simple passing of time, this is not common. In most cases, the Major-Context will cease to be applicable because of either a change in the external situation, or the completion of its task. Therefore, Major-Contexts are generally assumed to be active indefinitely, until bumped from active status by another Major-Context.

Each Major-Context possesses the following important attributes:

Initializer: References the name of the message that is executed to initialize all required variables, whenever the Major-Context is first activated.

Compatible-next-Major-Context: This attribute lists those Major-Contexts to which transition from the current Major-Context is acceptable.

Sub-contexts: Sub-contexts (described below) are used to carry out all complex actions done under the auspices of a Major-Context. This attribute is a list of all Sub-contexts that are encapsulated within the current Major-Context. Any Sub-context not on this list would not be compatible with this Major-Context and therefore not capable of being activated by it. For example, it would not be advisable for a submarine to dive when in a narrow channel while entering a harbor. Thus, the **dive** Sub-context would not appear on the list pertaining to a **Navigating-channel** Major-Context.

Additionally, some Major-Contexts will have slots that are specific only to them, and deal with universal variables that need to be known throughout the entire simulation. For example, the **Attack** Major-Context will have a slot defining the target of the attack and another defining the number of weapons to be used. Likewise, the **Under-Attack** Major-Context will have a slot defining the aggressor (source of the weapons bearing down on the AIP).

2.1.3 Sub-contexts

Sub-contexts are lower-level tactical actions that are typically carried out as part of a Major-Context. They represent procedures not directly critical to reaching the Mission objectives. Sub-contexts are finite in duration and typically, but not necessarily, of short duration. They are associated with one or more Major-Contexts but are mutually exclusive of one another. If a new Major-Context is activated while an incompatible Sub-context is active, that Sub-context is immediately deactivated. It is not necessary for one Sub-context to be active at all times, as is the case with Major-Contexts. An example of a Sub-context in the submarine warfare domain is **submerge**, which constitutes the closing of hatches, filling of ballast tanks and the downward angling of the planes. Since one possible result of **submerge** could be to descend deeper than the crush depth of the submarine, this Sub-context also monitors the depth continuously as part of its function. The result is a continuous increase in depth until the desired depth is attained. Of course, in a simulation, it is only necessary to represent the resulting depth change at the appropriate diving rate. On the other hand, an **emergency-dive** Sub-context would do the same but much quicker with a much steeper dive angle.

In summary, a Mission for a submarine in wartime could be described as **SEARCH-AND-DESTROY**, with Major-Contexts such as **Sector-Search, Approach-Target**, and **Fire-Weapons,** among others. Sub-contexts of the **Fire-Weapons** Major-Context could be described as **raise-periscope**, **energize-active-sonar**, **submerge, flood-tubes**, and **surface**.

2.2 Control of AIP through Contexts

Controlling the actions of an AIP is the main purpose of CxBR. This section describes how CxBR makes use of the context hierarchy described above to define the behavior of an AIP.

Mission contexts largely serve to define the goals and other parameters of the mission to be carried out. They assist in planning the mission by listing the Major-Contexts that are required to carry out the mission. While other Major-Contexts may be used during the simulation if the situation arises, these required Major-Contexts are necessary, although not necessarily sufficient for the mission defined. While a Mission Context indirectly controls an AIP through this set of required Major-Contexts, it does so from a much higher level.

The Sub-contexts, on the other hand, represent low level procedures that are auxiliary in nature to Major-Contexts. They do not have the proper perspective of the situation to be the main control element of the AIP.

Major-Contexts have the right level of perspective for controlling the AIP, and are thus the main control element. They rely on the Mission Context to determine the plan through the set of required Major-Contexts as well as their sequential activation, and on the Sub-contexts to abstract the relatively complex (yet auxiliary) procedures that are applicable under that Major-Context.

Upon activation, a new Major-Context will do the following for the AIP to which it is attached:

1) Immediately send an initialization message to the AIP. This message incorporates the immediate modifications to the behavior of the AIP that accompany the new Major-Context. This message could consist of activating a Sub-context, a sequence of Sub-contexts, or directly invoke simple actions that do not require the use of Sub-contexts, such as modifying the speed, depth and/or heading of the submarine towards a new destination.

2) Monitor the simulation environment to determine when one or a sequence of Sub-contexts is to be activated (if they are not to be activated upon initialization).

3) Monitor the simulation environment to determine when it becomes necessary to deactivate itself and transition to another Major-Context.

2.3 Situation Assessment and Transitions between Contexts

The above section described the procedure that governs how the appropriate contexts (the term "context" used alone represents a generalization for Major- and Sub-contexts) are activated to effect control of an AIP's action. This section will describe <u>which</u> context is to be activated. More specifically, it explains how a context is determined to be no longer valid and how to determine which one is to be activated to replace the newly irrelevant one. This deals with the situational assessment aspect of Context-based Reasoning.

