Reprint from *Maneuver Warfare Science 2001*. G. Horne, M. Leonardi, Eds. (USMC, Quantico, VA, 2001)

The Archimedes Combat Modeling Platform

William N. Reynolds and David S. Dixon
Least Squares Software LLC
PO Box 91405
Albuquerque, NM 87199

grendel@leastsquares.com
ddixon@leastsquares.com

1. Introduction

The Archimedes Combat Modeling Platform is a simulator system based on ideas developed from research into the behavior of complex systems.[1] The goals of Archimedes, and more broadly of its sponsor, The United States Marine Corps Combat Development Commands' Project Albert[2], is that it be able to represent the *intangibles* of combat and that it capture the *nonlinearity* intrinsic to battlefield situations. Intangibles include the roles of discipline, cohesion, morale and personality. Nonlinearity, also called the *butterfly effect*[3], applies to systems in which perturbations as seemingly insignificant as the flapping of a butterfly's wings can, over time, grow to the point where they can actually change the outcome of a battle.

Additional design criteria for Archimedes were that it be flexible enough to represent the new challenges currently faced by the United States Marine Corps, including NEO[4], SSC[5], and MOUT[6]. The system is also designed to be extensible enough so that it could be easily modified to address new problem domains and unforeseen contingencies.

2. Bringing the New Sciences to Combat Modeling and Simulation

Archimedes is based on the paradigm of *Agent based modeling (*ABM*)*[7]. This approach breaks a problem down into constituent elements, called *Agents*. Archimedes is based on a general high-level ABM software framework called the *Behavior Action Simulation Platform* (BASP)[1]. *Agents* are autonomous software units that work independently towards some goal (which may involve cooperating with other Agents).

Guided by the ideas of operational synthesis[8], Archimedes was designed first to be as flexible as possible in order to represent a broad variety of missions. The analyst defines all of the Agents within a simulation, including the meaning of an Agent for a given simulation, *e.g.* whether it represents an individual, a infantry, a mission objective or some other simulation entity. The simulation is multi-resolution: Agents may represent individuals or units of any size, including heterogeneous collections. The analyst specifies the physical capabilities and behavioral characteristics of each Agent appropriate for its function.

The system provides a high level language for specifying Agent behaviors. For example, within the same simulation, the analyst could specify the behaviors and capabilities of both an individual commander Agent and the commander's subordinate platoon Agents. The behavioral specification is extremely flexible and modular – an analyst may specify behaviors for Agents, and for the interactions between Agents (*e.g.* "IF *attitude towards non-*

---

[1] Portions of this paper appeared in Reynolds, William N. and Dixon, David S. "A General Framework for Representing Behavior in Agent Based Modeling" in *Complex Systems and Policy Analysis: New Tools for a New Millennium*, Sept 27-28, 2000, RAND Corporation, Science & Technology Policy Institute, Arlington VA.

[2] http://gilgamesh.mhpcc.edu/index.html

[3] Gleick, James, *Chaos*, Penguin Books, New York, NY 1987.

[4] http://www.aic.nrl.navy.mil/~aha/neos

[5] http://www.ndu.edu/inss/sa98/sa98ch10.html

[6] http://call.army.mil/homepage/mout.asp

[7] Weiss, Gerhard (ed.). *MultiAgent Systems*, MIT Press, Boston, MA 2000.

[8] Horne, Gary E., "Maneuver Warfare Distillations: Essence not Verisimilitude", *INFORMS 1999 Winter Simulation Conference,* . 5 – 8 December 1999, Institute for Operations Research and the Management Sciences, University Park, PA.

combatants IS *friendly* THEN *attitude towards militia* IS *neutral*. END IF"). Analysts are free to define virtually any behavior of their own devising. This freedom in specifying behavior allows the analyst to explicitly incorporate intangibles into their modeling approach.

As a concrete example, in a study on reconnaissance[9], the analyst determined that discipline was a factor important to the success of the mission. The analyst then defined behaviors wherein the amount of time spent at reconnaissance rally points and the Agents' adherence to tactical dispersion were dependent upon an analyst-defined Variable called "discipline". Outcomes were then studied as this Variable was varied.

