CROCADILE - AN OPEN, EXTENSIBLE AGENT-BASED DISTILLATION ENGINE

Michael BARLOW and Adam EASTON

Introduction

Simulation continues to grow as a vital tool for modern military forces. As an example there are approximately 70 simulations that are used throughout the Australian Army alone.[1] These include Live, Virtual and Constructive simulations.

There is a diverse range of applications that simulation is applied to within the military. These include individual training, strategic planning, decision support, capability and force structure development, mission rehearsal, and operational analysis.

Constructive Simulation has been used in all of the above roles. Constructive Simulation can be defined as computer models that represent the actions of people and/or equipment.[2] Currently it is the most commonly used form of simulation within the Australian Army.

In order to assist in their application of simulations the Australian Army has proposed a hierarchy of simulation.[3] This hierarchy defines a layered approach to simulation where scenarios are first examined with less detailed models which are quick and easy to use. More detailed models then further examine the insights gained from these models. This process continues until finally the most detailed simulations can be used with a better understanding of what the critical parameters for a given scenario are. A conceptual diagram of this hierarchy is shown in Figure 1.

At the very top of this hierarchy are what can be considered 'conceptual level simulations.'[4] These are simulations that do not directly model physical entities on the battlefield such as tanks or soldiers; instead they model generic capabilities on the battlefield. This reduces the need to gather data on specific battlefield entities. This in turn allows a more rapid development of scenarios.

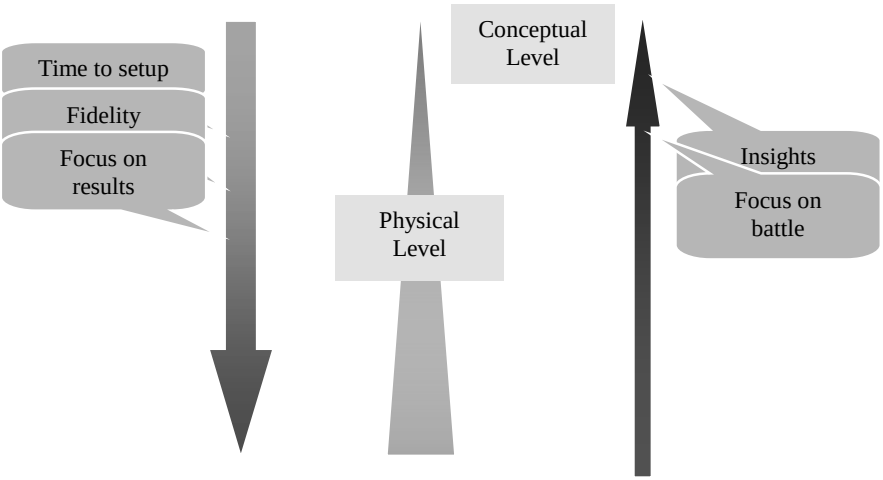

Figure 1: A diagrammatic representation of the Australian Army simulation hierarchy.

While these simulations obviously do not produce results that can be directly applied to the real battlefield, they do provide an insight into what capabilities appear to be effective against the enemy and which do not.[5] However, a limitation of these systems is that they have minimal ability to explore and further develop the insights that they provide. Instead other simulations must be used. This transition is not smooth and unfortunately much of the value of the insights gained can be lost in the transition from the conceptual to the more realistic level.

The move towards a more holistic approach to combat simulation has not been confined to the Australian military, nor has it been confined to a linear progression from lower fidelity models to higher fidelity models. A key example of this is the work being conducted under the Project Albert [6] banner into a concept termed Operational Synthesis.

Project Albert was initiated by the United States Marine Corp with the mission of addressing three key weaknesses in existing simulation models.[7] The following features were not adequately addressed:

- non-linearity of combat – how small changes in given factors can produce much larger changes in the combat result;

- co-evolving landscapes – how forces are able to adapt to each others tactics and alter their own course of action accordingly; and
- the effect of intangibles – factors such as morale, training, aggression and fear.

All three of these concepts are extremely difficult to model with traditional constructive simulations that rely heavily on mathematical formulas and a black and white rule set. As a result, Project Albert is using Operational Synthesis in an attempt to address these concepts.

Operational Synthesis involves the concurrent use of a range of simulation tools in order to gain a more complete understanding of a combat scenario.[8] It recognises that all simulation tools have strengths and weaknesses, and attempts to use each tool in a capacity that draws from its strengths to mitigate the weaknesses of the other tools.

Within its operational synthesis framework, Project Albert has examined the use of a variety of tools allowing the use of human in the loop and automated systems, stochastic and deterministic models as well as a varying degree of fidelity.[9] Broadly, however, the tools that it has used can be divided into four categories. These are:

- War games;
- Deterministic equations;
- Simulations;
- Distillations.

Of these, the tools known as distillations have received the most attention by Project Albert members.[10] They represent an emerging area in the combat modelling body of knowledge and have shown great promise in providing analysts with insights into areas such as non-linearity, co-evolving landscapes and intangibles where other tools have been unable to provide meaningful results.

Perhaps the oldest form of combat simulation is through the technique of war gaming. This process involves utilising personnel, and their experience to brainstorm scenarios and develop courses of action. War-gaming in its primitive form can be traced back as early as 490 BC and has played a significant role in military planning through the history of conflict.[11] More recently computers have been integrated into the war-gaming process; however, the strength of war gaming still relies on the contributions of personnel involved in the process.[12]

During the early 20th century a more mathematically based approach toward combat simulation began to emerge. This occurred through the development of formulas and equations which attempted to model combat. These equations were predominantly attrition based and embraced the concept that the losses by one side in a conflict

would be mathematically related to the size and strength of the opposition force. Perhaps the best-known pair of equations for combat simulation were the Lanchester Equations.[13]

The Lanchester Equations are a set of coupled differential equations that were first developed in 1914 by F.W. Lanchester. Similar to the predator/prey models, they describe a linear battle where the casualties of one side are dependent on the size and strength of the other.[14] These equations, while perhaps overly simplistic, were reasonably effective during early twentieth century warfare such as the trench warfare and artillery duels of World War One.[15] As the century progressed, however, warfare became a less linear relationship.

As the twentieth century progressed, the increasing power of computers paved the way for the development of computer based constructive simulation. Traditionally these computer based constructive simulations have been based upon mathematical attrition modes like the Lanchester Equations or derivations thereof.[16] In addition to this they have provided a way to integrate prescriptive rules and specific detail into models facilitating more detailed scenarios and consequently, the ability to investigate quite complex combat situations. These computer-based simulations are currently the most widely used tool for conducting combat simulation.

While these simulations are rich in detail and high in fidelity, they have several drawbacks. Probably the main one of these is that they are completely prescriptive. That is, behaviours of elements within the simulation are stringently specified. This makes it difficult to model all possible outcomes because of the level of detail and time required to set up an individual scenario run.

A second disadvantage of traditional simulations is that, to date, these simulations have been strongly equipment and firepower centric.[17] This is largely a result of their founding in attrition-based equations such as the Lanchester Equation. This basis means that all simulations, at their lowest levels, resolve combat as a mathematical relationship based on a purely attritionist model. As the nature of warfare changes, this is becoming an increasingly significant limitation.

The end of the twentieth century saw the emergence of Manoeuvre Warfare. This represented a fundamental shift in the way that warfare was fought. The Australian Army's definition of Manoeuvre Warfare is the application of combat power to defeat the enemy's will to fight.[18] It refers to the principle of a force applying its strength to an enemy's weak points so as to cause a disproportionate amount of damage to the enemy.

With this change in the nature of warfare, it can be argued that the Lanchester Equations and traditional constructive simulation techniques no longer provide as

accurate a result as they have in the past. This in turn leads to the question of whether a paradigm exists which would better model modern warfare. Significant research has been conducted in this area over the last decade, and arising from it has been a strong argument that combat may perhaps best be modelled as a complex adaptive system.

Complexity is a concept that is strongly coupled with chaos. While chaos can be considered an investigation of how simple microscopic behaviours produce a complicated macroscopic behaviour, complexity is the investigation of how complex microscopic behaviours can produce simpler macroscopic behaviours.[19] A good example of complexity can be seen in the output of cellular automata models.[20]

Complex Adaptive Systems (CAS) can be described as systems composed of many nonlinearly interacting parts that continually adapt by changing their internal rules as both their environment and knowledge of that environment evolve over time.[21] Within these systems, while the system parts act independently based on localized rules, an overall macroscopic behaviour emerges which appears to have some sort of natural coordination.

Further examination of these CAS, has allowed the following list of key properties to be developed:[22]

- non-linear interaction;
- hierarchical structure;
- decentralised control;
- non-equilibrium order;
- adaptation.

When comparing each of these to combat, similarities begin to emerge – the non-linearity due to environment, equipment, and doctrine; the hierarchical rank and command structure; decentralised control due to the fact that individuals compose a force and modern commanders are given scope in achieving their goals; plus non-equilibrium and adaptation as fundamental aspects of combat. The presence of all of the abovementioned factors lends evidence to support the hypothesis that combat is indeed a CAS.