The transition among the various contexts is a critical issue in CxBR. Each Major-Context has a set of rules, called *monitoring rules* that are used to identify when a transition to another Major-Context is indicated by the situation. Sub-contexts have similar rules, except that they only check for completion of the Sub-context action(s). Monitoring rules will fire continuously (every simulation cycle) as long as it's associated (as well as enabling!) Major-Context or Sub-context is active. In its right hand side, monitoring rules have conditional statement(s) that monitor the parameters in the simulation that are relevant to the continuation of the context. Examples of these parameters are: whether **ownsub** has been detected, whether it is moving towards or away from **opsub**, whether it is within firing range, whether it is being attacked, etc. Once any parameter is satisfied indicating a change of context, the monitoring rule retracts the flag that allows the present context to be the active context, thus deactivating it. This disables the monitoring rules associated with the defunct context.

The decision of which context to activate next can be made in one of two ways:

1) The *direct transition* approach: Transition information is encoded into the monitoring rules themselves. This allows them to directly enable the new context by calling its initializing message that will, in turn, announce the newly activated context.

2) The *competing context* approach: A competition between valid contexts is used to determine which one best addresses the key issues of the emerging situation.

In either case, the monitoring rule(s) for the newly activated context begin monitoring immediately upon its activation.

Monitoring rules can be implemented in a pattern-matching rule-based system. These rules will have a pattern in their premise that indicates the active Major-Context (and/or Sub-context) to which they are applicable. Only when there is a fact in the fact base indicating the active status of the appropriate Major-Context (and/or Sub-context) will these rules be "active" and capable of firing.

As previously discussed, the foundation of CxBR is that the currently active Major-Context limits what the AIP can expect to experience. Thus, the number of monitoring rules associated with any one context is limited. This greatly facilitates the task of situational assessment. Transitioning to another Major-Context is the manner by which the CxBR system addresses the current situation. This is a simple, yet effective means of doing situational assessment under CxBR.

In some cases, the transition to a new context would be the result of an "external" message (e.g., a communication from fleet command). Examples of this would be a message to return home, or go to periscope depth to receive a more detailed communication. This external message is represented as a fact that has a "prompt" indication (i.e., (communication prompt)). In such circumstances, an additional rule is required, which fires only once, to retract the current Major-Context fact (as well as the prompt fact) and to invoke the initializing message that activates a new Major-Context or Sub-context.

Transitioning to new Sub-contexts is somewhat easier. This is because the monitoring rules for Sub-contexts simply check to determine whether the Sub-context action has been completed. For example, in a **submerge** Sub-context, the criteria for deactivation is when the desired depth has been attained.

There can also be *universal monitoring rules* that are not associated with any specific context. These rules continuously search for situations that could occur under any circumstances, such as being fired upon by an aggressor, or the detection of an opponent. Such universal rules are not incorporated into any particular contexts, and are active during all phases of the exercise.

Lastly, in an effort to introduce the ability to reason temporally, the idea of a previous-context and previous-sub-context has been implemented. These represent the Major-Context and Sub-context that were active when the presently active ones bumped them off active status. If applicable, the AIP may re-activate the previous contexts as may be called for by the situation.

The following figure (Figure 1) provides a sketch of how CxBR is structured and its interaction with the AIP object.

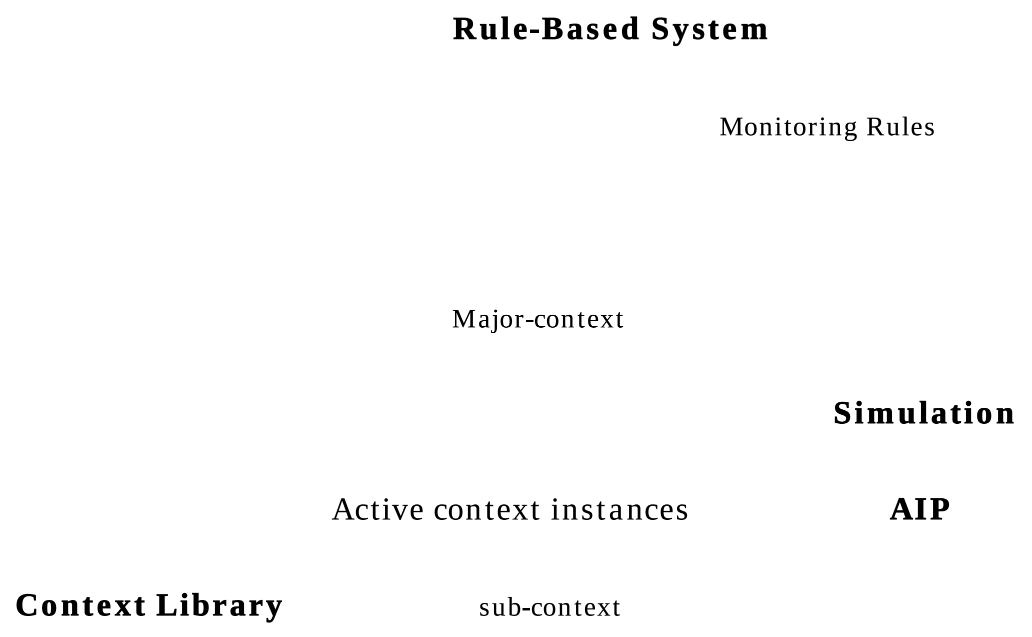

Figure 1 - General CxBR System Diagram

3. EXAMPLE OF CxBR APPROACH

To properly explain the ideas set forth in this article, we will describe here an application of CxBR to one-on-one submarine warfare. A prototype was built to control one AIP submarine in the process of carrying out a patrol Mission.

