Stottler Henke
Smarter Software Solutions

An Experimental Testbed for Tactical Command and Control

Alex Davis, Dan Fu
Stottler Henke Associates, Inc.

John Rushing
Univ. of Alabama Huntsville

Alexander Sarnacki
Air Force Research Laboratory

Stottler Henke Technical Report TR2007-02

June 30, 2007

Abstract

Recent concepts in the field of command and control (C2), such as Power to the Edge and Network-Centric Warfare, have indicated the need for a testbed for experimentation. We describe a gaming testbed, populated by realistic synthetic agents, for modeling the complex human interactions comprising C2 structures, and for exploring the effectiveness of C2 concepts in a variety of tactical circumstances.

A testbed for experimentation with C2 concepts must be capable of reproducing the complex, rapid, dynamic, and unpredictable unfolding of events in battle. Scenarios can be carefully limited, and realism diluted to a practical level, but enough of the complex interactions of individual and collective behavior must be preserved to capture the essentially human nature of C2. Experimentation involves the correlation of tactical events and outcomes with a host of variable parameters, such as C2 structure, communication patterns and reliability, and implementation of commander's intent.

This report describes the details of our testbed and experiment design and implementation, in addition to findings and lessons learned. Our approach included the integration and extension of three existing platforms: Counter-Strike, a multiplayer first-person tactical shooter; SimBionic, a visual behavior authoring and execution engine developed at Stottler Henke; and ADaM, a data mining tool suite developed at the University of Alabama, Huntsville. The testbed produced by this integration allows for the planning and execution of missions by human or synthetic agents, including specifications of different C2 structures, and the mining of experimental data for the effectiveness of the C2 concepts. We discuss the applicability of our findings to other echelons, as well as its contribution to the evaluation of new C2 concepts.

Keywords

Adaptive adversaries, Genetic algorithms

1 INTRODUCTION

Recent and emerging theories in the field of command and control (C2), such as *Power to the Edge* and *Network-Centric Warfare*, have afforded compelling ideas of how teams might be organized to maximize their effectiveness. For empirical purposes, there exists a need for testbeds which can serve as experimental environments for new ideas. A testbed for experimentation with C2 concepts must be capable of reproducing the complex, rapid, dynamic, and unpredictable unfolding of events in battle. Scenarios can be carefully limited, and realism diluted to a practical level, but enough of the complex interactions of individual and collective behavior must be preserved to capture the essentially human nature of C2. Experimentation involves the correlation of tactical events and outcomes with a host of variable parameters, such as C2 structure, communication patterns, and reliability.

At the same time as C2 theories gain currency, there has been a timely interest in leveraging game-based technology platforms, specifically for training and simulation purposes. These platforms feature already-completed engines, tools, and content which present low cost, low risk ways to build useful serious games. In this paper we describe our efforts to produce a game-based experimental testbed meant to enable researchers to:

- **Configure C2 structures**: Specify how command and communication happens between individuals.
- **Adjust performance parameters**: Change performance criteria of communication and command.
- **Visualize logged data**: Play back logged data.
- **Evolve behavior**: Genetic Algorithms (GA) vary structures and performance parameters to derive new behavior.
- **Perform data mining**: Run standard data mining algorithms on logged data to find patterns of interest.

Our approach includes the integration and extension of three existing platforms: a multiplayer game, AI authoring tool, and data mining tool suite. The resulting testbed produced by this integration allows for the execution of missions by human or synthetic agents, including specifications of different C2 structures, automated evolution and evaluation of those structures, and the mining of experimental data for the effectiveness of C2 concepts.

In this paper we describe the details of our testbed, experiment design and implementation, in addition to findings and lessons learned.

2 Testbed Description

Our testbed comprises three main components:

- **Counter-Strike game:** a 3-D tactical shooting game pitting terrorist versus counter-terrorist;
- **SimBionic AI authoring tool:** a visual behavior authoring and execution engine developed at Stottler Henke (Fu and Houlette, 2002); and
- **ADaM data mining tool suite:** machine learning software developed at the University of Alabama in Huntsville (Rushing et al, 2005).

A **C2 configuration** and **log playback tool** complete our testbed. Figure 1 shows the testbed's organization. It operates in one of two modes. In the first, the "configuration tools" are for a researcher who specifies a C2 configuration and performance parameters for an experiment. The Counter-Strike

game server starts and begins a round. The SimBionic AI tool uses the data to instantiate non-player characters (NPC's) that will conform to the experiment's specifications. Human participants may take part in these experiments. After the round is completed, the user may view the logged data through a playback tool, or invoke the data mining toolkit to search for interesting data.