This hypothesis is also mathematically supported. Analysis of cellular automata models and other CAS have shown them to produce fractal distributions in comparison to traditional constructive simulations which do not. The fact that studies into real combat data have shown the presence of fractal distributions in historical combat again suggests that combat may indeed be a complex adaptive system.[23]

If this hypothesis is accepted it gives some direction in answering the question of what new paradigm should be used to model modern combat. If, as complexity

suggests, the fundamental building blocks of a system are its sub-parts, a reasonable approach would seem to be modelling individual entities on the battlefield rather than attempting to develop mathematical models that approximate the outcome of force on force scenarios that contain many entities.

Agent-Based Simulations are systems in which the infrastructure of the simulation and the entities that participate within the simulation are logically separated.[24] This allows the agents to be easily added, removed or modified within a simulation.

Agent-based paradigms are a relatively new technology within the simulation domain. They emerged as an expansion from work on Cellular Automata [25] and initially focussed on the simulation of primitive insect colonies.[26] Multi-Agent systems have been used in a plethora of applications. These include modelling of nations, economic factors, businesses and neurons.

The military applications of agent-based technology are a recent development. The agents within these simulations range from simple agents that follow a basic set of rules, to highly detailed models with complex knowledge bases and rule sets. Their purposes are diverse including agents that control battlefield entities, command agents that act as high level commanders, air traffic controllers, information filtering agents and decision support agents.[27]

Broadly speaking, work on agent-based simulation has been in one of five areas. The first of these has been in creating more complex agent architectures. This work has been tightly coupled with work into creating higher fidelity simulation engines. As the complexity of the worlds in which the agents operate increase, it is necessary to increase the complexity of the agents' architecture so that they can relate to the world at this higher level of fidelity.

A somewhat related field of work to this has been the creation of team-based agent architectures. The majority of agent-based simulations currently model agents in relative isolation from each other. It can be argued, however, that as the military environment is fundamentally a team one, the inclusion for a team model in agent-based combat scenarios would be beneficial.[28] This argument has lead to work being conducted into team-based agent architectures that allow control over formations, agent connectivity and path planning.

Another area of work on agent-based simulations has been into integrating agent-based systems with distributed simulations. A principle component of this approach is the generation of Semi-Autonomous Forces. This is an inherently agent-based requirement that will require agents to be able to operate intelligently in a broad suite of applications.

Human-in-the loop simulation is another area that is being integrated with agent-based simulations. Through including humans within the simulation process, the effectiveness of the agents involved has been able to be better tested and consequently improved.[29]

The final, and most recent, development of agent-based technology has been in the development of simple agent-based distillations.[30] These distillations take a different approach to agent modelling, moving away from the traditional high fidelity approach towards a more compact agent architecture where the focus is on the interactions between agents and not so much on the agents themselves. This has allowed combat to be modelled in new ways and is providing promising results in representing combat as a complex adaptive system.

Distillation Systems

Broadly speaking, distillations can be defined as simulations that attempt to model warfare scenarios by implementing a small set of rules that allow agents to adapt within each scenario. Distillations represent a far closer modelling of CAS than any other constructive simulation paradigm.[31] Furthermore, distillation systems can be shown to be relevant to the conceptual end of the simulation hierarchy.

Distillations are far less detailed than traditional simulations and rely on sensible global behaviour to emerge naturally, unlike traditional models that require this behaviour to be explicitly programmed. This simplicity gives distillations the characteristics of speed, transparency, ease of configuration and the ability to use the systems with minimal training.[32]

Unlike the traditional firepower and equipment centric simulations, distillations can be considered to be manoeuvre centric.[33] This means that insights are predominantly gained not through the numerical results of simulation runs, but rather from an understanding of how the agents adapt to each other's tactics.

Agent-based distillations provide a bottom-up approach to modelling combat scenarios.[34] Unlike traditional constructive simulations which specify an overall scenario, and then layer more and more levels of detail as they generate the components of that scenario, distillations require the analyst to develop the individual components and then observe the overall behaviour that emerges within the model.

There are currently several agent-based distillations that have contributed to this field of knowledge. The most commonly employed are: Irreducible Semi-Autonomous Agent Combat (ISAAC), Enhanced ISAAC Neural Simulation Toolkit (EINSTein), Map Aware Non-uniform Automata (MANA), and Socrates.

All of these systems contain common characteristics. These include a two-dimensional world, a kill probability combat resolution algorithm which is used to determine combat outcomes at the lowest level, and an attractor / repulsor agent control paradigm.[35] This attractor / repulsor method can be described as a set of weights that determine what direction, and how far, an agent will move at any given time. It can perhaps best be likened to the spring-embedder method of multi-dimensional distance resolution described by Battista.[36] While all extant combat distillations possess these properties, they are not necessarily a property of agent distillations in general. Rather they represent the design decisions that have been made within extant systems.

ISAAC was the first Multi-Agent System to be developed which treated combat as a complex system. It was developed as a proof of concept model. In this system, as with all of the models described below, agent behaviour is not pre-programmed. Instead, agents are given a set of instincts such as aggression, self-preservation and attraction to friendly forces. These instincts are weighed off against each other to determine a course of action for that agent at a given point in time.[37]

As the name "Enhanced ISAAC Neural Simulation Toolkit" suggests, EINSTein expanded upon the ISAAC model, retaining much of the ISAAC engine but adding additional visualisation tools, simulation log utilities as well as meta-personalities [38] for the agents. These meta-personalities allowed agents to change their behaviour when they were in a group of agents of a sufficient size. Therefore an agent acting on its own could be made to be less aggressive than an agent collocated with 20 other friendly agents.[39]

MANA is a system developed in New Zealand that further expands upon the concepts validated by ISAAC and EINSTein. Unlike these systems, however, it attempts to model squad level scenarios as opposed to the strategic large-scale scenarios modelled by the other systems. Another key factor in MANA is the introduction of Agent memory. This allows agents to build a picture of the world as the simulation progresses. Consequently an agent's actions are not based solely on the situation it is currently faced with, but rather the current situation plus the situations it has faced in the past.[40]

Socrates is the most recently developed agent-based distillation. Fairly similar to the systems described above, Socrates implements the instinctive based attractor / repulsor agent control paradigm, a 2D world and a probability based physics model. In comparison to the other models, however, Socrates provides a reasonably rich set of agent capabilities and behaviours.[41]

There is little doubt that the behaviours exhibited within the above systems often bear a resemblance to behaviour that we would intuitively expect on the battlefield.[42]

Furthermore, while they take some time to learn, such systems are quick to use. Through the use of these systems a commander or analyst is able to gain understanding of the overall shape of a battle and what factors are more important than others in determining the outcome of a battle.

Despite these advantages, the question arises as to how effective these systems are in the larger simulation framework and whether they have attained the full potential that a distillation-based approach promises. This aspect of their role in the simulation hierarchy is currently problematic: the very abstraction that makes them an ideal tool for rapid exploration of the combat parameter space makes it difficult to link them and the insights they provide to higher fidelity simulation models. The difference in detail of representation is often too large to extrapolate from the distillation domain into the domain of higher fidelity simulations.

Secondly, current distillation systems have built-in a number of assumptions that serve as hard constraints on the battlespace domains that may be explored. These constraints, often tied to the origins of the systems in cellular automata, include amongst others the binding of agent behaviour and agent capability together, limitation to a flat 2D world, a single paradigm of behaviour for agents – the spring-embedder approach of weighted vector addition, and the lack of important modern military capabilities such as indirect fire, blast weapons, and round penetration. The constraints not only limit the range of scenarios that can be explored but also can significantly weaken the strength of the insights the distillations provide. For instance, how might an insight about the impact of a sensor overmatch on scenario outcome be altered by taking place in a 3D environment of hills and valleys, that impact detection ability, rather than in a flat plain? Alternatively, in what way would the possession of blast weapons that affect an area alter the insights about manoeuvre tactics, in particular unit clustering and dispersion?

CROCADILE - Design of a New Distillation

In an attempt to address the perceived issues with distillation systems, a new system CROCACDILE—Conceptual Research Oriented Combat Agent Distillation Implemented in the Littoral Environment—has been designed and implemented.

The following subsections identify the design goals and key features of CROCADILE, discuss the Object Oriented design of the system, its major components in the form of the simulation engine and the agent interface, before concluding with the issues faced and overcome in the area of computational tractability and complexity management.

Design Goals

The previous section describing distillation systems identified two issues that CROCADILE was designed to address. Firstly, that the abstraction of the current distillations is both a strength and a weakness. Secondly, current systems also have a number of in-built limitations that hinder their generality and applicability to represent certain aspects of modern conflicts.

The means that CROCADILE employs to address these issues is generality and multi-fidelity resolution through a clear Object Oriented (OO) design. CROCADILE is designed as an open and expandable distillation engine. Not only does it support multiple levels of fidelity and conflict resolution—2D or 3D world, probabilistic or projectile-physics hit resolution—but it is also expandable by the user. Users can write their own agents and add them to the worlds simulated. Additionally, the various objects that compose the simulation can be extended to incorporate new types of functionality. This is a key lesson from observing the current distillation systems in usage. It is impossible to envisage all possible future applications of the system. A versatile and extensible core simulation engine and components are key to longevity and wide applicability.

