This document is an author-formatted work. The definitive version for citation appears as:

A. Gallagher, A. J. Gonzalez, and R. F. DeMara, "Modeling Platform Behaviors Under Degraded States Using Context-Based Reasoning," in *Proceedings of the 2000 Interservice/Industry Training, Simulation and Education Conference (I/ITSEC-2000)*, Orlando, Florida, U.S.A., November 27 – 30, 2000.

Link: http://www.simsysinc.com/i_itsec00.htm

MODELING PLATFORM BEHAVIORS UNDER DEGRADED STATES USING CONTEXT-BASED REASONING

Anthony Gallagher, Avelino Gonzalez, and Ronald DeMara
University of Central Florida
Orlando, Florida

Abstract

The goal of military training simulators is to portray the realities of combat situations as closely as possible. During combat situations, the performance of military vehicles can sustain progressive degradation induced by a variety of factors that range from enemy fire to crew fatigue. Training simulators should model these degraded states in order to provide military personnel with realistic training environments. Unfortunately, current simulators use less than optimal techniques to model platform degradation. The current techniques are mostly based on *a probability of kill* (PK). As an example, the performance degradation of a tank is modeled by three states*: mobility kill, firepower kill*, and *catastrophic kill*. This model does not leave room for the myriad of degradation conditions that lie somewhere in between these three states, as well as not taking into account other system components, such as communication equipment, nor the degraded performance that can result from human factors unrelated to the state of the equipment such as crew stress and fatigue. Researchers at the Army Materiel Systems Analysis Activity (AMSAA) have developed a new model that proposes a *vulnerability and lethality taxonomy* (V/LT). This taxonomy serves as a much more realistically metric to describe platform degradation and its resulting consequences. Other researchers, principally Industrial/Organizational (I/O) psychologists, have been employed by the military to determine the influence of human factors in degraded platform behavior.

The purpose of this paper is to examine how to modify the behavior of autonomous intelligent agents (AIPs) given their current degraded state. The proposed method uses the Context-Based Reasoning (CxBR) paradigm to model AIP behavior. The AMSAA V/L taxonomy is incorporated into the model, and performance-degrading human factors are taken into account. To incorporate degraded state behavior into the CxBR paradigm, the current CxBR implementation was modified to incorporate the AIP's degraded state into its reasoning. The modifications changed the CxBR structure by including degraded state knowledge in the AIP fact database, and by altering the reasoning that CxBR uses to choose the appropriate next context. This reasoning is modified by adding weights to each context and functions that calculate these weights. The current context in the proposed implementation is chosen as the context that has received the highest weight. The proposed approach was tested using a small-scale tank warfare scenario with satisfactory results. Future work should implement the concepts presented in this thesis on a larger-scale scenario, and refine implementation details, such as finding optimal functions to calculate the context weights.

Biographical Sketch:

Anthony Gallagher is a Research Assistant at the Robotics Institute at Carnegie Mellon University in Pittsburgh, PA, where he is pursuing a PhD. His areas of interest include computer vision, image processing and artificial intelligence and their application to engineering problems. Anthony holds a Bachelor of Science in Computer Engineering and will graduate with a Masters of Science degree in Electrical Engineering from the University of Central Florida in December 2000. He served as a Research Assistant at the School of Electrical Engineering and Computer Science at UCF while performing the research described here.

Avelino Gonzalez is a Professor at the School of Electrical Engineering and Computer Science of the University of Central Florida. His main area of interest is artificial intelligence, specially as to how it applies to modeling human tactical behaviors. He has been the principal investigator in several DoD projects. He holds a Ph.D. degree in Electrical Engineering from the University of Pittsburgh.

Ronald DeMara is an Associate Professor at the School of Electrical Engineering and Computer Science of the University of Central Florida. Dr. DeMara specializes in distributed processing, artificial Intelligence, and simulation. He holds a Ph.D degree in Computer Engineering from the University of Southern California.

MODELING PLATFORM BEHAVIORS UNDER DEGRADED STATES USING CONTEXT-BASED REASONING

Anthony Gallagher, Avelino Gonzalez, and Ronald DeMara
University of Central Florida
Orlando, Florida

INTRODUCTION

Current simulators model platform degradation with very simplistic models that do not do justice to the vast variation of degradation conditions that can occur during combat. For instance, the mobility of a tank can be affected by several factors that can undergo degradation, such as the state of the track, the state of the engine, etc. Most of these factors, in turn, can undergo continuous levels of degradation. Both the state of the track and the state of the engine could be described by quantities that range from the minimum state (catastrophic loss) to the maximum state (fully functional.) Modeling the mobility of a platform as either mobile or not, as many simulators do, is a huge oversimplification of reality that can no longer be justified, given the increasing demands for realism placed on training simulators.