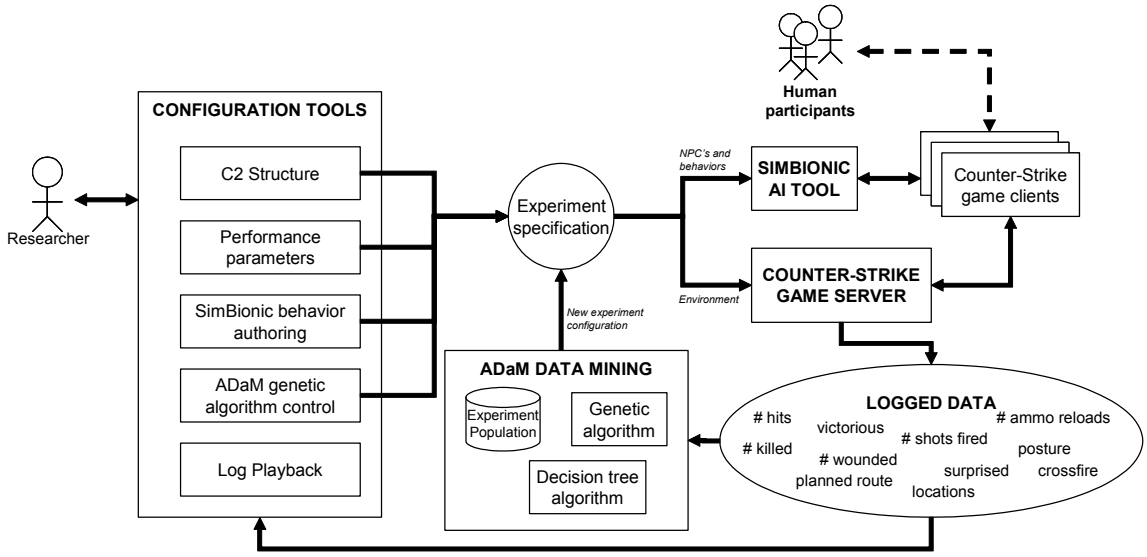

Figure 1. Testbed overview.

In the second mode, the testbed will automatically experiment with various configurations to discover effective C2 structures. Here, the testbed uses a genetic algorithm to explore the space of possible structures, discovering, combining, and testing successful C2 structures and evaluating their efficacy. In the rest of this section, we describe the components in more detail before proceeding to analysis in the next section.

2.1 The Counter-Strike Game

Counter-Strike is a 3-D multiplayer first person shooter game based on the Half-Life game. While this game hosts up to thirty-two human players, there exist alternate ways to insert NPC's into the game. Their participation in the game is similar to a human player. An important distinction of Counter-Strike is its encouragement of team-based gameplay. Individual players are rewarded only if their respective team wins. This implies that successful members coordinate and support each other as opposed to "death matches" where everyone competes against each other.

Our approach using this game is from the bottom up, with a low level, tactical game that can provide interesting results in the evaluation of C2 concepts, and form a basis for C2 analysis techniques that potentially scale to higher levels of command. Squad-based use of CS gives three types of actors in the chain of command:

1. **Soldiers in the field**, having direct interaction with the mission environment producing immediate effects, embodying a fast-paced, stressful environment for mission execution and leadership in the field.
2. **Intermediate C2 staff** that must coordinate different units and levels of command. They may be present in the field, in a staging area, or offsite.
3. **Command leadership** that must enact intent, with varying levels of intervention into mission execution.

Scenarios in Counter-Strike involve up to a platoon-sized contingent in tactical urban operations, such as clearing buildings or rescuing hostages held by terrorists. Communications are all-channel radio broadcast

of simple commands and information, and control is left up to the players in the field. We augmented this system with explicit C2 mechanisms, allowing players on and off site to monitor and adjust the close battle, as well as leadership and functional roles that transform the free-for-all environment of commercial Counter-Strike into optional elements such as fire teams and squads fulfilling assault and support roles.

This allows for some low-level versions of C2 elements, such as unit coordination and logistics, while preserving fidelity and individuality of behavior at both command and execution levels.

2.1.1 Counter-Strike AI

NPC's are capable of a variety of basic behaviors, such as following specified paths or other agents, searching, and engaging observed enemies (Fu, Houlette, Jensen, & Bascara, 2003). They also are capable of more complex behaviors such as taking cover on contact, calling for backup, responding to calls, and coordinating with another team of agents to prepare and time a synchronized attack. These behaviors are configured through the testbed into C2 schemes; for instance, a command hierarchy is formed by establishing each agent's policies for whom to follow (or lead), with whom to communicate, and how to react to various events.