CROCADILE follows a very strong object oriented design in all aspects. Its internal design and implementation is object oriented and, being written in the Java language, it will run on any hardware/OS platform. That OO design extends through to the simulation and the world itself that is seen by both the agents and the user. Agent behaviour is separated from agent capability, with capability being defined by individual weapon, sensor, communication, and movement objects with which that agent is equipped. Similarly, the world and the items that occupy it—agents, munitions, terrain features, objectives, etc.—are all objects with their own properties and capabilities.

This clear OO design, combined with the goal of not imposing any unnecessary constraints, has led to a number of key features in CROCADILE:

- 3D or 2D environment in which the agents interact;
- Probabilistic or Projectile-Physics combat resolution;
- Movement by Air and Water, as well as by Land;
- User extensible agent behaviours allowing users to code different control paradigms;
- Sophisticated Command, Mission, and Communication structures for agents;
- Higher fidelity combat resolution models that incorporate blast effects, round penetration, rates of fire, and line-of-sight;
- Database of world objects—agents, agent groups, behaviours, weapons,

sensors, etc.—that can be saved individually and reused in subsequent scenario building;

- Comprehensive result logging including time-line and individual event information. Analysis possible via a visualisation tool part of the system, or commercial spreadsheet;
- Multi-team structure including neutrals and levels of enmity/alliance and communication between teams.

CROCADILE realises a 3D environment. This includes a location in 3D space for all physical objects such as agents and munitions. Further, digital terrain is supported and can be imported into any scenario meaning that agents can exist in a world of synthetic or actual (drawn from some part of the Earth's surface) terrain. Agents exist on that 3D landscape and their actions may be modified by that same landscape. Terrain affects movement, line-of-sight issues such as sensor detection, and hit resolution - the flight of projectiles and blast effects. Agents are also aware of that terrain, if employing their sensors, meaning that it becomes part of their decision process. Terrain features—water, vegetation, obstacles, and objectives—may also have a shape and location prescribed, once again enriching the environment in which the agents exist. However, simply by not incorporating these elements, such as digital terrain, in a particular scenario it is possible to simplify the simulation to a 2D world. Indeed, it is possible to start from a 2D scenario and progressively layer-in additional 3D aspects – enriching the environment and gauging how that modifies scenario outcomes. Scenario runs that are being visualised present a top-down view of the world in which terrain height is coded in greyscale. Figure 2 is a screenshot of a scenario taking place on terrain in south-west New South Wales, Australia.

Traditionally, distillation systems have employed a probabilistic model for hit resolution. Each weapon has certain chance of hitting any target regardless of factors such as target size, distance to target, or the terrain. CROCADILE supports this model but also incorporates a projectile-physics model. Munitions are fired with given speed and headings and 3D collision detection is used to detect when agents are hit or where explosions occur. Factors such as target size, speed, and distance away, together with the terrain itself become an important aspect of whether individual shots hit or not. Scenario runs that are being visualised show the projectiles. Figure 2 is a screen-capture of a CROCADILE run – munitions can be seen as the small black dots.

CROCADILE supports not only land-based movement but also air and sea. Movement capabilities can be defined to function in these domains also. Providing an agent with that capability then equips the agent to travel in that type of environment. This facilitates 'joint operations' style scenarios.

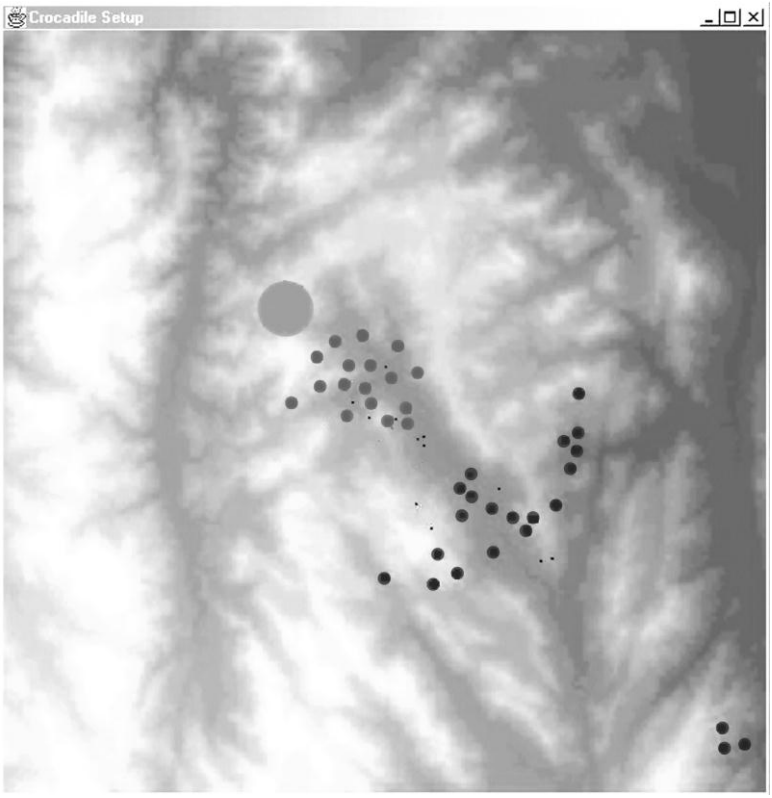

Figure 2: A screen-capture of CROCADILE while running. The conflict between two opposing forces occurs atop a 3D terrain coloured in greyscale - darker colours are lower. Agent munitions are also seen as the small black dots, while the large circle is an explosion.

One means of viewing CROCADILE is as an agent test-harness – a means of contrasting different types of agents. CROCADILE defines an interface for its agents – their means of acting within a CROCADILE scenario. CROCADILE can load and run any agent which conforms with that interface. While CROCADILE incorporates an instinctual agent with behaviour resolution via weighted vector addition similar to that of other distillations, it is entirely possible for a user to write their own agent and add that to any CROCADILE scenario. Thus Belief, Desire and Intention (BDI), learning, or user controlled agents can all be added to CROCADILE.

Recognising that the complexities of agent interaction provide the emergent behaviour of distillation systems, CROCADILE provides a rich set of command,

mission, and communication structures. Hierarchies of command and communication can be established between groups of agents. Agents can issue orders to fulfil a mission to their subordinates, while subordinates have a propensity, or lack there of, to follow orders. Similarly, missions include not only the destruction of enemy agents but features of the environment itself, reaching a goal or destroying a static feature.

CROCADILE provides for a higher fidelity, though still abstracted, hit resolution than other distillation systems. Each agent has a health score that is reduced by the damage of the round that hit it. Each agent's initial health may be set differently. Further, each agent has a 7-point adjectival armour scale. Weapons are similarly rated for penetration. Only if the damage rating of a round that hits an agent exceeds the armour rating of that agent, will the round cause damage. This, for instance, can be used to stop small arms fire destroying heavily armoured vehicles in a scenario. Different weapons may have different rates of fire and number of rounds with which they are equipped. Terrain plays a part in hit resolution for explosions. Only if there is LoS (Line-of-Sight) between the centre of the blast area and the agent inside that area will the agent potentially suffer damage. If there is no LoS, then the agent is sheltered from the blast by the terrain.

CROCADILE provides the ability to save, as separate items, the individual components that form a scenario. Users may save terrain, agents, agent groups, agent behaviours, weapons, movement capabilities, sensors, command structures, and communication structures individually. These libraries of scenario components can then be employed in the rapid creation of new scenarios.

A rich range of data logging is built into CROCADILE. Each run results in a set of logs output to files. These logs include information about the state of the scenario at each timeframe – number of agents and health of each team. Further, they include a record of each significant event in the game. Each time an agent is hit, an objective hit or entered, or an agent is destroyed, the particulars of the event are recorded. These include the location, agent(s) and team(s) involved, damage etc. Logs are output as comma-separated-values, making them compatible with spreadsheet applications. CROCADILE also includes a data visualisation tool, allowing users to view various aspects of the run. CROCADILE can be run in an interactive or batch-mode thus facilitating extensive analysis of a scenario.

Finally, CROCADILE can support scenarios with any number of different teams involved. Relationships between pairs of teams can be friendly, neutral, or enemy allowing more complex situations to be designed. Communications for agents can be configured to occur at several levels, including whether to share information with friendly teams or not.

Major Components of the System

The CROCADILE design is split into two logically distinct sections. The first of these is the simulation itself, consisting of the simulation engine, specification of the world, agents, agents' capabilities, world objects and how they all inter-relate. The second of these is the instinctual agent control paradigm, which specifies how the agents behave within the world.

Both of these sub-parts can be further broken down to examine the major functionality groups within them. The simulation component of the system is by far the larger of the two sub-components. It is responsible for specifying all of the aspects of the system except for how the agents within the system 'think.' This cognitive simulation is carried out by the agent behaviour sub-component.

The overarching principle in the conceptual design of CROCADILE is to keep the two levels of separation as strong as possible. The first is the logical separation between the agents and the rest of the world, and the second is the separation between agent capabilities and agent behaviour.