Recent efforts conducted by researchers from several fields have tackled several aspects of modeling platform degradation. These researchers attempt to produce more realistic models through which to categorize the damage sustained by a military platform. These new degradation models can then be used to give *computer-generated forces* (CGFs) more realistic behaviors given their degraded state. Two major communities that have addressed two large parts of the degradation-modeling problem are the Vulnerability/Lethality (V/L) community, and Industrial/Organizational (I/O) psychologists. V/L community researchers have created several V/L models throughout the years. These models are specific for every platform, and attempt to model the degradation on a military platform caused by enemy fire. Industrial/Organizational (I/O) psychologists, on the other hand, have developed models to approximate the influence of human factors in platform behavior. A third area that has not been thoroughly addressed is the degradation induced by normal platform use. It is evident, however that such factors (e.g. state of fuel or ammunition), influence crew decision-making. In this context, these factors can also be viewed as creating a degraded state that can influence the behavior.

Most current simulators *use Probability of Kill* (PKs) to model degradation. These metrics were produced by the Vulnerability/Lethality (V/L) community, beginning with World War II. Although they are the best-known metrics, many bear no direct relationship to observable field occurrences, and they don't address other factors not related to combat damage. Recent efforts by the V/L community have developed more realistic metrics to assess platform damage. AMSAA researchers, in particular, have led the field by developing a V/L Taxonomy that attempts to bring greater rigor and clarity to the discipline (Deitz et al., 1997). Other researchers, principally Industrial/Organizational (I/O) psychologists, have made progress in incorporating stress and fatigue as factors that produce degraded platform performance.

THE CONTEXT-BASED REASONING PARADIGM

Context-Based Reasoning (CxBR) is a knowledge representation paradigm developed to efficiently model the behavior of humans in tactical situations. It is especially suited for modeling of tactical behavior in military conflict [Gonzalez and Ahlers, 1998].

CxBR is based on the concept that humans think in terms of the context in which they find themselves. Furthermore, CxBR is also based on the concept of contexts controlling an intelligent agent's actions. A context does two things: 1) It defines the actions that the agent takes under the context; and 2) it defines to what other context can the control of the entity be passed when the situations changes such as to make the currently-active context no longer relevant. By doing these two things, contexts limit the things that can happen, thus obviating the need for an exhaustive search of all the possible actions, which is a drawback of rule-based systems. In short, contexts define the actions and events applicable in a given situation, thus limiting the scope of knowledge required [Gonzalez and Ahlers, 1998].

As an example, during tank warfare, the crew can expect certain things to happen while they are in a given situation, while not others. For instance, let's define a context called **Attack-enemy**. During this context, it is expected that the tank will be undergoing aggressive maneuvering, while firing its guns, and advancing towards the enemy. Certain events are expected, such as the enemy firing back at the tank. Other events would be unexpected, such as enemy soldiers appearing on the tank, and engaging in hand-to-hand combat with the crew. Therefore, the **Attack-enemy** context would incorporate the functionality necessary to carry out aggressive maneuvering and firing at the enemy. But it would not contain any capability for the crew to to do hand-to-hand combat. As such, the amount of knowledge contained in the context definition can be limited to what would be expected under the circumstances that define the context in question.

Contexts are divided in three main levels. On the top of the hierarchy is the *Mission Context*. The Mission Context contains the goal or orders given to the intelligent entity (or as we call it here, an *Autonomous Intelligent Platform*, or AIP). In the case of the tank example, a mission called **Movement-to-contact** could be given to the tank, thus providing it with an overall purpose. The mission defines the entity's objective, providing it knowledge of what its goal is, and when this goal has been reached. Mission Contexts are not control elements, but rather, only serve to define the parameters of the overall mission.

Below the Mission Context are the *Main Contexts*. Main contexts are the main control element for the AIP, and thus form the backbone of CxBR. They encapsulate the knowledge needed for the different situations that the AIP may encounter while striving to complete the mission. As an example, a tank with a **Movement-to-contact** mission could have as main contexts **Attack-enemy**, **Search-for-enemy**, **Tactical-Retreat**, and **Surrender**. Main contexts are mutually exclusive in the sense that one and only one can be in control of the AIP's behavior. This is called *the currently active context*, or simply the *active context*.

Below the main contexts are the S*ub-contexts*. Sub-contexts encompass lower level actions that are repeatable, reusable, and can be easily abstracted. For example, the **Attack-enemy** main context could have as its sub-contexts **Fire-main-gun**, **Advance-to-enemy**, **Pop-smoke**. There is no limit to the number of levels of sub-contexts that could be used. For instance, the sub-context **Fire-main-gun** could be further subdivided into the sub-contexts **Load-main-gun**, **Aim-main-gun**, and **Shoot-main-gun**. The context hierarchy is shown in Figure 1.