A second design criteria was that Archimedes be able to capture the nonlinearities of combat. An exciting approach to this problem is the technique of *data farming*[8], in which low fidelity, fast running *distillation* models are run multiple (*e.g.* millions of) times to exhaustively explore the outcome space of a particular configuration. Those regions of outcome space of interest – which might include areas where the outcome is the desired one, or particularly, the boundary regions between two different outcomes – can then be explored in detail using a high fidelity model. To accommodate this, Archimedes has been endowed with scalable fidelity, allowing the analyst to run both distillations and high fidelity explorations using the same system.

3. The Archimedes Combat Modeling Platform

Archimedes uses the BASP modeling architecture, which represents both Agents and the interactions between them as software objects. A BASP interaction is a software object called a *Nexus*, the most common of which is the two-Agent Nexus, or *Connection*, and the three-Agent Nexus, or *Triple*. At its highest level, BASP separates simulation dynamics into a behavioral component (wherein Agents choose a behavior) and an action component (wherein Agents manifest a behavior). The analyst designs and builds the interactions between Agents, which includes specifying both physical actions and behaviors that occur on the interaction. For example, a commander Agent will have different behaviors and actions with regards to an objective Agent (*e.g.* move towards*,* defend*)* than with regards to a subordinate (*e.g.* communicate with, move in formation with) than with regards to an enemy Agent (*e.g.* detect, avoid, fire upon).

The analyst performs these specifications by building template objects that represent types of Agents, Connections, Triples and Nexuses. When a simulation is to be run, the template objects are used to generate instances of the Agents and their interactions. Interactions are directional: for example, between two Agent templates, the analyst may specify two Connection templates (in which each Agent template alternatively plays the role of the *owner* and *partner*). Between three Agent templates, the analyst may specify up to six distinct Triple templates. Associated with each of these templates are *Variable Templates*, which represent the behavioral state of an object (in our example above, "discipline" was a Variable). Agents and interactions are also endowed with an additional type of software object: the *Aspect.* Aspects represent a physical capability of an object, for example, movement, direct fire, communication or detection (Aspects are a key concept in the Archimedes system and are discussed in detail below). Like Variables, the analyst attaches Aspect templates to both Agents and interactions. Specific instances of these templates interrelate as depicted in Figure 1.

BASP uses *heavy* Agent and interaction objects; which is to say there is a great deal of breadth and flexibility in specifying the capabilities of an object. Fundamentally, the attributes are divided into two regimes: *actions* and *behaviors*. Actions are the physical capabilities possessed by the Agent or interaction, while behaviors are decisions that are made by the Agent or interaction, which may cause actions to be performed. Every Agent and interaction is endowed with a set of quantities that represent its state. This state consists of an arbitrary number of parameters. The state is divided into two domains:

- *Physical state*: This is the action part of the state. It represents the relationship of the Agent or interaction to the physical world in which the simulation is embedded. Examples of physical state include the position and velocity of an Agent in a physical environment, the distance along a Connection separating two Agents in that

[9] Study performed by Capt. M. Leonardi at the Project Albert International Workshop, Maui High Performance Computer Center, June 2000.

environment, or the capital held by an Agent in a trading simulation. All physical state information is encapsulated in the appropriate Aspect object.