Note that the submarine warfare tactics described here represent the authors' interpretations of the contents of unclassified material [Khboshch, 1990; Clancy, 1985, 1986, 1993], and were not reviewed by subject matter experts. The mission represented is called **SEARCH-AND-TRACK**.

This prototype was used to gain experience with the CxBR approach and to see how easily a tactical knowledge base can be developed for this highly realistic Mission. Thus, its purpose was simply to determine whether CxBR is a feasible means of representing tactical behavior. It implements the Context-based approach using a combination of objects and a production system. It should be noted that it does not fully adhere to object-oriented programming concepts, as its purpose was simply to apply the concept of CxBR for the purposes of evaluating its effectiveness. As implementation issues are not the central focus of our work, this was not considered to be detrimental to the investigation.

The AIP that is to behave intelligently through CxBR will be referred to as the *opponent submarine*, or ***opsub.*** The trainee's platform, on the other hand, is referred to as ***ownsub,*** as is the custom in the U.S. Navy. **Ownsub** depends on the human trainee to control its actions, and has no intrinsic intelligence. The objective of **opsub** is to assist in the training of the trainee that controls **ownsub** by representing an intelligent unmanned opposing force. The scenarios described will focus on **opsub** and how it adapts its behavior automatically in response to the situation.

Opsub and **ownsub** are implemented as object instances (of class SUBMARINE). Their *static* slots (defined as those whose values will not change during the simulation) define their capabilities. Examples of these are their maximum speed, quiet speed, maximum depth, periscope depth, weapon systems, (e.g., number and ranges of torpedoes and missiles), sensors (e.g., range and types of passive sonar, active sonar, and towed arrays), and Electronic Warfare capabilities (e.g., sonar decoys). Additionally, their *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 information, as well as the supply of weapons and food, are also found in dynamic slots.

The data containing all of the external information relevant to the mission are found as slots in various other class objects. For example, if a mission calls for a sector to be patrolled as part of the **SEARCH-AND-TRACK** mission, then a sector object is instantiated that contains the points in the ocean that define the area to be patrolled. It may also contain other information related to the waters within that sector, such as the depth, the location of the thermal layer (the *thermocline*), and the location of any known underwater geological formation or shipwreck that can affect **opsub**'s ability to navigate that sector.

Weapons (e.g., torpedoes, missiles, mines) and evasive devices (e.g., sonar decoys) are also instances of classes that describe the specific device. They have member functions that move them towards the target or weapon. These functions also use probability functions that decide whether they will successfully destroy their target or not. This probability is based on the presence of evasive maneuvers and deployment of decoys.

Visibility of data to the two opposing sides (**opsub** and **ownsub**) is related to a combination of several factors. Some of these factors include the distance, speed, and depth with respect to the other submarine as well as to the thermocline (a layer of water with high temperature gradient that has the effect of reflecting sound), and the sensors being employed. In cases where there is only one other submarine to contend with (e.g., **ownsub**), a set of functions attached to the SUBMARINE class decide when the data are to be "seen" by its AIP. The right to see these data is a result of a computation using the related factors mentioned above.

Monitoring the simulation for changes in the situation and implementing the resulting context transition is carried out with the help of a pattern-matching, forward reasoning production system. All information that needs to be acted upon immediately upon occurrence is posted in the production system fact base.

If a change of mission is dictated by the external inputs (e.g., new orders from fleet command), then the fact that enables the old mission is retracted from the fact base, and one indicating the new one is asserted. However, as mentioned earlier, this is not expected to happen often.

When a particular Major-Context is in effect, it is considered to be the *active-major-context.* If the Mission is replaced, however, its active-major-context must also be deactivated. When a mutually exclusive Major-Context is to be activated onto an AIP, the presently active Major-Context is deactivated and considered to be the *previous-major-context*. This need only be done if there is reason to believe that the AIP should resume the current actions upon completion of the new Major-Context. This feature introduces the ability to reason temporally when it is important to know what **opsub** was doing previously.

The activation of a Sub-context that is compatible with the active Major-Context is done through the active Major-Context by posting a fact onto the fact base that indicates that that particular Sub-context is now active.

When one of the monitoring rules associated with the active Major-Context determines that that active Major-Context should be deactivated, it first fires a rule that retracts the fact advertising it as the active-context from the fact base. Secondly, in the case of fixed transition, it proceeds to send an initialization message to the class object for the new Major-Context to be activated. Which Major-Context to activate is determined by the monitoring rules themselves. The initialization action of the newly activated Major-Context begins by posting the appropriate active-context or active-sub-context fact onto the fact base to enable the new activation. Secondly, it implements any changes in the control variables of the AIP that may be required by the new Major-Context. In the case of **opsub**, this may indicate a change in depth, speed or heading, or to fire its weapons or its defensive countermeasures.