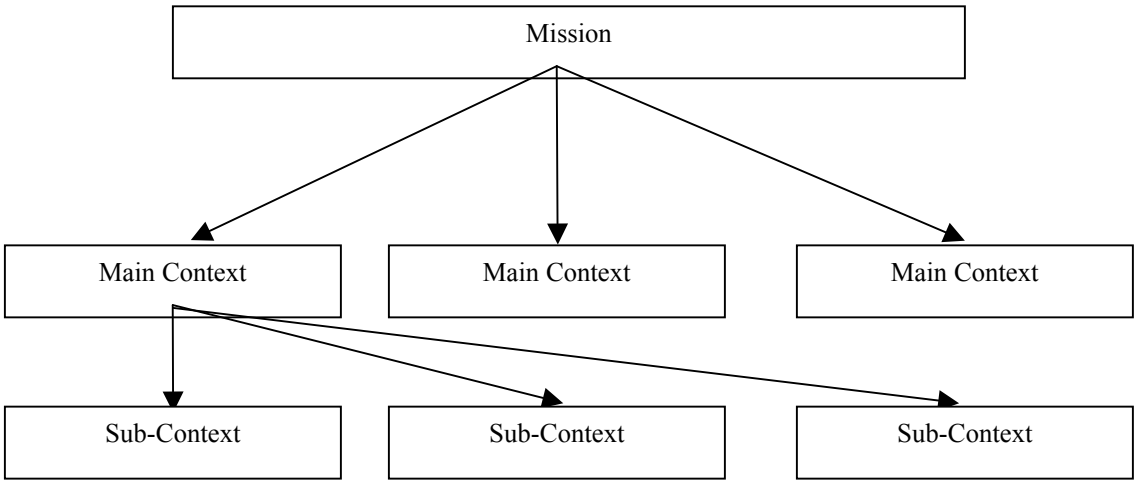

FIGURE 1- GRAPHICAL DEPICTION OF THE CXBR HIERARCHY

By dividing the knowledge base in a hierarchical fashion, CxBR facilitates development and enhances execution speed. Unlike rule-based systems where the knowledge base can become unmanageably large, contexts encapsulate the knowledge required, thus reducing the size of the rule base. A context is aware of the situation in which it is applicable, and other situations that might arise from that context, which gives the agent situational awareness [Norlander 1999].

Knowledge in CxBR is stored using modularity. Each entity has a local fact base containing the facts pertinent only to itself. All entities also have access to a global fact base, where facts that pertain to all entities and the simulation as a whole are stored. Each entity is equipped with an inference engine that makes use of these facts. The inference engine also checks the contexts in order to transition to one that is applicable to the current situation faced by the entity.

AMSAA's V/L Taxonomy

A new effort by AMSAA researchers led by Deitz is creating a V/L Taxonomy that aims to replace PKs and bring greater rigor and clarity to the V/L discipline. The V/L Taxonomy is composed of six levels of temporally ordered states known as mathematical spaces. Each mathematical space represents observable or derivable conditions of the threat and target platforms. The first few of the states represent conditions prior to weapon firing, while the latter few represent the condition of the target after the firing. The observable or derivable features of each space are represented in a vector, which can be characterized as the physical parameters of the platform and the environment (Gonzalez et al., 2000).

Two sequential spaces are joined together by a mapping function called an *operator*, which maps the vectors in the upstream space to one in the downstream space. These mappings can be done via physical equations, engineering design of the platform, or through operations research, depending on the two spaces being mapped. These mappings can be linear or non-linear in nature, and may be expansive (one to many), or contractive (many to one.) Figure 2 depicts the mathematical spaces and the operators; the six levels are the following (Deitz et al, 1997).

1. Weapon detection – identification conditions.
2. Threat-launch initial conditions.
3. Threat target initial conditions.
4. Damaged components in target.
5. Measures of target capabilities after taking the hit.
6. Measures of target utility after taking the hit.