Both of these divisions are largely established through the development of a set of classes that act as interfaces between the logically separated components. These classes are the agent's Capability Manager, Information Manager and Status Monitor. The agent control components of CROCADILE cannot access the agent that they are controlling directly, but can only gain information or affect this agent and the broader world through using these interfaces. Broadly speaking, the Capability Manager is responsible for allowing an agent to affect the world around it. The Information Manager is responsible for allowing an agent to gain knowledge on the world around it, and the Status Monitor is responsible for allowing an agent to gain knowledge about its own condition and status.

A conceptual representation of the CROCADILE distillation is provided in Figure 3. This diagram shows the major functionality elements within the system and how they relate to one another in greater detail than depicted in the logical diagrams included as Figures 4 and 5. The arrows within Figure 3 represent the directions of data flow within the system, and the labels describe the nature of the relationships that the major functional components have.

Because of the complexity and size of these two system components, it is necessary to break each one down into its major functionalities in order to better understand how the system works.

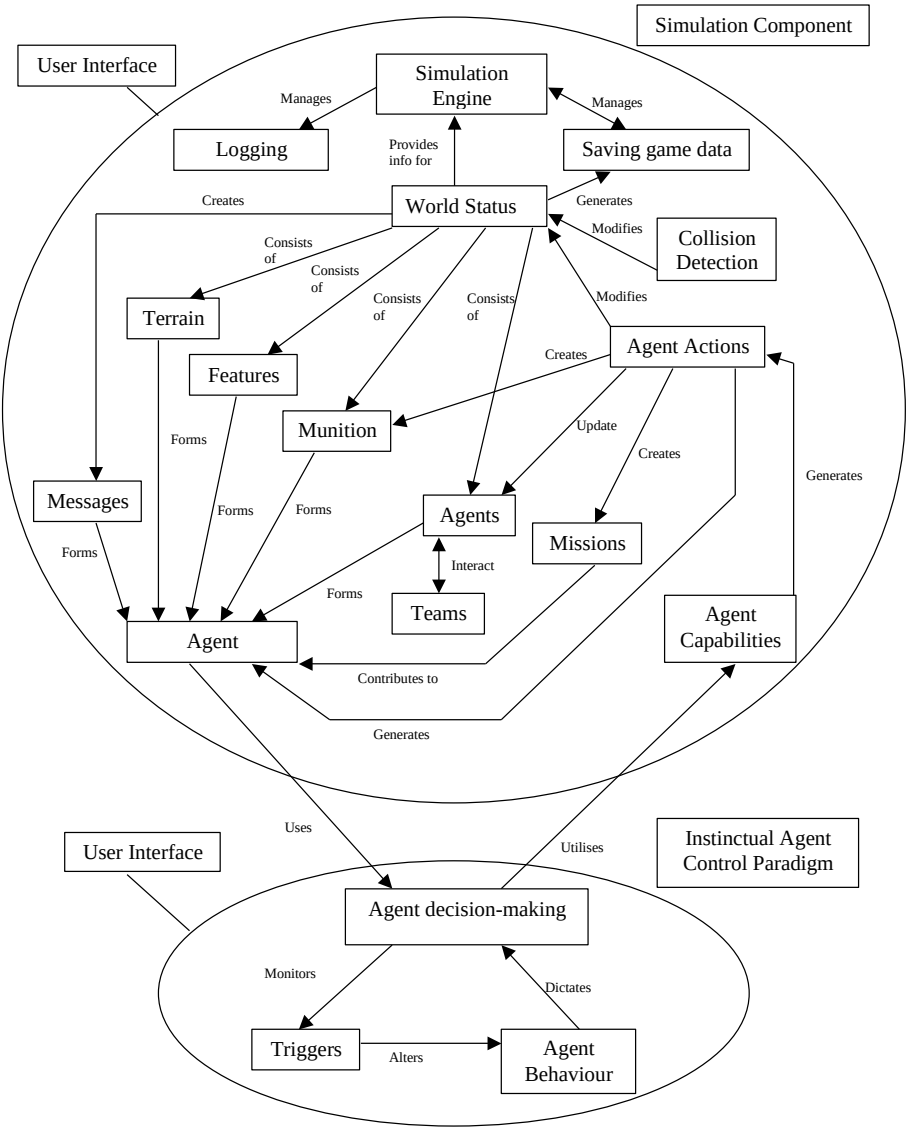

Figure 3: A conceptual representation of the CROACDILE distillation illustrating the major functional classes and their relationships.

The Simulation Engine

The simulation component of CROCADILE is the core of the Agent Distillation. It is the component of the system that manages how objects interact, the status of all objects in the world, what data is recorded about a scenario, generally how the simulation runs, and when it stops.

The object that is central to the simulation component is the World. It represents all of the rules that relate to the physical interaction of objects in the world as well as storing all of the objects present within it. The major elements that are present within the world are munitions such as explosions and weapon rounds, physical features such as vegetation and obstacles, the 3D terrain landscape, and the agents themselves. The physical interactions between all of these objects are resolved by the world's collision detection elements.

The World itself has no knowledge of the concept of a simulation run, which the simulation engine manages. This engine sits above the world and utilises the World's functionality to run the simulation. The simulation engine is responsible for managing the simulation sequence, recording data about the simulation run, and determining the success or stopping conditions of a simulated scenario.

The simulation component of CROCADILE also consists of a suite of library classes. These classes are responsible for saving components of the world such as agent capabilities, whole agents, other world objects or indeed, an entire world scenario.

Finally, the simulation component consists of a user interface component that is responsible for both displaying and allowing the user to modify the details that relate to the simulation or simulation scenario.

Figure 4 shows the major logical elements of the simulation component of CROCADILE, along with lines to represent relationships between these logical elements.

The simulation engine is the core of CROCADILE. It contains the main simulation loop that controls the sequence in which the simulation is run. This core functionality is contained within the Simulator class. The Simulator sits on top of the world and manages when and how it applies its rules to the elements within it.

In addition to simply managing the simulation sequence, the simulation engine is also responsible for performing several key tasks. The first of these is the management of the agent behaviours that are used within the simulation. As any class that extends the AgentBrainClass and uses the correct methods to interact with the world, it can be used to control agents within CROCADILE. However, it is necessary to protect the rest of the system against poorly written or malicious agents that are being used within the simulation. The simulation engine manages this protection.

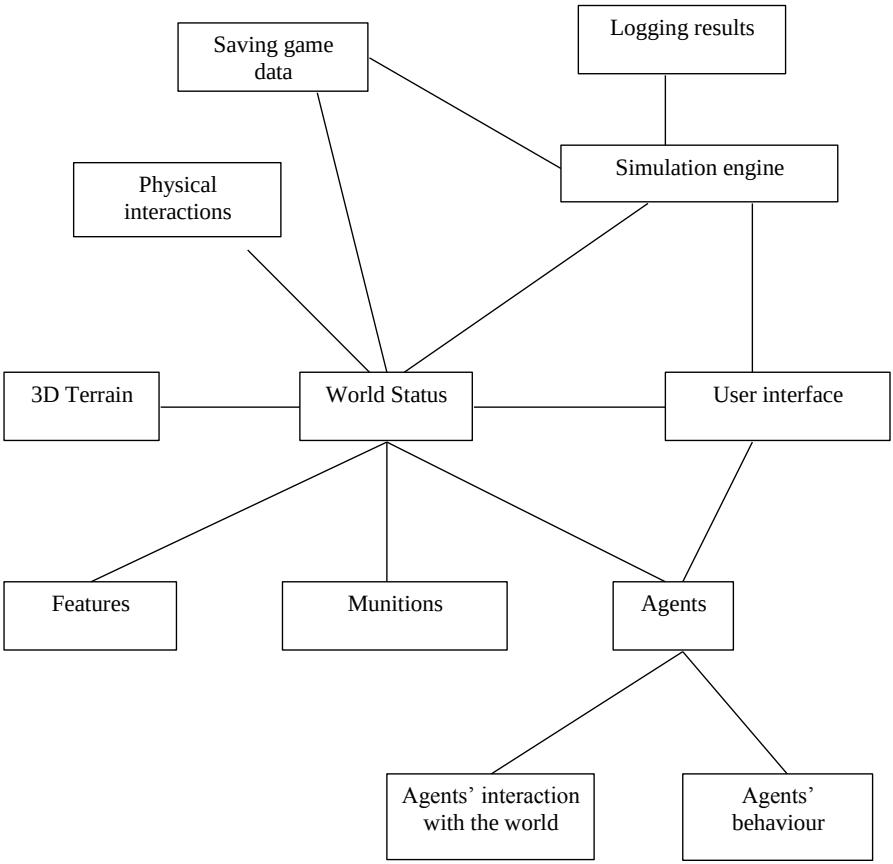

Figure 4: The major logical elements of the CROCADILE distillation.

Another important attribute of the simulation is its ability to specify the time step used for a simulation run. Rather than updating the simulation for each time unit, it can be specified exactly how many times per time unit the world is updated, or conversely, how many time units should pass between each update of the world. When this value is changed, all corresponding velocities, forces and time factors are scaled accordingly. This allows a balance to be struck between the accuracy of the simulation and the time that it takes to run.