3.1 The SEARCH-AND-TRACK Mission

The **SEARCH-AND-TRACK** Mission focuses around **opsub** (the CxBR-driven AIP) patrolling a specified rectangular sector of open sea until told to return to base port. If it detects an opponent (namely **ownsub**, which is driven by the student), **opsub** is to track it from its rear (called its *baffles*) until either it loses it or **opsub** is recalled by fleet command (in reality, the instructor). The objective of this mission is not to destroy **ownsub**, but rather to keep it under surveillance. Nevertheless, unprovoked attacks by **opsub** upon **ownsub** are permitted, but a higher authority (i.e., the instructor) must order them. If counter-detected by **ownsub**, **opsub** is to break contact and evade the opponent. If attacked, it is to escape the in-coming weapons and then counter-attack. **Opsub** must always seek to avoid counter-detection by **ownsub** as well as its own destruction.

This mission has nine Major-Contexts and eight Sub-contexts associated with it. The plan for executing this Mission (the set of required Major-Contexts) consists of five Major-Contexts: **Transit-To-Sector, Sector-Search, Maneuver-Into-Position, Target-Track**, and **Transit-Home**. Figure 2 depicts the transitions among all the Major-Contexts and Sub-contexts within the **SEARCH-AND-TRACK** Mission. These contexts will be briefly described below.

3.2 Major-Contexts in the SEARCH-AND-TRACK Mission

The first five (5) Major-Contexts described below are found in the plan to execute the **SEARCH-AND-TRACK** mission. The other four (4) tactics may be required by opsub in response to abnormal circumstances, but unlikely to be needed in this mission. The Major-Contexts, with a short description, are as follows:

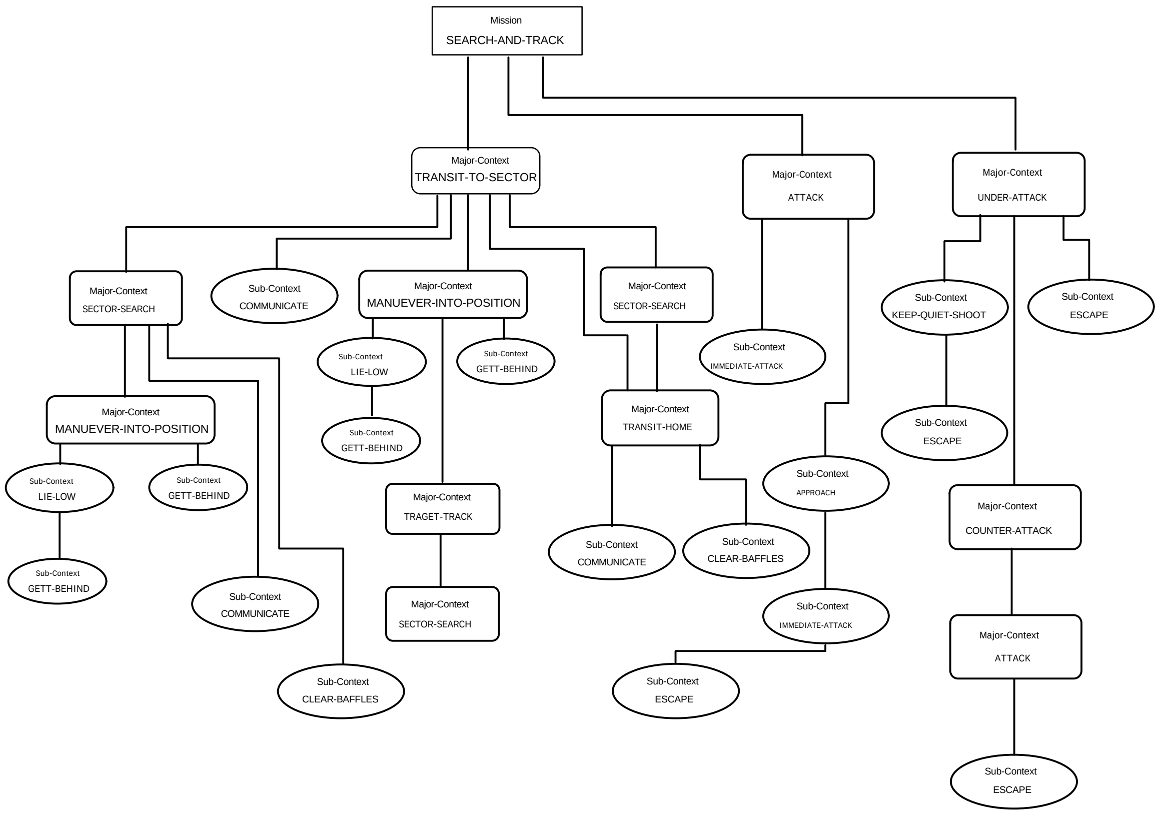

Figure 2 - Graphical Depiction of the Tactics and the Transitions

1) Transit-To-Sector: This context sets course and speed to get **opsub** to its designated sector.

2) **Sector-Search:** Causes **opsub** to search the sector until it finds **ownsub** or is recalled home.

3) **Maneuver-Into-Position:** Once **ownsub** has been found, this context puts **opsub** into position to track **ownsub** by "getting on its baffles".

4) **Target-Track:** Causes **opsub** to follow the detected **ownsub** wherever it goes.