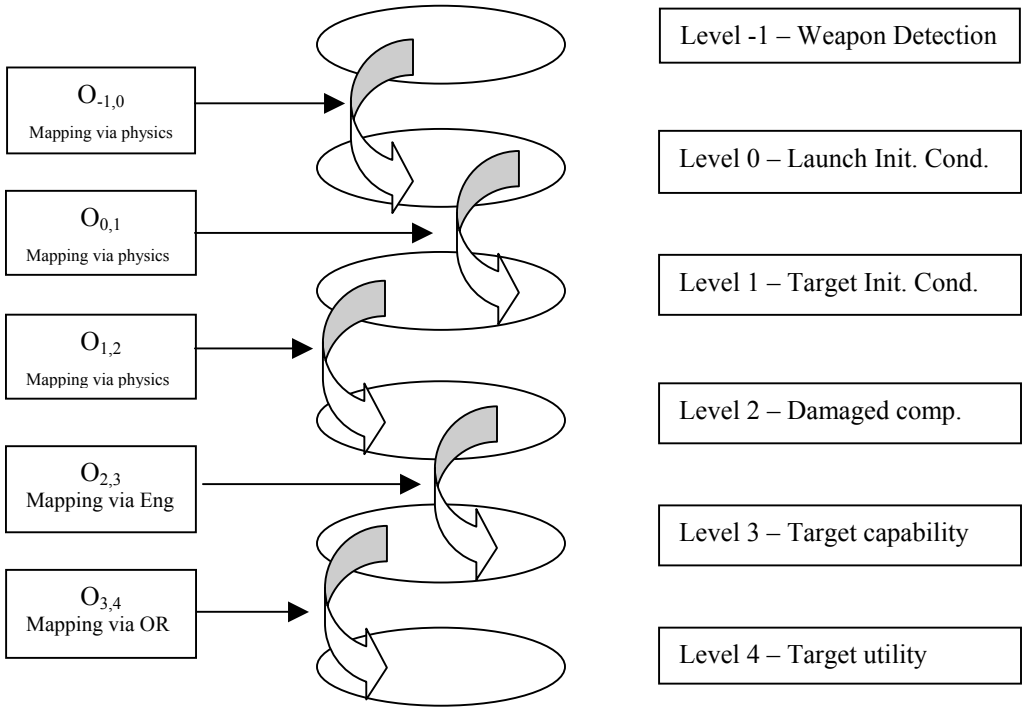

FIGURE 2 – GRAPHICAL DESCRIPTION OF THE MATHEMATICAL SPACES AND OPERATORS [GONZALEZ ET AL., 2000]

The V/L Taxonomy then separates the V/L metrics for aggregate damage, reduced platform capability, and reduced platform utility that had until then been incorporated into a single metric. It is shown that these three distinct and separable classes of metrics are linked by operators that are multivariate, stochastic, and nonlinear (Deitz et al., 1997). The current work will mainly utilize the last two levels, namely levels 3 and 4 that measure the target capability and the target utility and the mapping between these two levels.

See Deitz [1997] for further details on the V/L Taxonomy.

Further work has refined the V/L Taxonomy to include a Military Operations Context (MOC) block. This construct has been added in recognition that each of the operators takes input from the operations context in which a platform is performing. The MOC defines doctrine, tactics, leadership, materiel, scenario, terrain, weather, and all the factors external to the platform itself. As an example, during a live-fire test, the volatility of the ammunition is dependent on the ambient temperature (Deitz, 1999).

The V/L Taxonomy defines a rigorous method for implementing degraded state modeling, as opposed to DSWARS and CASTFOREM, both of which implement specific degraded state models. As such, the V/L Taxonomy is not incompatible with these two previous efforts, but rather can make use of the degraded states defined in these efforts. For example, the CASTFOREM degraded state tables can be viewed as implementing the V/L Taxonomy's Levels 1 through 3 and their respective mappings. These tables map the location of a hit on a target (Level 1) to a list of damaged components (Level 2), and give a degraded state for the platform capability (Level 3).

MODELING DEGRADED PLATFORM BEHAVIORS CONCEPTS

While the V/L Taxonomy addresses the effect of a weapon firing on the state of the target, it does not address the decision making process followed by commander of a platform that has suffered degradation as a result of enemy fire. Furthermore, enemy fire is not the only cause of degradation. The entity may find itself in a degraded state because of the psychological and physical state of the crew as well as that of normal wear and tear on the platform. In the work described here, the concept of degraded state behavior has been incorporated into the CxBR behavior paradigm in order to model this decision making process. Prior to this effort, the AIPs developed by researchers conducted themselves in an ideal fashion. They never made mistakes, were never damaged, and the crew never became fatigued. This lack of ability to represent degradation did not permit CxBR (or any other modeling paradigm, for that matter) to accurately depict the real world. In this real world platforms degrade, and for different reasons, the best decision is not always taken. This issue becomes especially important in military simulation applications. Military simulations need to represent the realities of combat, where platforms degrade due to factors such as enemy fire, crew fatigue, and others. Lacking the capability to infuse a platform with degraded state behavior invalidates any representation of human tactical behavior. This work, therefore, seeks to adapt CxBR to incorporate decision-making under degraded states.

CxBR applications are developed using the CxBR Framework, a custom-made application for CxBR. The previous CxBR Framework implementation lacked the following capabilities necessary for implementing degraded state behavior:

1. AIPs lacked knowledge of their degraded state.
2. It included no provision for dynamic degradation of the AIPs throughout the simulation.

To solve the first problem, an efficient and reusable manner of incorporating degraded state knowledge was needed. The chosen approach had to satisfy the following requirements:

- Degraded state knowledge should be accessible only to the degraded AIP.
- Procedures for creating, adding and modifying degraded state information had to be incorporated in the CxBR Framework to make them available for future researchers.

The second problem is more complex. As described above, the reasoning process in CxBR is done through context transitions. An AIP has a set of behavior that is specified by the context under which it is working (the current context). The major behaviors of the AIPs are specified by the Main Contexts. The Main Contexts in all previous CxBR applications were designed to be mutually exclusive. Transitioning among main Contexts was done through single-event conditions. For example, the detection of an enemy platform would be sufficient to transition from one Main Context to another, regardless of any other factor. But this is not realistic, as often, a decision between two or more plausible next Main Contexts may depend on several factors. One of these factors is indeed the platform's own degraded state. A way to choose the best

context possible had to be devised while meeting the following constraints:

- It should be easy to use and clearly demonstrate the effect of the AIP degraded state in selecting the most appropriate main context.
- The chosen method had to be flexible enough to allow for future researchers to implement methods for choosing the optimal context to control the AIP.

To demonstrate the added functionality of the CxBR Framework for degraded state behavior, the concepts had to be tested in a relevant application where the effects of degradation can greatly modify an AIP's behavior. The University of Central Florida (UCF) is part of a project to develop a military simulation environment where the vehicles involved display appropriate behaviors corresponding to their degraded state. The final prototype has to display the AIPs in a graphical environment, as well as display the interaction between the AIPs' degraded state and their behavior.