2.1.2 Counter-Strike Scenario

The baseline scenario within which these agents operate includes two opposing teams, CT (counter-terrorist) and T (terrorist). They start at opposite ends of the map and attempt to destroy the opposing team, at which point the game ends. Our analyses tended to stage the T in a defensive position that the CT would have to assault, but the testbed supports a variety of other scenarios, such as the T sending some members to sniper locations, or both sides actively searching for the enemy.

2.2 C2 Configuration

The configuration editor is the starting point for the end-to-end analysis cycle. This editor allows the experimenter to set up each team with an arbitrary command hierarchy expressed as trees, and with arbitrary communications scheme expressed as nets of common communication between agents. It also allows the configuration of a variety of variables that affect gameplay and decision making.

Two of the variables correspond to command: **promotion delay**, which is the time that must pass before a fallen leader is replaced, and **backup contingent**, which is the number of units sent in response to a backup call. These parameters are intended to implement a portion of C2 policy, as an abstraction of the effects of decisions made by an on-scene or off-scene commander, or the by the teams themselves.

Two other variables affect communications: **drop probability** and **message delay**. These variables simulate variance in reliability of communications, and in their time of travel along different avenues (for instance, via a third off-scene party).

Other variables are also configurable, chiefly as parameters to agent decision-making behavior, such as whether a team defends, searches for targets, or attacks along planned routes (which are also configurable), and whether agents attempt to cover and synchronize engagements upon contact with the enemy. These variables are provided in order to implement particular policies, plans, and scenarios within the baseline scenario, and to guide the action into particular points of interest, such as the moment of synchronized assault on a defensive position (Stottler, Lackey, & Kirby, 2004).

2.3 ADaM Data Mining

ADaM (Algorithm Development and Mining) is a freely available data mining toolkit designed for use with scientific data. The ADaM toolkit provides a suite of tools for each of the basic data mining processes, including classification, clustering, association rule mining, and preprocessing. The toolkit is packaged as a series of independent components. Each component can be used either as a standalone

executable, or from within the Python scripting language via a wrapper. As mentioned earlier, ADaM's role within the testbed is to (1) data mine for patterns, and (2) evolve more effective C2 structures.

2.3.1 Data Mining Analysis for C2

We use decision trees to determine which C2 parameters contribute most to victory. To accomplish this, the logged data serves as input to ADaM, which then produces a decision tree that can be used to predict outcomes. Decision trees select attributes based on relevance to the category of interest (in this case, victory conditions) using information gain.

2.3.2 Genetic Algorithms for Evolving C2

Genetic algorithms (GA) are general purpose search algorithms that are used to solve difficult optimization problems. Genetic algorithms work by generating potential solutions to the problem of interest in some problem specific search space. Typically, solutions are described by bit strings. The genetic algorithm randomly generates a population of bit strings and evaluates them using an objective function. It then generates a new population of bit strings by recombining the ones from its existing population, with the higher rated strings contributing more to the new population.

Genetic algorithms are used to optimize C2 parameters with respect to a particular scenario. In order to accomplish this, it is necessary to represent the relevant parameters as a bit string for the GA. This is generally done by quantizing each parameter within a fixed range. Assigning more bits to a parameter expands the search space and allows for finer tuning. Given a coding scheme, the bit string is evaluated by running the game engine many times with the corresponding AI settings and deriving statistics from the results.

2.4 Visualization

The testbed is designed to lend insight into the operation of lifelike missions, and so visualization of individual experimental runs is important for understanding the reasons and implications of any given game outcome. Two tools were supplied: the game itself, where an experimenter could fly around in the game space and observe activities, or even participate in them, and a tracking tool which renders the game in a single 2-D view.

For in-game observation, agent and terrain markup is supplied to annotate behaviors. The map shows waypoint locations of various types (such as transit and sniping points), and the agents show their navigating, following, engaging, and communications activities through lines connecting them to waypoints and other agents. This view can only be used while the game is in progress.

The agent tracker, shown in Figure 2, shows similar information, but reads a log file and can be rewound and replayed. It shows a top-down view of all agents on the map (red or blue dots) and their direction of sight (line from dot), and marks events such as deaths ("X"), and engagements connecting shooters to targets (arrow to target).