Other functions performed by the simulation engine are determining the stop conditions for a given simulation run, managing random number generation, and determining what hit resolution method is to be used for the given simulation run.

Agents and Agent Capabilities

The agent control component of CROCADILE specifies how the agents within the simulation behave, and can be seamlessly integrated with CROCADILE's simulation component. As described previously, any paradigm for controlling the behaviour of the agents can be used, providing that its inputs and outputs match those specified by the simulation component.

The default instinctual paradigm that is implemented incorporates a set of behavioural triggers which facilitate agent meta-personalities. The paradigm consists of four main elements. The first is the behaviour element that stores the set of weights that dictate how an agent will behave. The triggers within the paradigm are responsible for altering this behaviour according to the situation that the agent is currently in. The user through the user-interface element sets up both these triggers, and the behaviours, before run time.

The final element of this agent control component is the Agent decision-making element. This element contains no configurable local data but rather holds the methods that translate the behavioural weights stored within the behaviour element into a specified course of action. The decision-making element is also responsible for activating the trigger tests at appropriate times.

Agents in their capacity as cognitive entities interact with the world and each other in many ways other than through the process of collision detection that characterises the interaction between physical objects. A class diagram showing some of this interaction is found in Figure 6.

Agents are created as being part of an agent family and a team. Agents are able to identify whether other agents are part of their agent family and whether they are part of their team. This allows agents to react differently to other agents depending on their relationship to them.

Every agent within the world has a reference to an Agent Brain. This is the class that dictates how that agent should behave. This agent brain cannot access the agent itself but is rather given access to a set of tools, namely the Capability Manager, Information Manager and Status Monitor. Using these tools, Agent Brains are able to control the physical agent that they belong to, and interact with the world around them. Figure 5 shows the relationship of the AgentBrain class to other elements of the system.

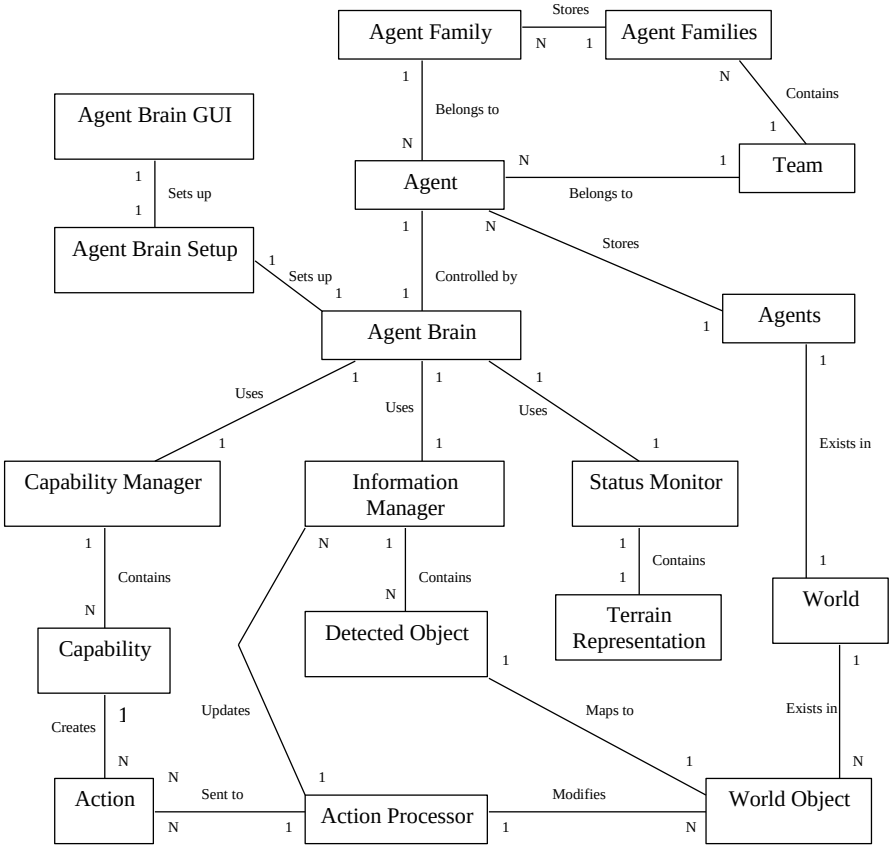

Figure 5: Relationship of the AgentBrain class to other aspects of the CROCADILE system.

As is the case with real life, perception often does not match reality. The fact that an agent observed another object several moments ago does not mean that its knowledge of that object automatically updates as the object changes. Rather, that detection was a snapshot of that object and will not be updated unless the object is detected again. CROCADILE caters for this divergence between perception and reality through creating a parallel set of classes for all physical objects within the world. These objects are termed DetectedObjects. The hierarchy of these detected objects can be seen in Figure 6.

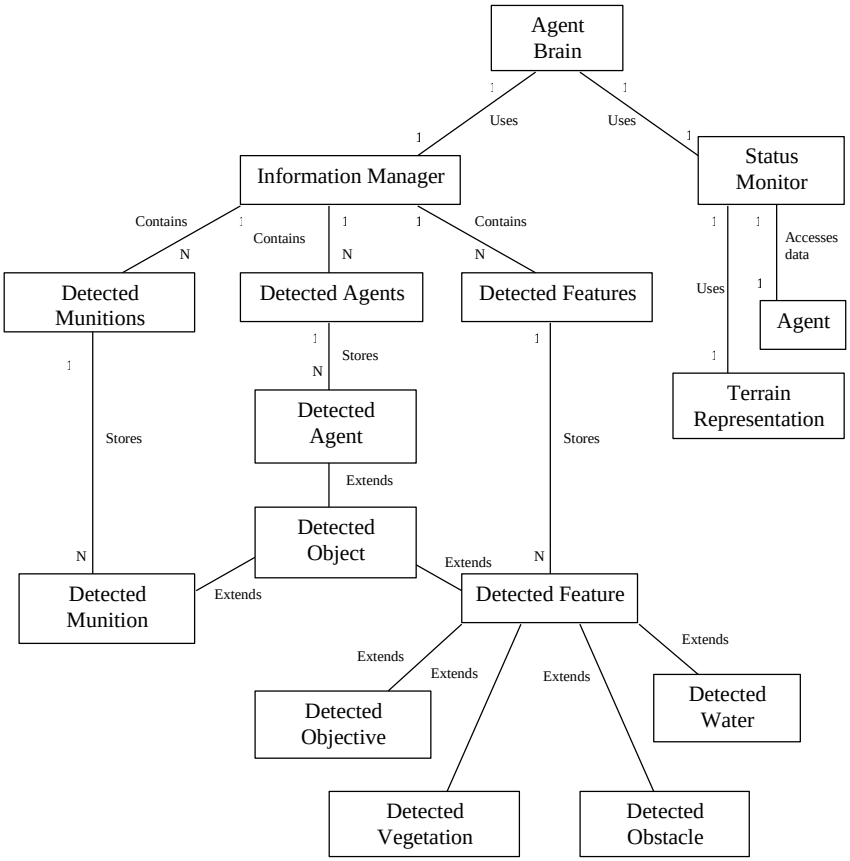

Figure 6: Knowledge representation classes employed by agents in CROCADILE.

As an agent becomes aware of objects within the world, a detected object is created from the real object. This detected object is then sent to the agent. The information contained within this detected object may be varied depending on the level of detail that the agent was able to determine from the physical object. This level of detail is a function of distance and can be assigned to each sensor during the set-up phase.

There are two fundamental aspects of an agent's knowledge. The first is the knowledge of the world around it and the second is of the agent itself. CROCADILE provides an agent brain with the tools to process both forms of information. These two tools are the Information Manager, which handles an agent's understanding of the broader world; and the Status Monitor, which allows an agent to gain information on its own status.

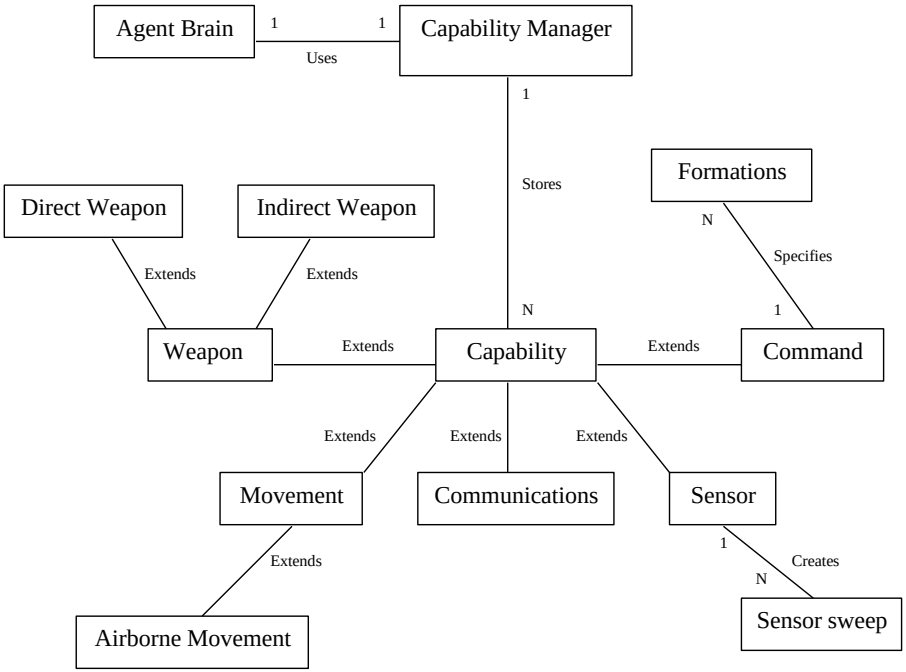

Figure 7: Capability management classes employed by agents.