Project Hypothesis

The hypothesis of this project is that CxBR is a viable vehicle for representing degraded state behavior. To demonstrate this, the CxBR implementation was enhanced to address its lack of degraded state representation capability as described before.

The current project has made the following contributions to the development of the CxBR paradigm.

1. The CxBR Framework has been enhanced by incorporating degraded state knowledge into the AIPs' local factbase. This was done through a well-defined interface that allows the developer to easily create degradable qualities of the AIP, and give them a range of values. It also provides a preset value under which the AIP can no longer perform its function.
2. The CxBR Framework has been enhanced by a new context switching mechanisms that allows the creation of Main Contexts that are mutually compatible. The contexts themselves have been modified by the addition of new methods that allow the CxBR developer to give each context a weight that can be used to select the context that controls the AIP. These weight mechanism has been designed to allow for further research into optimal context selection.
3. A method for selecting context weights has been developed that emphasizes the effects of a platform's degraded state in context selection. Such a mechanism meets the project requirement of showing a correlation between the AIP's degraded state and its behavior.
4. A graphical simulation interface has been developed where the AIPs and their current degraded state are shown graphically. The simulation has been designed in VBA for easy reuse by future researchers with the appropriate technical knowledge.

CxBR Structure

During the development of the project, further enhancements were made to the CxBR Framework's capabilities and to its structure.

The CxBR structure has been restructured to provide for more modularity that will allow future developers to separate their CxBR application into separate pieces that contain related functionality. Depending on the application developed, it can be divided into two or three separate pieces. The first two are required and the third is optional if graphical display of the simulation is desired through the simulation tool developed in this project. The two required parts of the new CxBR application structure are the following:

- The CxBR Framework – The CxBR Framework has been developed into a stand-alone static library that can be use from any C++ CxBR application by adding the appropriate header files and including the library in the settings. This Framework has been revised to eliminate all of the functionality that required the use of a Windows compiler. The redesigned Framework is now written in standard C++ and can be compiled under any operating system with a C++ compiler that support the Standard Template Library (STL).
- The CxBR Simulation – The CxBR simulation links statically to the CxBR Framework and can be designed as a stand-alone application, or a Windows DLL that can be used from the simulation environment developed in this project. This simulation should contain the CxBR simulation itself, and if desired, the standard DLL calls for use by the simulation environment.

If a graphical simulation environment is desired, and the developer wishes to use the simulation environment developed in the current effort, the simulation will contain a third element:

- The simulation environment – The simulation environment was developed in VBA under PowerPoint 97. To use this environment, it is understood that the CxBR simulation was developed as a DLL using standard calling conventions. The simulation environment has been designed in a modular fashion with a set of core modules that interface with a Windows DLL, and a second set of modules that deals with the current application. Future researchers can make use of this environment by simply importing the core files into their VBA PowerPoint applications, and making sure that the correct set of interface methods are implemented in their DLL.