5) **Transit-Home:** Guides **opsub** back to its home base.

6) **Attack:** Gives **opsub** the ability to fire its weapons at **ownsub** when permitted.

7) Escape: Causes **opsub** to move away from **ownsub** as quickly as possible, in a direction opposite to **ownsub's** bearing.

8) **Under-Attack:** Will cause **opsub** to evade the attack by maneuvering and releasing decoys.

9) **Counter-Attack:** Causes **opsub** to attack the aggressor **ownsub** after its own threat has ended.

3.3 Sub-contexts in the SEARCH-AND-TRACK Mission

The Sub-contexts used in this prototype are the following:

1) sprint-and-drift: Used when there is a need for **opsu**b to get to a destination fast, yet, also be aware of the surrounding environment.

2) lie-low: Causes **opsub** to go very quiet until **ownsub** has passed.

3) get-behind: Causes **opsub** to get behind **ownsub**.

4) **keep-quiet-and-shoot:** Activated when, upon initial detection of an attack, the distance is judged to be too short for a successful escape.

5) **communicate:** Causes **opsub** to go to periscope depth to send and/or receive a message.

6) **clear-baffles:** Causes **opsub** to suddenly and radically change direction to ensure that no one is behind its baffles.

7) **approach:** Gets **opsub** into position to attack **ownsub**

8) **immediate-attack:** causes **opsub** to launch weapons at **ownsub**

An example scenario was carried out on the prototype in which **opsub** sprints and drifts to the assigned sector and carries out as sector search. After an arbitrary period of time, it detects **ownsub** and gets behind its baffles to track it. It tracks it for an arbitrary period of time, clearing its own baffles periodically. Upon command by fleet headquarters, it attacks **ownsub** and then attempts to escape. **Ownsub** launches a counter-attack and **opsub** attempts to evade it. Depending on the success of both attacks (decided on a random basis), **ownsub** and/or **opsub** are either destroyed or escape. If **opsub** escapes, it is told to return home, and it does. Figure 3 shows a time-line describing the sequence (but not the duration) of events, both at the Major-Context and the Sub-context level.

3.4 Discussion of Example

The prototype was implemented in CLIPS 6.04. All Missions, Major-Contexts and Sub-contexts were represented as classes in the CLIPS Object Oriented Language (COOL), as were the SUBMARINE class and the various weapon classes. Monitoring rules were implemented as CLIPS rules. Initialization messages were implemented as message-handlers in COOL. The output is in the form of text and is displayed every five seconds. The instructor can interrupt the simulation (the execution of CLIPS) in order to introduce a measure of control over the AIP. Some of the interactions allowed through this interruption included orders to attack, communication prompts, baffle-clearing prompts, and orders to return home.

Figure 3 - Prototype Scenario Time-line

Major-contexts and sub-contexts were activated in CLIPS by asserting a fact that indicated the activation of the context (mission, major- and sub-). For example, the active Mission context is represented in CLIPS as the fact:

(mission-context **SEARCH-AND-TRACK**)

Likewise, activation of a major-context was done by asserting the following fact:

(active-major-context **Sector-Search**)

Activation of a Sub-context is done with the active-sub-context fact:

(active-sub-context **clear-baffles**)

Retracting these facts, of course, has the effect of deactivating the contexts. A new fact is asserted immediately thereafter, announcing the newly activated context(s). The situations faced by the AIP are also identified through facts that are posted to the fact base by the simulation part of the system every 5 seconds. The old facts are simultaneously retracted.

This CLIPS-based environment proved to be an adequate framework for the task, but certainly not ideal. It was quite effective in carrying out the reasoning required to determine which context was to be made active. The availability of COOL was also a significant advantage in representing the contexts as well as the submarines themselves. However, CLIPS is a rather cumbersome tool with a lot of overhead. Furthermore, it was also used to do the simulation of the submarines, which it did not do well. However, in its defense, it should be noted that CLIPS was not designed to do simulations, and it did perform well in the tasks that it was designed to do.

However, the general-purpose aspect of CLIPS and its corresponding heavy overhead do not make it an ideal tool for ultimate use in implementing CxBR. It would be a problem if one copy of CLIPS were to be necessary to represent each of several AIP's in a large combined arms simulation. A better approach would be to develop a special purpose, concise inference mechanism that can be made resident in each of the AIP instances themselves. It would approximate the human model of each intelligent entity having its own personal inference mechanism (i.e., brain). This limited inference mechanism would simply make instances of the context objects, and allow them to control the AIP as long as they are active.

4. EVALUATION OF CONTEXT-BASED REASONING

The objectives of the evaluation task were to determine whether CxBR represents a viable representation and execution paradigm for use in tactical behavior of simulated entities. This consists of the following aspects:

1) Face validation of behavior - Does **opsub** behave as it should under a variety of realistic circumstances? This will evaluate the effectiveness of CxBR in representing and manipulating tactical knowledge.

2) Quantitative Validation - Does the use of CxBR represent an advantage over other approaches? We compare the prototype described above to another prototype to determine its overall conciseness. Furthermore, we built and evaluated a second prototype in the automobile-driving domain to determine whether CxBR is efficient in representation and computation. This prototype is compared to a system that addresses the same problem using a rule-based expert system.