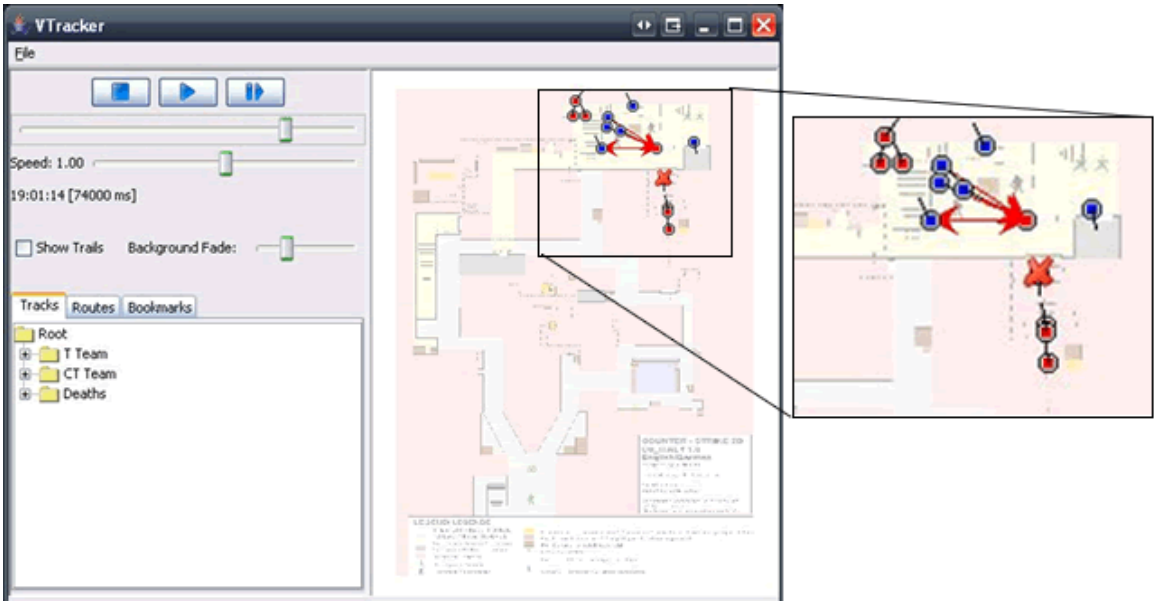

Figure 2. Log playback tool.

3 Analysis

The testbed can be configured to run a predetermined number of times, for a given fixed configuration, or by automatically incrementing variables over specified ranges. Other routines are provided for performing analyses on the logs of these runs, which record moment-to-moment events and agent activities, such as moving, following, pointing, targeting, and engaging. At the end of the process, the testbed produced human-readable results from analysis products. We performed two analyses in order to exemplify findings derivable from the C2 testbed.

The first phase of analysis pitted counter-terrorist (CT) against terrorist (T), each using one of three organizational schemes: hierarchy, hive, and independent. The hierarchy consists of a two-tiered tree, with two teams, and one additional leader whose team was made up of those teams' leaders. The hive is a single-tier hierarchy, with one leader leading all other players on a side. The independent scheme included no teams, each players deciding and acting individually. Along with the nine combinations of those organization schemes, T's were configured to either defend or go on offense, searching the map for targets. The number of units responding to a backup call was also varied.

The full set of possible configurations was run in the testbed, and ranked according to kill ratio, the ratio of CT dead to T. This ranking confirmed expectations regarding game mechanics, such as the fact that shotguns were better one-on-one and at short range, while rifles were superior at long range and in groups; and regarding light infantry engagements in general, such as the superiority of defense to offense. C2 configurations tended to succeed when they favored the appropriate tactic for weapon choice, such as hierarchical search teams, which tended to stay in groups, bearing rifles.

The second phase of analysis focused on decision making by the CT team. It varied weapon choice, planned assault vs. target search, whether to engage targets of opportunity, whether to synchronize assault, and the sizes of the two teams. The hierarchical organization was fixed, as was the T team's defensive posture.

This set of variables was encoded in a bit string and used for the objective function of a genetic algorithm optimizer, using kill ratio as a measure of fitness. We ran two generations of the algorithm, generating ten results for each of 78 bit strings. The results showed the superiority of the rifle to the shotgun (here,

engagements tended to initiate at longer range), of large groups to small, and of a search approach to the planned assault. The latter result was the most prominent.

Table 1. First phase experimental results.