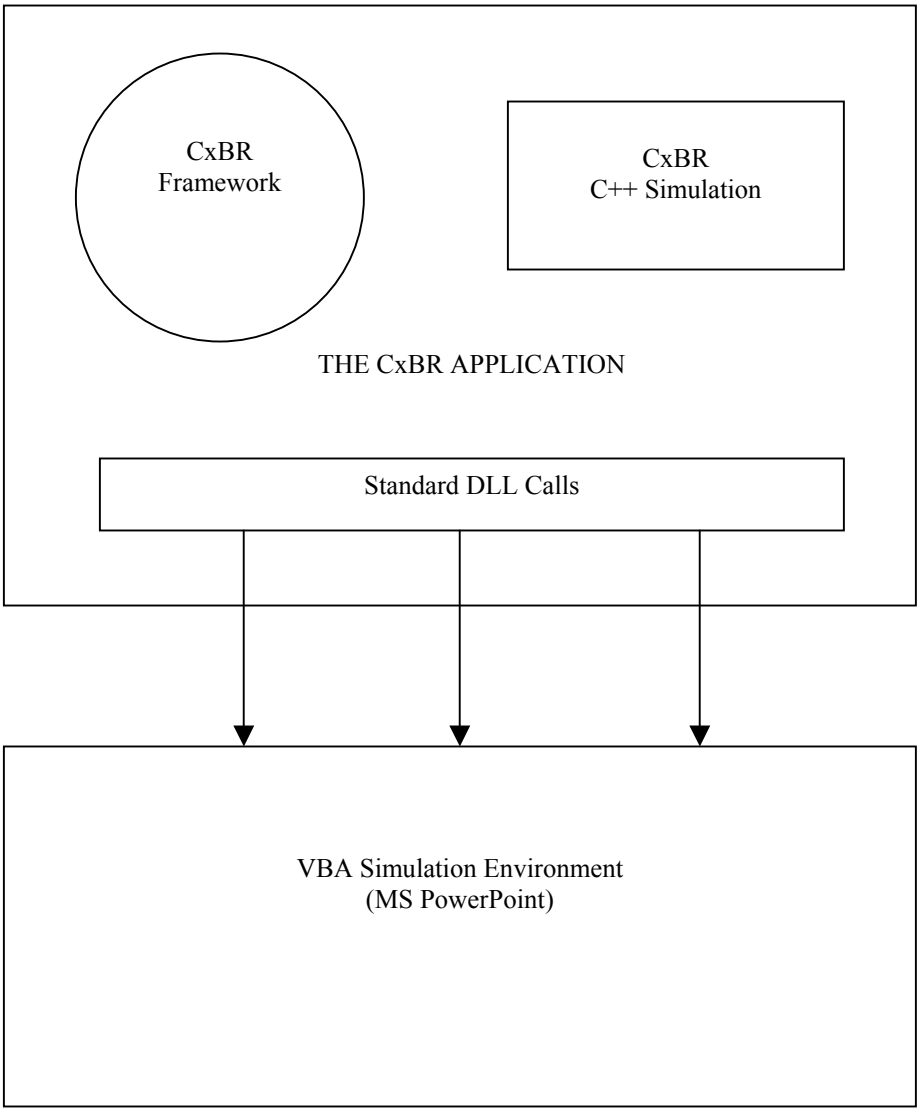

FIGURE 3 – THE CXBR APPLICATION STRUCTURE

A figure showing the new structure given to CxBR applications is shown below in Figure 3. As shown, the CxBR Framework is the core of the application; the CxBR C++ simulation should contain the CxBR Framework by statically linking it in compile time. The simulation environment shows the current state of the simulation by linking dynamically with the C++ simulation.

IMPLEMENTING DEGRADED STATE BEHAVIOR IN CxBR

In this section, we explain in detail the concepts used to implement degraded state behavior. The section is divided into three sub-sections. The first one explains how degraded state knowledge can be incorporated in the AIP class. The second discusses how the contexts can be modified to give them a weight according to their appropriateness given a situation.

Incorporating Degraded State Knowledge in the AIP's Knowledge Base

The first step to incorporate degraded state behavior to the CxBR Framework is to give the AIPs knowledge of their degraded state. The degraded state representation should be a general-purpose approach that could model different kinds of degradation types. A single method was desired that could model such diverse degradation factors as battle damage-induced degradation, psychological degradation, and normal wear-and-tear degradation. Since the method devised had to be reusable by future researchers, the functionality had to be included in the Framework itself. The AIP class was enhanced with the addition of *degradable attributes*. These degradable attributes are initialized at the instantiation of the object, and consist of three required quantities, and an optional number of additional quantities. The required quantities are:

1. Current Value – This value provides the AIP with knowledge of what is the current value of the degradable attribute. This value should not be confused with the actual current value of the associated attribute. For example, the current value of the degradable attribute "maximum speed" can be 75 mph. But the current value of the speed however can be anything up to 75 mph.
2. Maximum Value – This value provides the AIP with the maximum range of the degradable attribute.
3. Minimum Default Value – This value indicates what is the minimum value that the degradable attribute can assume before the AIP loses the functionality associated with the quantity. This quantity is the default value, and it is applicable for all the main contexts of the AIP unless otherwise specified, as described below.

A further capability was included to give each of the AIPs Main Contexts different minimum values for the degradable attributes (than the default). As an example, you would expect the minimum value for the maximum speed of a tank could be different for the **Attack-Enemy** context and the **Tactical-Retreat** context. In the former, the AIP would need speed to carry out the attack successfully. In the latter, it just needs to get away. The developer can then specify quantities specific for each Main Context instead of using the default. In another scenario, one Main Contexts may have different requirements for one of the attributes than do the rest of the contexts. In such cases, the developer can assign a default value that is used for all contexts, and give the one context with different requirements its own value. This value then overrides the default, while the rest use the default.

To make this mechanism available to future CxBR applications, the functionality was included in the AIP class. The methods store the values for the degradable qualities as facts in the local factbase of the AIP. The local factbase of the AIP is accessible only to the AIP. It consists of a set of facts that pertain only to the AIP, and together with the global factbase, it encompasses the state-of-the-world knowledge of the AIP.

Although the format of the facts are hidden from the developer by the methods, the degradable qualities can be created before the application starts by creating factbase files that the application reads in, and initializing the local and global factbases. This is the preferred method for creating the degradable qualities, since it allows the modification of the initial quantities without requiring program recompilation

Adding Weights to the Main Contexts

The CxBR Framework was modified by adding weights to the Context class. These weights are changed throughout the simulation duration according to the situation. The mechanism for changing the weights however was not implemented directly into the Context class, however, as each simulation will produce different situations that require the weights to be adjusted in different ways. Furthermore, a new method for choosing the optimal

context to control the AIP is currently being developed in a parallel project, and it is expected that the functionality developed there will be used to assign the weights to the contexts.