The Information Manager is probably the most complex of these two tools. It is essentially a data store of all Missions, Messages and Detected Objects that an agent is aware of within the world at that given time. Of these three entities, the Detected Objects represent the largest volume of data that is processed by the Information Manager. The structure of the Detected Objects class hierarchy and their corresponding relationship to the Information Manager is shown within Figure 6.

As well as understanding the world, agents need a way to affect it in some capacity. They are able to do this through the use of the Capability Manager tool. This tool is a central repository for all of the capabilities owned by an agent. Agents are not restricted to one capability of any type. The Capability Manager contains a constant that specifies the maximum number of capabilities of a given type that an agent may posses, agents may have any number up to and including the value of this constant. Figure 7 shows a diagram of the classes related to capabilities and capability management in CROCADILE.

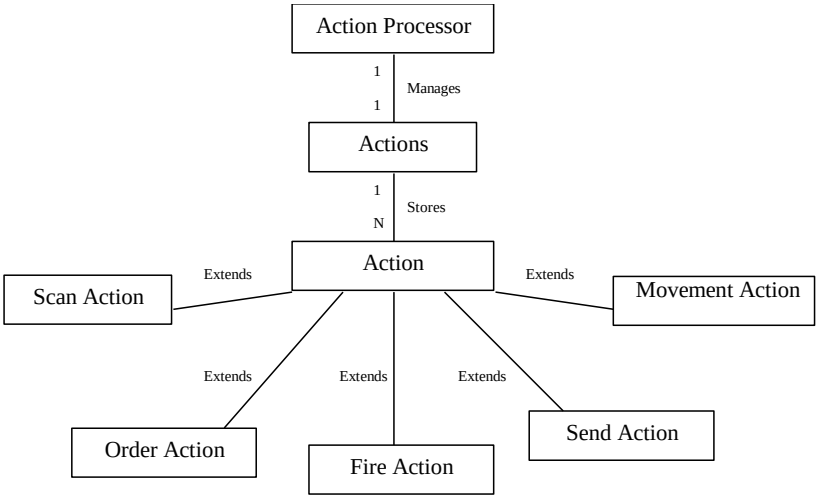

Figure 8: The action class hierarchy employed by CROCADILE.

At this stage, it is necessary to describe how capabilities interact with the world. When a capability is exercised, the effect on the world is not immediate. Instead, an object called an Action is created. This action is added to a central world store of Actions called the Action Processor. Once all of the agents within the world have completed their turn, the simulation engine activates the Action Processor. This loops through all of the actions that have been created and transforms them into effects on the world. Through this process it can be ensured that the world does not change between agents' turns and consequently, that the order that the agents think in (their turn sequence) has minimal impact on the simulation.

Every capability has an associated action. The Weapon class creates a fire action, the Sensor class creates a scan action, the Movement class creates a Movement action, the Communications class creates a send action and the Command class creates an order action. The class diagram of these actions and how they relate is shown in Figure 8.

Instinctual Agent

The three principal elements of the instinctual agent are the trigger mechanisms, the Instinctual agent itself and the behaviour templates that it uses. These elements and their related classes are shown in Figure 9.

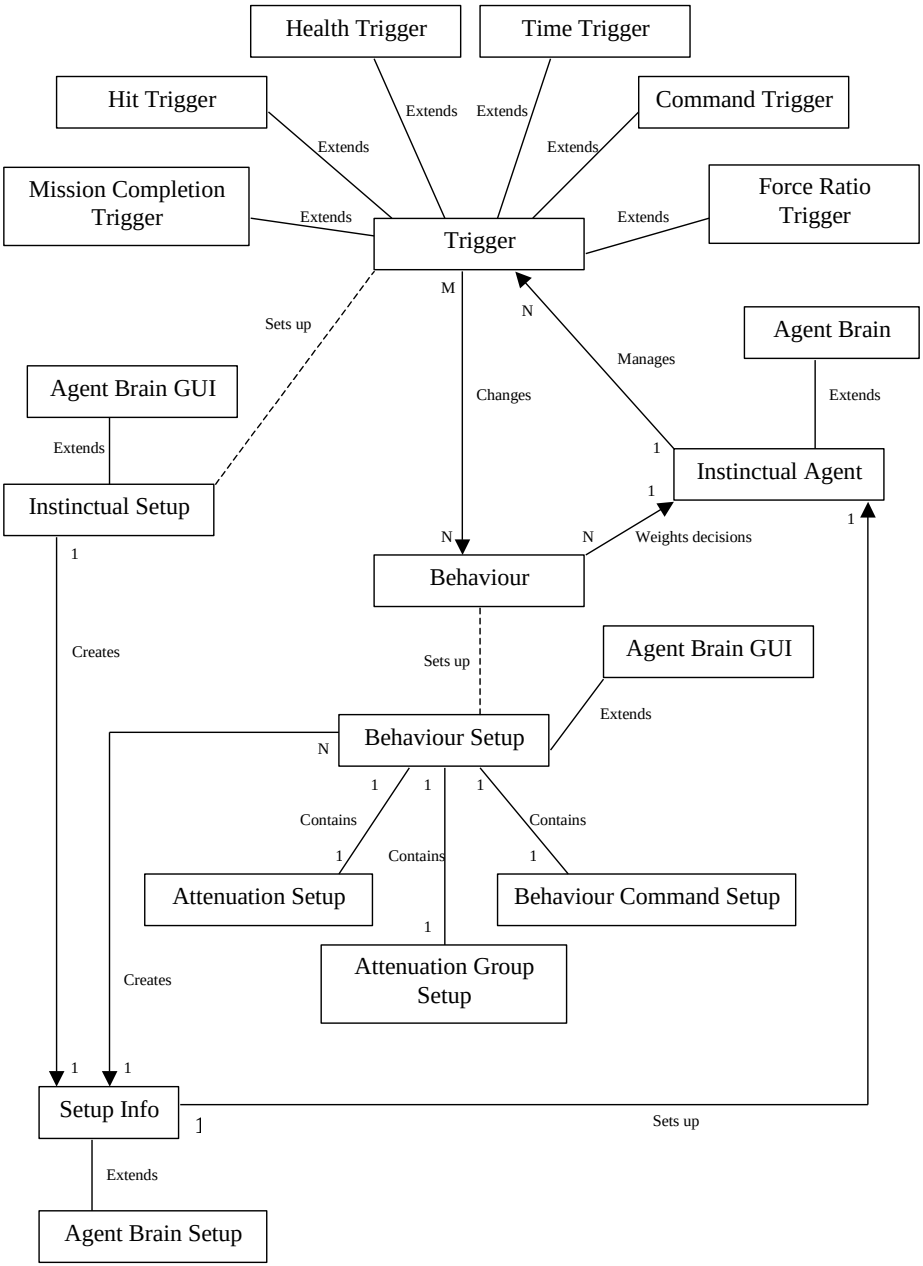

Figure 9: The instinctual agent class and mechanism. The instinctual agent is the default agent behaviour supplied with CROCADILE.

The Instinctual Agent class is the core of the instinctual agent. It is this class that extends the Agent Brain class. It is responsible for determining the course of action that will be adopted at a given time. The information that it uses to make this decision is the data that it gains from the Information Manager and Status Monitor, and the relevant set of behavioural weights. These behavioural weights are stored in the Behaviour class.

The trigger mechanism allows these behavioural weights to be changed depending on the situation that an agent is immersed in. It is again the Instinctual Agent that manages how and when these triggers are checked. When a trigger fires, the currently selected collection of behavioural weights is swapped with the new set. This requires no change on behalf of the instinctual agent that continues using the new set of weights, unaware that a change has even occurred.

In addition to the core functionalities of the instinctual agent paradigm, Figure 9 also shows the mechanism used for setting up an instinctual agent. This happens through the InstinctualSetup class that allows a definition of the triggers within the world. From this class, dialogs can be invoked which allow individual behaviour weights to be specified.

Computational Efficiency and Usability Issues

CROCADILE presents a far more complex system than previous distillations due to a number of its features. That complexity presents a particular challenge in the areas of computation and usability. With its 3D physics model, continuous domain, and more involved hit resolution the computation requirements are potentially far higher than previous distillations. However real-time, interactive runs are a key feature in the usability of distillations. Similarly the range of additional functionalities afforded by CROCADILE could potentially overwhelm a user, again eliminating that ease-of-use that characterises distillations.