These efforts are described below.

4.1 Face Validation of SEARCH-AND-TRACK Prototype

The first question faced was: Can CxBR be effectively used to represent tactical behavior? In order to answer this question, we subjected the prototype **opsub** described in Section 3 to several different situations within the context of the **SEARCH-AND-TRACK** Mission and recorded its reaction. These various circumstances called for application of all the specific tactics in **opsub's** behavioral repertoire (context library). These situations were brought about by controlling **ownsub**'s location, bearing, speed, depth and use of weapons, as well as **opsub**'s orders.

Test #	Situation	Description	Success
1	Transit1	**Transit-To-Sector** doing a **sprint-and-drift**, and **clear-baffles**. No contact with ownsub	**yes**
2	Transit2	**Transit-Home**, doing a **sprint-and-drift**, and **clear-baffles**. No contact with ownsub	**yes**
3	Detect1	**Transit-To-Sector** doing a **sprint-and -drift**, **clear-baffles** and making contact with ownsub prior to reaching sector. Immediately goes into **Maneuver-Into-Position** and **Target-Track.**	**yes**
4	Detect2	**Transit-Home** doing a **sprint-and-drift** detects ownsub and simply **communicates** its position and heading.	yes
5	Detect3	Reaches sector after a **Transit-To -Sector**, and begins **Sector-Search**. Finds ownsub and executes **Maneuver -Into-Position** using **get-behind. T**hen tracks ownsub indefinitely.	yes
6	Detect4	Variation of 3 with different relative position to ownsub which results in the use of **lie-low** as well as **get-behind**.	yes
7	Attack1	Attacks ownsub from "on-baffles" position, then **escapes** immediately.	yes
8	Attack2	Attacks ownsub from "behind" position, then **escape**s immediately.	yes
9	Attack3	Attacks ownsub from "in-position", then **escape**s immediately.	yes
10	Attack4	Attacks ownsub from "ahead" position, then evades immediately.	yes

11	Udr-atk1	**Under-Attack** by ownsub causing it to use the **escape** Sub-context.	yes
12	Udr-atk2	Under-Attack by ownsub causing it to activate the **keep-quiet-and-shoot** Sub-context.	yes
13	Udr-atk3	Under-Attack by ownsub causing it to use escape Sub-context, and then **Counter-Attack**ing ownsub	yes

Table 1 - Test Situations Imposed upon opsub in SEARCH-AND-TRACK

Table 1 indicates the situations to which **opsub** was subjected under this part of the evaluation procedure. As the objective of the research is not to implement tactically correct submarine behavior, but rather, to evaluate the CxBR concept, we did not deem it important to ensure that the behavior displayed was tactically correct. We only evaluated whether **opsub** behaved as expected by its developers, not whether the behavior was tactically accurate or not.

It can be seen from Table 1 that the face validation phase of this evaluation was successful. This allows us to conclude that the CxBR paradigm can be used to effectively represent the tactical behavior of an AIP from a qualitative standpoint. This conclusion is an essential one, since inability to be used to represent tactical behavior would invalidate the CxBR paradigm without the need for any further evaluation.

4.2 Quantitative Evaluation

This phase of the validation process sought to show that CxBR represents a more concise means of building AIP models from the development effort standpoint as well as from that of computational efficiency.

This required the development of a prototype rule-based system that recommends a course of action for a simulated automobile in a driver training simulation [Brown, 1994]. We compared the internal composition of this rule-based prototype with that of the submarine prototype described in Section 3.

This prototype used an existing driver simulator system [Klee, 1991] and implemented a simple and short scenario where the AIP automobile (labeled as *aip-car*) is cruising on a two-lane road near an intersection where another car (labeled as *sim-car*) is waiting to turn left ahead of it. To complicate matters, a van (*sim-van*) is coming around a curve just beyond the intersection in the opposite direction. Figure 4 graphically depicts a bird's eye view of the scenario faced by aip-car (called the "student's car" in this figure). Aip-car is tasked with avoiding a collision with both sim-car (called "simulator car") and sim-van (called "simulator van"), as sim-car attempts to cut in front of aip-car and the approaching sim-

Figure 4 - Collision Avoidance Scenario

van. It should be noted that aip-car's task is simply to recommend the action to be taken, not to actually implement the control, as the latter is impossible to do in the test-bed simulator.

The scenario is varied by running various tests with different "release distances" for sim-car and sim-van. Release distances are defined as the distance away from aip-car at which sim-van appears around the bend and sim-car initiates the left turns into the path of aip-car. This meant that the distance available for aip-car to maneuver ranged from one where no real danger was present, to one where a collision was practically inevitable. It should be noted that sim-car and sim-van have no intelligence and continue on their paths without concern for the imminent collision. The choices for aip-car were the following: 1) Do nothing if the distance was great enough that no danger was present. 2) Slow down if that was to be sufficient and necessary to avoid a collision. 3) Hard braking if that was to be necessary and sufficient to avoid a collision. 4) Brake and veer off the road to the right if none of the other maneuvers were sufficient.