The context class was modified by adding a method for retrieving the current weight and another method for updating the weight that the CxBR developer must implement in a simulation. Further, a transition weight was created that can be used by developers as part of the weight calculation mechanism. This transition weight is set to one if the context is a transition of the current context, and to zero if it isn't. The context is a transition to another if it is a possible next context. This possibility is defined as membership in a list of contexts in the active context that identifies the contexts that can follow the active context. For example, a tank in the **Tactical-Retreat** context can transition to the **Surrender** context, but not to the **Attack-Enemy** context. The **Surrender** context then has a transition weight of one, while the **Attack-Enemy** context has a weight of zero. The methods for changing the weights in the current context are found in the Main Context classes. They will be explained here as an example of how a weight changing mechanism can be designed.

The current project involved the development of a tank warfare simulation. As such, it was decided that the weights of the contexts should depend on three variables:

1. The Transition Weight – This weight determines whether the context is a plausible transition ton the current context.
2. The Threat Weight – This weight varies depending on the level of threat in which the AIP is. Each context will have a different weight conditioned on certain variables on which they depend. For example, the **Surrender** context only has a positive threat weight when the AIP is under attack. The A**ttack-Enemy** context, on the other hand, has a positive threat weight when there is an enemy in sight.
3. The Degradation Weight – This weight varies in accordance to the degraded state of the platform. In the tank warfare simulation developed, the tanks contain a degradation model that incorporates degradation factors such as battle damage, psychological state, and wear-and-tear of the platform. An appropriate weight for each context is set depending on the state of the entity.

These three weights are combined by taking their minimum as the current weight of the context. This weight is stored to allow for later accessing by the context switching mechanism. All the weights used in this simulation were normalized to fall in the range of zero to one. This, however, does not need to be the case, and the developer is free to choose ranges appropriate to the simulation at hand.

The Degradation Weight

Out of the three weights used to calculate the context weight, the degradation weight calculation, is the most complex and important. Further, this weight is directly correlated with the AIP's degraded state, and it is use to give a quantitative measure of the utility of the platform to complete its different tasks. Therefore, the calculation of the degradation weight is one of the main thrusts of this project, and thus warrants further explanation.

The degradation weight depends on three factors:

1. Damage induced by enemy fire.
2. Sub-optimal performance caused by crew psychological factors.
3. Wear-and-tear degradation factors.

Other factors, such as leadership and level of training, although potentially significant, will not be modeled here. They will be left for future research.

We will refer to the example used in the prototype in order to explain this concept. Two types of AIPs were designed for this prototype example: a Tank AIP and a Cannon AIP. Since the project was geared towards tank performance degradation, a degradation model was implemented only for the Tank. The degradation model took into account four major system components, and considered two degradable attributes for each of these systems. Table 1 shows the four system components under consideration with their associated degradable attributes, and an indication of under what type of degradation factor the degradable attributes fall.

The different types of degradation types are obtained in different manners. The enemy fire related degradable attributes are obtained by following the process designated by AMSAA's V/L Taxonomy Levels 2 through 4. The degradable attributes are the platform's subsystems affected by a hit, and they directly correlate to Level 2 of the taxonomy. These systems are affected in different levels of severity depending on where the hit took place. After the subsystem damage is calculated, the Level 2 to 3 mapping can be performed. This mapping consists of determining how the system components (mobility,

firepower, etc.) are affected by the damage to the subsystems. It is noteworthy to point out here that these mapping from degradable qualities damage to system component damage includes the non-enemy fire related degradable qualities, in this case, fuel, and crew energy. Having the system component's capability (Level 3 of the Taxonomy), the utility of the platform to perform different tasks is calculated. This corresponds directly to the Taxonomy Level 3 to 4 mapping. The tasks that the platform can perform correspond to the behaviors that the tank can implement (its Main Contexts). Then the mapping involves calculating the utility of the tank for **Attack-Enemy, Tactical-Retreat, Search-For-Enemy**, and **Surrender**. The utility assigned to the Main Contexts is then taken to be the degradation weight of the context.

Table 1 – List of System Components Considered Along with their Associated Degradable Qualities

System Component	Associated Degradable Qualities
Mobility	Max Speed (Enemy Fire)
	Fuel (Wear-and-Tear)
Firepower	Gun Range (Enemy Fire)
	Ammunition (Wear-and-Tear)
Crew	Number (Enemy Fire)
	Energy (Psychological)
Communications	Radio State (Enemy Fire)
	InterComm State (Enemy Fire)

The various degradation types are obtained in different manners. The degradable attributes related to enemy fire are obtained by following the process designated by AMSAA's V/L Taxonomy Levels 2 through 4. The degradable attributes are the platform's subsystems affected by a hit, and they directly correlate to Level 2 of the taxonomy. These systems are affected in different levels depending on where the hit took place. After the subsystem damage is calculated, the Level 2 to 3 mapping can be performed. This mapping consists of determining how the system components (mobility, firepower, etc.) are affected by the damage to the subsystems. It is noteworthy to point out here that these mapping from degradable attributes damage to system component damage includes the non-enemy fire related degradable qualities (fuel, and crew energy). Having the system components capability (Level 3 of the Taxonomy), the utility of the platform to perform different tasks is calculated. This corresponds directly to the Taxonomy Level 3 to 4 mapping. The tasks that the platform can perform correspond to the behaviors that the tank can implement or its main contexts. Then the mapping involves calculating the utility of the tank for **Attack-Enemy, Tactical-Retreat, Search-For-Enemy**, and **Surrender**. The utility assigned to these Main Contexts is then taken to be the degradation weight of the context.