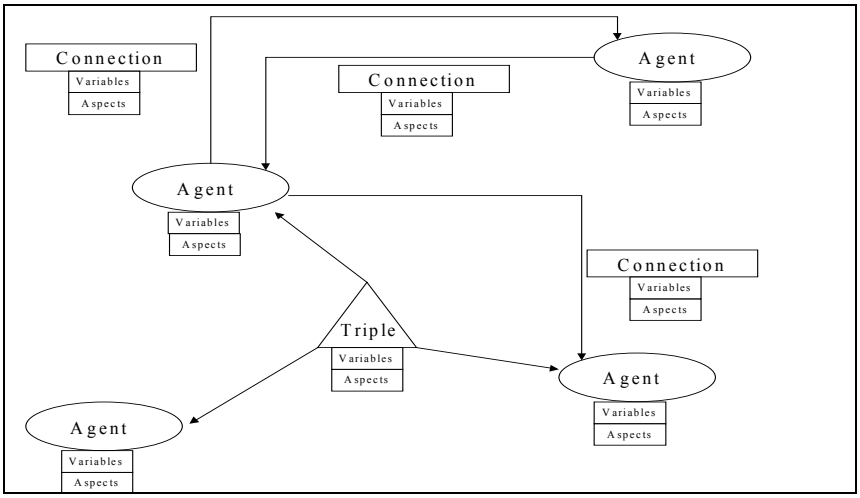

Figure 1: Schematic of the Simulation Architecture

- *Behavioral state*: This represents the intent of the Agent or interaction, which may or may not be associated with physical reality. For example, the "Movement State" of an Agent embedded in a physical environment is the Agent's desired speed. The physical speed will reflect the realities of the local terrain, the Agent's movement capabilities and other physical effects. On a Connection, the "Perceived Threat" would represent the degree to which the Connection's owner Agent is threatened by its partner Agent, regardless of the actual threat that the partner poses the owner.

The behavioral state of an Agent or interaction is represented as a collection of Variables. These are instances of analyst-defined Variable templates, one of which may, for example, divide “Distance” into five fuzzy categories, such as “Here”, “Very Near”, “Near”, “Far”, and “Very Far” representing a range of one thousand meters.

The physical state on an Agent or interaction is represented by Aspects attached to that Agent or interaction. There is an Aspect *class* associated with every physical action in the simulation, (*e.g.* movement, combat or communication). Aspects encapsulate any environment-specific and model-specific code and data associated with the performance of actions. Aspects transmit information from the physical state to the behavioral state. For example, an Aspect on a Connection updates the behavioral Variable "Distance", which, based on the analyst’s definition, may translate 750 meters into the category “Far”.

The analyst controls Aspects in two ways: by choosing which Aspect templates to associate with an Agent or interaction template, and by invoking Aspect actions via changes in the behavioral state.

New Aspect algorithms are incorporated merely by making the appropriate code available to the simulator, allowing the analyst to select the new Aspect template. The modularity of Aspects allows an analyst to explore the system’s nonlinearity by running both low fidelity distillations and high-fidelity simulations. For the distillation, the analyst would choose simple, fast running Aspects, and for the full simulation, he chooses slower running, high-fidelity Aspects.

Similarly, Aspects provide the simulation with the extensibility and flexibility necessary to address new, difficult problems. If the platform lacks a feature necessary to perform a given simulation (for example, movement in urban terrain, or air operations), the analyst would direct a programmer to develop a new Aspect with the requisite

capability. The programming interface to Aspects is well defined, and their modular nature provides a firm foundation that allows the analyst and the programmer to easily communicate and quickly implement the new capability.

Each Aspect template provides a graphical editor for specifying its state. For simple Aspects, the programmer developing the Aspect simply specifies a list of parameters and behavioral Variables that the Aspect will access, and the interface is generated automatically. For more sophisticated Aspects, the programmer may implement a custom interface.

Aspects allow the system's physical actions to be controlled by the behavioral Variables, and feed information from the physical to the behavioral domain. With one important exception, Aspects do not impose any dynamics on the behavioral state. The exception is the *Behavioral Aspect* class. This class of Aspects provides dynamics for the behavioral state. Like all Aspects, it is a software object, albeit a rather sophisticated one. The interface to this Aspect template is one of the principle means whereby the analyst may control the simulation. This is done by specifying *behaviors*, which are simply dynamics on the behavioral state. Like other Aspects, there can be multiple types of Behavioral Aspects, using different formulations of behavioral dynamics. The current implementation has one Behavioral Aspect, which uses a natural language-like *fuzzy logic* mechanism for specifying behaviors. As an example, Figure 2 is the editor for the behavioral (or *rule*) Aspect template. The rules depicted are attached to the Connections linking a team of Agents, these rules enable the team to maintain tactical dispersion.