Considerable effort was spent in the design and implementation of CROCADILE to minimise the impact of these elements. The result is a distillation that runs in real-time even on older desktop PCs (e.g., Pentium II) for a projectile-physics resolved scenario involving over 60-agents on 3D terrain. Similarly, time for scenario design is of the same order as other distillations for similar complexity of scenario.

In terms of computational load, the collision detection required for a 3D world with projectile-physics combat resolution is a great burden. Every object – agents and munitions – must be checked for a collision with every other object and the terrain, every round of the simulation. Several strategies were implemented to ameliorate this computational load. Level-of-Detail (LoD) modelling was employed for the terrain at four separate levels with collision detection occurring at the minimum level of detail

necessary to resolve the situation. Similarly, terrain features such as water or vegetation are defined in CROCADILE by arbitrarily complex polygons. In order to simplify checks for agents entering such features, bounding spheres around the features were employed as a first level of check. Finally, and most significantly, CROCADILE employs a tiling mechanism in order to reduce the number of redundant checks. The world is subdivided into a number of tiles with checks for collisions between objects only being made for objects, which occupy the same tile. For an `n x n` tiling scheme that reduces inter-object collision checks by a factor of n^2 (for an evenly spread group of objects).

The issue of complexity of usage is addressed in two ways by CROCADILE. Firstly, a hierarchical user-interface with sensible default values for a number of parameters is provided as a shield from the potential complexity. Users may then navigate the higher-level UI and employ the default set of values. Secondly, CROCADILE provides for a database of world objects – agents, weapons, agent groups, sensors, etc. This enables users to compile libraries of world objects. These objects can then be employed for the rapid creation of new scenarios.

A Future Land Force Scenario in CROCADILE

This section illustrates some of the features of CROCADILE by detailing the design of a scenario and the subsequent analysis of the results. A scenario representing one possible structure under the Australian Army's "Army After Next" concept is built. The batch processing facilities of CROCADILE were utilised to run the scenario 100 times, keeping a log of all results. Finally, those results were briefly analysed and are presented below.

It is worth noting that this is not an analysis of potential force structures – for that the parameter space of sensor, weapon, communication, and mobility would need to be explored, along with the relative worth of the three different force elements. Rather the following section presents possible analyses performed at one point in that multi-dimensional space.

The Scenario

The test case chosen was based on an examination of the force structure that could be used to combat a traditional armoured style battle group. The scenario consisted of two sides. The red side resembled a traditional tank group, characterised by high firepower, relatively slow mobility and an average sensor range. This group consisted of 40 agents that exhibited these characteristics.

These red agents were attempting to move from the top left of the world to the bottom right. Furthermore, they were aggressive and would attack and advance toward any

enemy that they sensed. In contrast, the blue side consisted of a three-tier force structure based on the Army After Next model. It had 15 agents which formed its recon element, 5 agents which formed its strike element and 3 agents which formed its reach back capability.

The recon elements exhibited poor firepower but moderate movement and good sensor capabilities. Their mission was to keep any detected red agents in sensor range without moving into the weapon range of the red agent. They also had a desire to stay dispersed in order to cover a larger area. The strike element possessed a highly damaging, but short-range firepower capability, rapid movement and a limited sensor capability. These strike assets were intended to stay out of range of the enemy and then attack the enemy when vulnerability was detected.

Finally the reach-back element was static with a long-range area effect firepower capability, and limited sensor range. All elements of the blue force structure were able to communicate with each other. The agents that formed the reach-back element were intended to engage the enemy as soon as it was detected.

The scenario was framed as an encounter between Australian forces and an Enemy that was moving southerly from Sydney towards Canberra. Consequently, real world terrain data for this area was used in CROCADILE.

Terrain

Digital terrain data was obtained of the region of NSW between Canberra and Sydney. This was scaled into a 500x500 grid of points on which the simulation runs took place. Figure 10 shows one projection of that terrain.

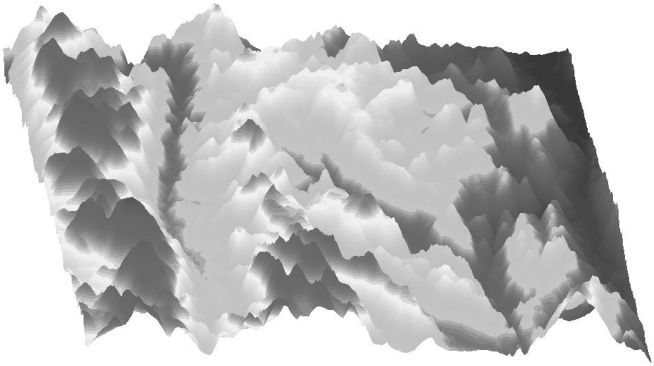

Figure 10: The 3D terrain on which the scenario runs took place. Red agents start the scenario in the top-left, while the static elements of blue begin in the bottom-right.

Agent Behaviours & Capabilities

Table 1 summarises the agents and their capabilities that form each side. Red is a homogenous single group of 40 agents, while blue is a heterogeneous group of three sub-groups – the most numerous light reconnaissance units, the fast strike units, and the long-range reach-back units with area effect weapons.

Table 1: Agent properties, capabilities, and behaviours as used in the example scenario of a conventional red force against a heterogeneous blue force.

		Red	**Blue**		
		Armour	**Recon.**	**Strike**	**Reach Back**
Physical	**Number**	40	15	5	3
	Health	100	100	100	100
	Initial Location	Top-Left	Middle	Bottom-right	Bottom-right
Capability	**Sensor Range**	250	250	250	250
	Comms Range	100	500	500	500
	Move Speed	4	4	10	0
	Weapon Range	150	120	150	special
	Weapon Damage	15	10	50	45
	Damage Type	kinetic	kinetic	kinetic	explosive (18 radius)
	Fire Rate	1/4	1/4	1/6	1/7
	Munition Rounds	100	100	20	6
Behaviour		Aggressive - close with blue, maintain friendly spacing	Passive - Keep at sensor range from enemy, maintain spacing, close with wounded enemy	Controlled aggression - Keep at weapon range from enemy	Fire at leading elements of enemy

Like all distillations the particular values are unitless, it is only their relative strength or weakness within the scenario that has meaning. Hence weapon damages range from 10 to 50 points – these points have no units and they have meaning only when considered relative to the health statistic of agents: 100. The reach back units of blue do not have a weapon range. Rather the weapon's muzzle velocity of 150 units/round is used in combination with the firing angle to calculate its munitions' parabolic path and, hence, where it strikes the terrain.

Analysis

As mentioned previously, one hundred runs of CROCADILE were made for the scenario as described above. This was done in order to capture the range of variability that is one of the features of distillation systems. The log-files were processed in order to analyse the scenario outcomes. For the purposes of this illustrative example the mean across was employed as a means of summarising the results of the one hundred runs.

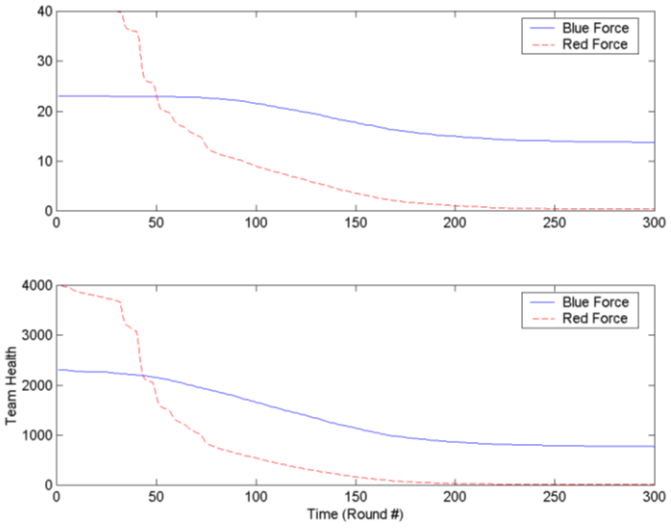

Figure 11: Number of red and blue agents alive (top) and total team health (bottom) as functions of time within the scenario. The plots represent the mean of 100 individual runs.

Figure 11 shows the number of agents alive, and total team health as a function of time within each simulation run. Despite starting the scenario with superior numbers

and superior health (both nearly 2:1), in the average case red is decimated, while blue fairs far better.

Red appears to suffer a large number of casualties and damage between rounds 40 and 75. Observing scenario runs it was noted that this appeared due to the indirect, explosive rounds of blue's "reach back" group. In order to check this hypothesis damage in the scenario was subdivided into explosive—fired by the reach-back units—and impact – fired by all other units and plotted as a function of time. Figure 12 is the result.