Note that non-enemy fire related degradation was included to show that they can be easily incorporated into the decision-making process. No attempt was made to ensure the realism of the models representing these factors. For example, fuel was included in the calculation as a wear-and tear factor. It is obvious that there are some degradation factors that are associated with the normal use of the platform, and fuel was included as a representative degradable quality, since it is mainly a function of the distance traveled. The crew energy was included as a psychological factor that affects behavior. In this application, the crew energy was merely set as a decreasing function of time. No claim as to the realism of this approach is made. I/O psychologists have developed already sophisticated methods to model crew degradation, and these models should be included in a more advanced application than the one developed in this project.

For more details on the process of selecting the next current Main Context, please refer to Gallagher [2000].

RESULTS

A prototype was developed which set up the following scenario: A Bluefor tank section, with a mission of **Movement-to-Contact** is moving through the battlefield. It suddenly detects the presence of a fixed artillery piece (called the Cannon AIP), that has a longer range than the tanks, but is immobile. They maneuver to destroy the cannon by moving towards it as fast as possible in a zig-zag pattern until they are within range. Upon reaching their range, they begin firing at the cannon. The cannon, having a longer range, is firing at them upon detection. It only operates in the **Fire-at-enemy** main Context, which simply includes firing at the attacking tanks.

The strike point of each round, from the Bluefor tanks as well as from the Oppfor cannon is based on a random process. When struck, the Bluefor tanks will use a simplified V/L taxonomy to determine the effect of the hit, and the utility of their vital attributes. The commander then must decide whether to continue with the attack, or to shift contexts and perform alternate actions, such as surrender, or retreat. As the point of round impact is random, the results of the various simulations may vary depending on the seed used.

A second scenario was added that includes 2 Oppfor tanks appearing in the distance. Now the Bluefor tanks must decide on whom to fire first. The Bluefor tank's range is longer than that of the Oppfor tanks.

The results indicated that the Bluefor tanks correctly performed the decision-making based on their degraded states. In some of the runs, one of the Bluefor tanks was killed and required surrender of the crew. In most cases, they succeeded in destroying the two opposing force tanks as well as the cannon. In some cases, both tanks were damaged and had to implement a retreat, popping smoke and firing as they retreated. The detailed results can be found in Gallagher [2000].

CONCLUSION

The current project has shown the utility of CxBR in representing degraded state reasoning. The CxBR paradigm has been accordingly extended, and the new capabilities have been demonstrated by the development of a tank simulation under a VBA environment with a PowerPoint engine. The degraded state behavior has been incorporated into CxBR by following two steps: First, degraded state knowledge was incorporated into the autonomous entities. Second, this knowledge was used to change the weights of the transition contexts, which will affect the decision of which context to choose. The battle related degradation factors were determined by using an approximation of the approach proposed by AMSAA's new vulnerability/lethality taxonomy.

By incorporating degraded state knowledge into the CxBR paradigm, it now has the required capabilities to develop realistic intelligent agents that act differently according not only to the situation they are in, but also according to their current degraded state.

Future research includes integrating better models for the degradation of the factors related to enemy fire (the actual V/L Taxonomy). Additionally, better models of the factors that affect the crew, such as fatigue, morale, crew health, availability of leadership, training, and other such physical and psychological factors need to be integrated. Lastly, models dealing with wear-and-tear degradation need to be developed. These should be significantly simpler than the previous two models, however.

REFERENCES

[Dietz and Starks, 1997] Dietz, P. H. and Starks, M. W., "The Generation, Use and Misuse of "PK's" in Vulnerability/Lethality Analyses", Proceedings of the 8th Annual TARDEC Symposium, Naval Postgraduate School, Monterey, CA, March 1997.

[Gallagher, 2000] Gallagher, A., "Modeling Platform Behaviors under Degraded States Using Context-Based Reasoning", Master's Thesis, Electrical Engineering Program, School of Electrical Engineering and Computer Science, University of Central Florida, Orlando, FL December, 2000.

[Gonzalez and Ahlers, 1998] Gonzalez, A. J. and Ahlers, R. H., "Context-based Representation of Intelligent Behavior in Training Simulations", Transactions of the Society for Computer Simulation, Vol. 15, No. 4, December, 1998.

[Gonzalez et al., 2000] Gonzalez A., Georgiopoulos M., DeMara R. "Context-Based Representation of Intelligent Behavior in Degraded States Simulation. " Naval Air Warfare Center Proposal (Unpublished). University of Central Florida, Orlando, FL, 2000.

[Norlander, 1999] Norlander, L., "A Framework for Efficient Implementation of Context-based Reasoning in Intelligent Simulations", Master's Thesis, Department of Electrical and Computer Engineering, University of Central Florida, Orlando, FL, January 1999.