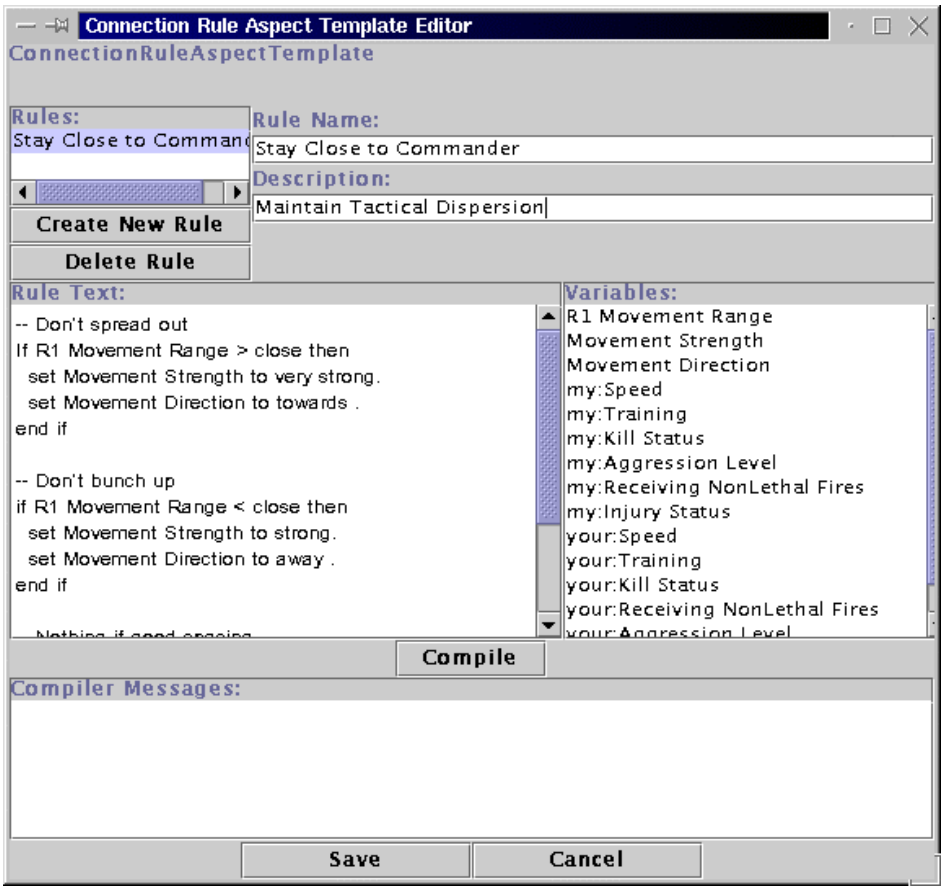

Figure 2: Editor for the Behavioral Aspect

The Behavioral Aspect is designed to be the one of the principal interfaces to the simulation for the end-user analyst. The analyst can quickly specify or modify the behaviors of the entities in the simulation. Using the Aspect and its interface, the analyst can easily understand, specify and modify the system's behaviors, which are couched in familiar technical terminology.

Behaviors are specified by separate Behavioral Aspects for Agents, Connections and other Nexuses. This turns out to be a very powerful formalism – assigning behaviors to Connections between different Agent types allows multi-dimensional (N-sided) scenarios to be readily modeled, a capability that has proven to be quite difficult to implement using other formalisms. Higher order Nexuses enable the specification of even more sophisticated behaviors.

While the fuzzy logic Behavioral Aspect has proven to be very general and very useful by a wide class of analysts, it is not the final word. For problem domains where experts require a different formulation of behavioral dynamics – for example, stochastic dynamics – a new Behavioral Aspect could be implemented on fairly short order. The only requirement is that the dynamics be defined on a set of behavioral Variables. Hybrid simulations, which use different behavioral dynamics on different Agents and interactions, are then easily realized. In this way, intangible factors that are not amenable to study using the fuzzy logic formalism can still be modeled by the system.

4. Example

To illustrate some of these concepts we present the following example. The scenario depicted is a reconnaissance scenario. The (very simple) playbox consists of a 10x10 grid, each element on the grid represents a distinct terrain element. Each terrain element is characterized by three quantities: trafficability, which affects movement; cover, which affects combat adjudication; and concealment, which affects detection. Blue represents water, yellow open terrain, green forest, gray mountain and dark gray mountainous forest. All accesses to this playbox are moderated by a *Position Aspect* attached to each Agent. In the event a different terrain representation was desired, one would simply attach a new Position Aspect that accessed the new terrain representation (naturally, all of the other Aspects that accessed terrain elements, such as movement and detection, would have to be modified to utilize the new features of the terrain).