CT Structure	T Structure	T Stay	CT Wins	T Wins	Draws
Independent	Independent	No	46	62	0
Independent	Hive	No	43	65	0
Independent	Hierarchical	No	43	65	0
Hive	Independent	No	52	56	0
Hive	Hive	No	42	66	0
Hive	Hierarchical	No	44	64	0
Hierarchical	Independent	No	47	61	0
Hierarchical	Hive	No	42	66	0
Hierarchical	Hierarchical	No	49	59	0
Independent	Independent	Yes	13	90	5
Independent	Hive	Yes	37	71	0
Independent	Hierarchical	Yes	41	66	1
Hive	Independent	Yes	11	95	2
Hive	Hive	Yes	21	87	0
Hive	Hierarchical	Yes	28	78	2
Hierarchical	Independent	Yes	9	95	4
Hierarchical	Hive	Yes	26	80	2
Hierarchical	Hierarchical	Yes	28	78	2

4 Lessons

The preliminary analyses described above demonstrate the effectiveness of the testbed as an analysis tool, but did not present novel results. The primary reason is that kill ratio is not an ideal measure of fitness for a C2 configuration in this game. Because the game mechanics are not deterministic, and the agent behaviors are not robust enough to handle every kind of contingency, final outcome appears often to derive from other factors than those being studied. There are, however, other products of analysis that can serve better. For synchronization, the testbed could be used to measure the success of the tactic by examining the achieved surprise, as a tally of enemies engaged without seeing their attackers, and crossfire, as a tally of enemies engaged from multiple directions. Synchronized attack in this scenario is designed to maximize these measures. While a great number of experimental runs might be necessary to observe the effects of these achievements on final outcome, the experimenter could optimize surprise and crossfire directly. The agent tracker and in-game observation can be used to observe the reasons for the outcome, and suggest other indicators to be studied.

The influence of game mechanics was in general problematic, in the failure of many experiments to isolate the influence of the controlled variables. The variety of weapons available tended to interact strongly with certain tactics and situations, overwhelming the influence of many leader decisions. When weapon choice was placed among optimization parameters, it tended to predominate as a factor in the final outcome, particularly when combinations of tactics led to consistent encounters at long or short range, and between large or small groups. The fine-grained game world also posed a variety of difficulties for agent behaviors, when tended to affect their ability to travel in close groups, the timing of their actions (as opposed to their decisions), provide backup without encountering other trouble, and in general their ability to put weapons on target.

More sophisticated behaviors could account for each of these obstacles, but it would require substantial work on aspects of behavior that do not speak directly to C2 considerations.

Perhaps as a consequence of using a game platform meant for human players, the time to conduct experiments was nontrivial, taking approximately four days to run only two generations for the second analysis phase. Each objective function evaluation took about 1.23 hours on average. The possibility of running the game faster than real time was not explored.

5 Conclusion

The C2 testbed in its current form can be used for extensive analysis to greater extend and with different emphases than the preliminary analyses we performed. Further development could include the implementation of an off-scene commander agent, who could be configured to communicate with various participant agents and issue orders to them. This would establish the influence of a common operating picture, or complete knowledge about the ongoing state of the game, which the commander would attempt to synthesize as a basis for decisions. This would allow for analyses of the ideal communications content, level of control by the commander, and distribution of decision-making responsibility over all agents, in order to determine optimal empowerment of the 'edge', or the players on the ground, while still making maximal use of available information.

This testbed approach could also be applied to other domains, such as strategy games. This would vastly reduce the state space of the game, and so reduce the amount of data available for the mining of interesting results, but would also allow for more complex behaviors to be implemented and more extensive analyses to be performed.

6 ACKNOWLEDGEMENTS

This work was funded and sponsored by Alexander Sarnacki of the Air Force Research Laboratory. Opinions expressed are those of the authors and do not necessarily represent an official position of the Department of the Air Force or the Air Force Research Laboratory.

7 REFERENCES

Fu, D., Houlette, R., Jensen, R., and Bascara, O. (2003). "A Visual, Object-Oriented Approach to Simulation Behavior Authoring," in Proceedings of the Industry/Interservice, Training, Simulation & Education Conference (I/ITSEC 2003).

Fu, D., and Houlette, R. (2002). Putting AI in Entertainment: An AI Authoring Tool for Simulation and Games. IEEE Intelligent Systems, August 2002.

Rushing, J., Ramachandran, R., Nair, U.J., Graves, S.J., Welch, R., Lin, H. (2005). ADaM: A Data Mining Toolkit for Scientists and Engineers. Computers and Geosciences, Volume 31, issue 5, pages 607-618 (June 2005)

Stottler, R., S. Lackey, J. B. Kirby. (2004). "Formalized Behavior Models for MOUT OPFOR Individual Combatant Weapon Firing," Proceedings of the Industry/Interservice, Training, Simulation & Education Conference (I/ITSEC 2004).