In the evaluation of the submarine prototype for conciseness, we count the number of elements that make up the Section 3 prototype, and compare this to the number of elements used in the entirely rule-based driver-training prototype. This is shown in Table 2 below.

Prototype Elements	**Submarine**	**Driver-rule-based**
Classes	28	3
Functions	33	5
Message Handlers	21	n/a
Monitoring Rules	31	30

Table 2 - Element Comparison between CxBR and rule-based Approach

The significant difference in the number of classes, functions and message handlers speaks to the vast difference in scope between the two prototypes. The Submarine prototype is capable of representing a much broader as well as deeper set of behaviors than is the driver-training prototype with its limitations to one rather constrained scenario. However, the telling number in Table 2 is that in spite of this great difference in scope, the number of rules is practically the same.

In the evaluation for efficiency, we focus on the situational assessment task. As the scope of a system increases, this task will become the most computationally expensive of all because each rule will have to have its premises checked during every simulation cycle to determine its applicability. Therefore, the situational assessment task in a rule-based behavioral model will become more and more computationally intensive as the number of rules grows. This task can be said to be O(n), where n is the number of rules in the system. CxBR, on the other hand, is more efficient because it not only limits the number of total rules that are written, but also eliminates the need to check the premises of those rules whose associated context is not active. The number of rules that are actually checked during each context is O(k), where k is the average number of rules attached to the active contexts. The significant issue is that k is independent of the total number of rules or of contexts in the system, as only the rules defined within a context are actually active – a small fraction of the total number of rules. For example, the total 17 contexts used in the Submarine prototype (9 Major-Contexts and 8 Sub-contexts) constitutes an average of slightly less than 2 rules per context. Because only one Major-Context and (at most) one Sub-context are active at any one time, on the average, only four rules (at most) will be active at any one time. This is nearly an order of magnitude less than the entire 30 rules being active all the time in the driver training simulation.

This is shown by the comparison of the driver-training rule-based prototype to a CxBR prototype that addresses the same problem as the former - collision avoidance for aip-car in the same scenario. This CxBR prototype does not completely adhere to the CxBR formalism. That is the reason why its composition was not used in the conciseness comparison. However, it does segregate the rules by active context, thus serving to show that this concept results in improved efficiency, even when the number of rules used is actually greater. Table 3 depicts the results of this phase of the evaluation.

Criteria of comparison	**CxBR Implem.**	**Rules-only Implem.**
CLIPS Rules	34	30
Total CLIPS Elements	53	43
Execution time avg. (sec)	124.1	126.91

Table 3 - Summary of Quantitative Evaluation of CxBR

More recently, Grejs [1998] expanded on the work of Brown by developing a more robust intelligent automobile model that is capable of complex behaviors in several different situations. He also used CLIPS as its inference mechanism. Grejs concluded that the CxBR approach was able to more concisely represent behaviors in this domain when compared to other techniques. See Grejs [1998] for details.

5. SUMMARY AND CONCLUSIONS

The use of contexts to represent and reason with tactical knowledge has the advantage of encapsulating all facets of such knowledge as it applies to a small slice of the entire domain knowledge. By modularizing the knowledge in such a way, and by explicitly expressing the relationships between the various contexts such that the transitions between contexts are inherently limited, efficiencies can be implemented. These efficiencies are in terms of knowledge acquisition as well as in execution of the system. By and large, CxBR was successful in achieving its objectives.

One weakness in the design of the prototype was the direct transition between contexts. It is difficult in many cases to determine at runtime to which new context/sub-context to appropriately transition. The direct consequence of this is that "pre-chaining" of contexts/sub-contexts had to be done, as depicted in Fig. 2. Some of the contexts had numerous possible next contexts, making such pre-determined transitions quite complex and inflexible. This is how the transitions between FSM's are done in the FSM-based CGF systems described in Section 1.2. While this may not be a problem for missions with a relatively small number of behaviors, it would seriously complicate the design of a CxBR system that encompasses numerous behaviors. A better approach in the future will be to implement the competing context method of transition discussed in Section 2.2 above. This would eliminate the need for the transition diagrams such as that of Figure 2. Instead, the context that best addresses the needs of the new situation would be the one to be made active.

The ability of the CxBR concept to serve in multiple AIP situations is essential if it is to be accepted as a useful means of representing behavior. As such, Proenza [1997] extended the work reported here to a multiple agent simulation in a mission called ESCAPE. ESCAPE is a variation of the submarine pursuit game originally described in Stephens [1990] and Singh [1993]. It involves a single submarine (red agent) attempting to escape four surface ships (blue agents) that are trying to capture it. In his results, the *success efficiency* (as defined by Stephens [1990]) that was calculated for the CxBR implementation, was 38% higher than that of the original distributed AI system. Refer to Proenza [1997] for details.

6. BIBLIOGRAPHY

[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.

[Brown, 1994] Brown, J. C., "Application and Evaluation of the Context-based Reasoning Paradigm", Master's Thesis, Department of Electrical and Computer Engineering, University of Central Florida, Orlando, FL, July, 1994.

[Calder, 1993] Calder, R., Smith, J., Coutemanche, A., Mar, J. M. F. and Ceranowicz, A., "ModSAF Behavior Simulation and Control", Proceedings of the Third Conference on Computer Generated Forces and Behavioral Representation, Orlando, FL, March 1993, pp. 347-356.