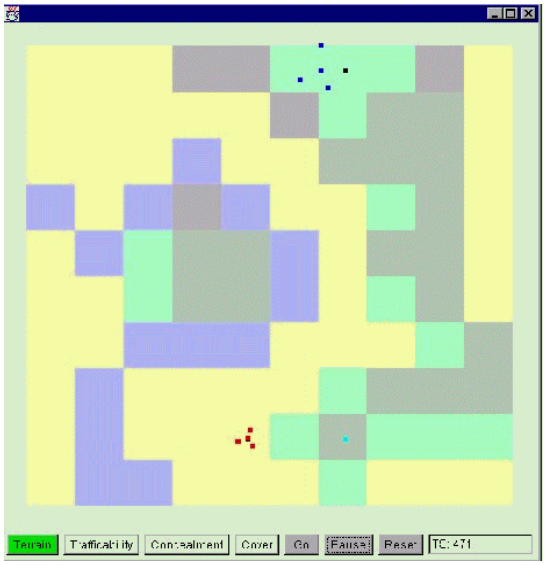

Figure 3: High Discipline Reconnaissance Team

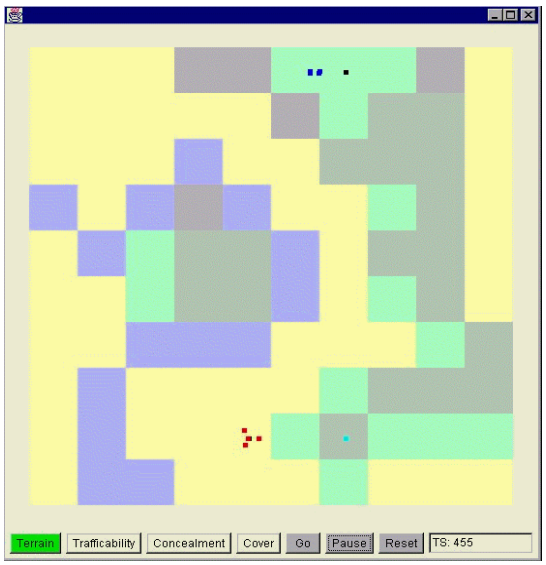

Figure 4: Low Discipline Reconnaissance Team

In this scenario, a recon team (blue Agents) traverses a series of recon checkpoints (black Agents), searching for an objective (turquoise Agent) that is under guard by the enemy (red Agents). The blue Agents' goal is to locate the objective without being detected by the red Agents. The blue Agents are endowed with behaviors designed by the analyst to enable them to achieve their goal. These behaviors are dependent upon an Agent Variable, *discipline.* Discipline affects the simulation in three ways. Agents with high discipline maintain good tactical dispersion and remain observing at reconnaissance checkpoints for long periods of time, as discipline decreases, so does the degree of tactical dispersion and the delays at checkpoints. Connections linking the blue to red Agents and the red to blue Agents are endowed with *Detection Aspects* that determine whether the Connection's owner has detected the Connection's partner. This Aspect uses an algorithm that explicitly considers both owner and partner Agents' discipline when determining whether a detection event has occurred. High discipline Agents are more likely to detect and less likely to be detected and vice versa. Figure 3 shows a high discipline recon team – note that the blue Agents are well dispersed and are waiting at the recon checkpoint. Figure 4 shows the low discipline team, note that they are not as well dispersed, they will also spend less time at the recon checkpoint.

5. Conclusion

Designed using principles taken from the sciences of Complexity, Archimedes is a flexible and powerful combat simulation platform. Archimedes employs software Aspects, a novel extension to the method of ABM, to address the issues of nonlinearity and intangibles that are of interest to the modern military analyst. More generally, the BASP modeling framework provides a general approach for simulating complex systems that is both powerful and accessible to the analyst and programmer alike.

6. Acknowledgments

This work was supported under contract with the Marine Corps Combat Development Command's Project Albert, Contract No. M 00027-96-D-0012. The authors wish to thank Capt. M. Leonardi and Mr. A. Sawyers for useful conversations. The authors would also like to thank Project Albert, in particular Dr. A. Brandstein and Dr. G. Horne for the technical vision that has allowed this project to be realized.