[Card, 1983] Card, S., Moran, T. and Newell, A., The Psychology of Human-Computer Interaction, Hillsdale, NJ: Lawrence Erlbaum Associates, 1983.

[Castillo, 1991] Castillo, D., "Toward a Paradigm for Modeling and Simulating Intelligent Agents," Doctoral Dissertation, Department of Industrial Engineering and Management Systems, University of Central Florida, December, 1991.

[Catsimpoolas, 1992] Catsimpoolas, N. and Marti, J., "Scripting Highly Autonomous Behavior Using Case-based Reasoning", Proceedings of the 25th Annual Simulation Symposium, Orlando, FL, April 1992, pp. 13-19.

[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.

[Clancy, 1985] Clancy, T., Red Storm Rising, New York: Berkley Books, 1985.

[Clancy, 1986] Clancy, T., The Hunt for Red October, New York: Berkley Books, 1986.

[Clancy, 1993] Clancy, T., Submarine: A Guided Tour Inside a Nuclear Warship, New York: Berkley Books, 1993.

[Danisas, 1990] Danisas, K. E., "Investigation of Simulated Opposing Force Model and a Method for Implementation", Master's Degree Research Report, Department of Industrial Engineering and Management Systems, University of Central Florida, 1990.

[Dean, 1995] Dean, C. J., "Semantic Correlation of Behavior for the Interoperability of Heterogeneous Simulations", Master's Thesis, Department of Electrical and Computer Engineering, University of Central Florida, May 1996.

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

[Dreyfus, 1986] Dreyfus, H. L. and Dreyfus, S. E., Mind over Machine, New York: MacMillan/The Free Press, 1986.

[Golovcsenko, 1987] Golovcsenko, I. V., "Applications of Artificial Intelligence in Military Training Simulations", Master's Degree Research Report, University of Central Florida, 1987.

[Grama, 1998] Grama, C., Gonzalez, A. J., Pollak, E., Brasch, R. and Wartsky, J., “ Automated Plan Generation for Mid-Echelons in the Military Decision-making Process”, Proceedings of the Spring 1998 Simulation Interoperability Workshop, Orlando, FL, March 1998.

[Grejs, 1998] Grejs, P.F., “Autonomous Automobile behavior through a Context-based Approach”, Master’s Thesis, Department of Electrical and Computer Engineering, University of Central Florida, Orlando FL, Summer 1998.

[Klee, 1991] Klee, H. I., "Development of a Low Cost Driving Simulator", Report, Department of Computer Engineering, University of Central Florida, 1991.

[Khboshch, 1990] Khboshch, V. A., Submarine Tactics, Voyenizdat, 1989, translation in 1990 by the Foreign Broadcast Information Service.

[Laird, 1994] Laird, J. E., Newell, A. and Rosenbloom, P. S., "Soar: An Architecture for General Intelligence", Artificial Intelligence, 33(1), 1987, pp. 1-64.

[Morgan, 1993] Morgan, P. D., “Simulation of an Adaptive Behavior mechanism in an Expert Decision Maker”, IEEE Transactions on Systems, Man and Cybernetics, 1993, 23(1), pp. 65-76.

[Olsen, 1990] Olsen, J. R. and Olsen, G. M., "The Growth of Cognitive Modeling in Human-Computer Interaction since GOMS", Human Computer Interaction, 5(2&3), 1990, pp. 221-265.

[Ourston, 1995] Ourston, D., Blanchard, D., Chandler, E. and Loh, E., "From CIS to Software", Proceedings of the Fifth Conference on Computer Generated Forces and Behavioral Representation, Orlando, FL, May 1995, pp. 275-285.

[Proenza, 1997] Proenza, R., "A Framework for Multiple Agents and Memory Recall within the Context-based Paradigm", Master's Thesis, Department of Electrical and Computer Engineering, University of Central Florida, Spring, 1997.

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

[Singh, 1993] Singh, M. P., Huns, M. N. and Stephens, L. M., "Declarative Representation of Multi-Agent Systems", IEEE Transactions on Knowledge and Data Engineering, Vol. 5, No. 5, October 1993, pp. 721-739.

[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.

[Stephens, 1990] Stephens, L. M. and Merx, M. B., "The Effect of Agent Control Strategy Performance of a DAI Pursuit Problem", Proceedings of the 10th Conference on Distributed Artificial Intelligence, Bandero, TX, October 1990, pp. 1-8.

[Tambe, 1995] Tambe, M., Johnson, W. L., Jones, R. M., Koss, F., Laird, J. E. and Rosenbloom, P. S., "Intelligent Agents for Interactive Simulation Environments", AI Magazine, Spring, 1995, pp. 15-39.

[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 36th Annual Meeting, 1992.

[Zachary, 1992] Zachary, W., Ryder, J. and Ross, L., "Intelligent Human-Computer Interaction in Real Time, Multi-tasking Process Control and Monitoring Systems", in M. Helander and M.

Nagamachi (Eds.) Human Factors in Design for Manufacturability, New York: Taylor and Francis, 1992, pp. 377-402.

[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.