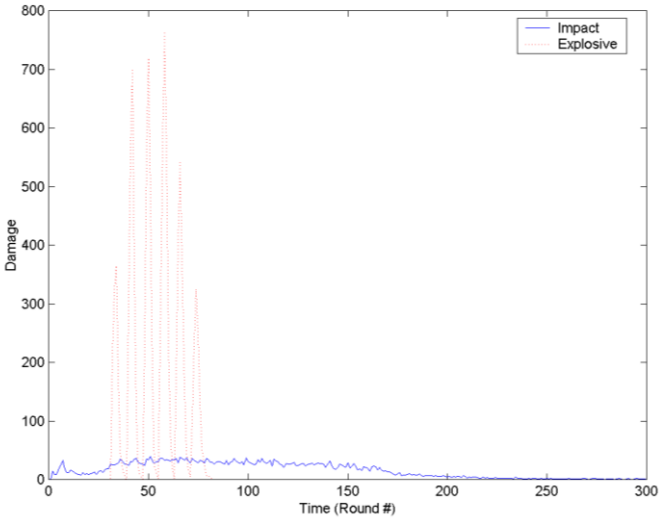

Figure 12: Damage within the scenario runs as a function of the time (round number) within the scenario. Damage is subdivided into impact (ballistic) and explosive rounds.

As is clearly seen from the figure, impact damage is spread rather evenly throughout the scenario, tailing off towards the end as the number of agents becomes small. On the other hand, explosive damage all occurs between roughly round 40 and 80. This is indeed the period when blue's indirect rounds fall amongst the red agents. Indeed those rounds are shown to be particularly devastating, causing very large amounts of damage. Observation of the log files showed that this was the result of the burst effect of the rounds – a single round might damage as many as five red agents. The explosive damage itself is confined to a relatively short time span. This is attributable to the limited number of rounds possessed by the reach back units. A final

observation from the figure is the periodicity of the explosive damage – this is due to the fact that the reach-back units were constrained to only firing once every seven rounds.

As CROCADILE keeps a log of all major simulation events, such as every round that hits – that includes where the event took place, it is possible to spatially analyse the events of the simulation run.

Figures 13 and 14 provide such an analysis. Both are contour plots – showing the regions where hits occurred, where damage was taken, and where agents were destroyed. Figure 13 groups all, red and blue, agents together; while Figure 14 separates the red and blue agents, placing the respective representation side by side in two columns.

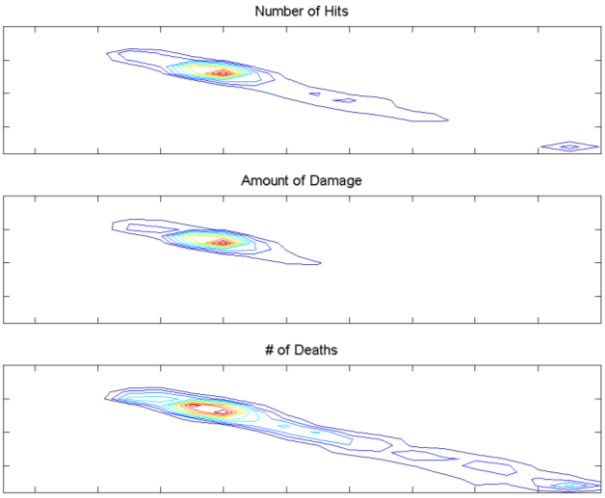

Figure 13: Distribution of hits, damage, and agent deaths across the physical landscape. Results are grouped across all agents, regardless of team. Red agents begin the scenario in the top-left of the world while blue's position is the bottom-right.

Several observations are possible from the figures. Firstly, just as red follows a simple approach of chasing blue units, while most blue retreat or are fixed in the bottom-right corner; so the course of the conflict is distributed along the diagonal from top-left to bottom-right. The non-linearity inserted by the reach-back units of blue means that most hits and damage occur where the munitions from those units fall. Because of munition flight-times and the delay in communication of information

from the reconnaissance units to the reach-back units, most hits remain roughly at the same location across the runs – approximately one-third of the way from red's starting position to blue's static position.

Contrasting between the two forces, as shown by Figure 14, it is once again clear that the red force suffers most of its damage and hits from the indirect explosive fire of the blue reach-back units that occurs in the first third of red's advance. On the other hand, the blue force suffers most of its hits, damage and casualties in the "second half" of the battle as red approaches blue's defensive position – blue looses room to manoeuvre, its stationary units come within reach of the advancing red units, and a number of blue units become more aggressive (the "fast strike" units have entered the conflict while the recon units become aggressive in response to wounded red units).

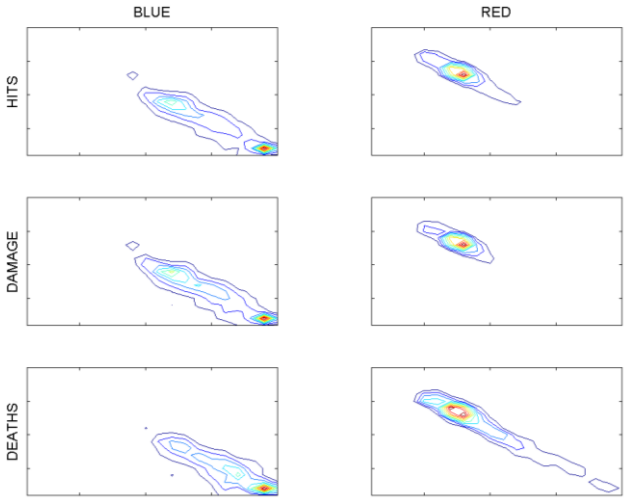

Figure 14: Distribution of hits, damage, and agent deaths across the physical landscape. Results are subdivided on the basis of team – blue in the left column and red in the right. Red agents begin the scenario in the top-left of the world while blue's position is the bottom-right.

Discussion

This paper has presented a new multi-agent-based combat distillation known as CROCADILE—Conceptual Research Oriented Combat Agent Distillation Implemented in the Littoral Environment—that has been developed at the Australian Defence Force Academy. The system is not intended to replace existing distillations

such as MANA, EINSTein or Socrates; rather it complements them by building on their core features while incorporating several key new aspects. These key aspects have been previously summarised but lie in two areas. Higher fidelity models in the form of a 3D world, including 3D terrain; probabilistic or projectile hit resolution; more complex combat phenomenon – round penetration, continuous health, burst effects and indirect fire; and the ability to load user-written agents are all supported by the system. The conventional 2D world and instinctual agent behaviour is also supported, allowing users to select the level of fidelity they desire. This moves the realm of distillations closer to conventional constructive simulations, arguably making it easier to transfer insights gained in the distillation realm into more detailed simulations and thus supporting the operational synthesis, or holistic approach to simulation. Secondly, CROCADILE delivers an open, extensible simulation engine that can be extended simply and efficiently. This is directly attributable to the strong object-oriented approach that permeates all levels of CROCADILE. Just as it is possible for the user to design a new sensor with which to equip an agent, so also is it possible to write code for a new agent, or even extend the definition of the Communication capability within the simulation engine in order to support fallibility of communication equipment.

CROCADILE is a free system and its usage by the distillation, operation analysis, Alife and agent communities is welcomed. The web site http://www.cs.adfa.edu.au/VESL/Croc contains not only the latest version of CROCADILE but also supporting materials such as a user manual, more detailed technical information, together with a growing database of terrain files and pre-built scenarios, agents, and equipment.

CROCADILE is not a static system. While it is already being used to investigate agents that learn tactics, teamwork within a heterogeneous group of agents, the impact of asynchronous updates within a network of relationships, and the importance of intangibles such as morale or personality on conflict outcome; these are only a fraction of its potential applications. At the same time CROCADILE has a developmental future – a full 3D visualisation of running scenarios, alternate agent behaviour paradigms such as BDI, high-level real-time control of agents by a human, and a centralised database accessible over the network through CROCADILE itself are all planned. Resource availability will dictate how quickly these and other planned features are realised.

Notes:

1 Trevor Colton, "The Army Synthetic Environment," in *SimTecT Proceedings 2001* (Canberra, Australia, 2001), pp. 305-308.

2 Simon Mepham, "Synthetic Environments – Delivering Real Benefits to UK Defence," *SimTecT Proceedings 1998* (Adelaide, Australia, 1998).

3 Trevor Colton, Private communication (2001).

4 Conceptual level simulations model conflict at an abstract level, modelling generic capabilities and effects rather than specific entities and weapons.

5 Andy Illachinski, *Towards a Science of Experimental Complexity: An Artificial Life Approach to Modelling Warfare*, Research Memorandum CRM 99-61 (Center for Naval Analyses, 1999).

6 The United States Marine Corps established project Albert in 1995 with the mission of examining new sciences to provide quantitative answers to important military questions. Since then Project Albert has become an international project with participants from Australia, New Zealand, Sweden, Germany and Canada all actively involved with research in this domain.

7 Gary Horne, "Beyond Point Estimates: Operational Synthesis and Data Farming," in *Maneuver Warfare Science 2001*, ed. Gary Horne and Mary Leonardi (US Marine Corps, 2001).

8 Alfred Brandstein, "Operational Synthesis: Applying Science to Military Science," *Phalanx* 32, 4 (1999): 1, 30-31.

9 Horne, "Beyond Point Estimates: Operational Synthesis and Data Farming."

10 Alfred Brandstein, *Introduction to Project Albert*, Briefing slides to 4th Project Albert International Workshop (2001).

11 J. Clavell, *The Art of War By Sun Tzu* (London, England: Hodder and Stoughton, 1981).

12 Alfred Brandstein, "Foreword," in *Maneuver Warfare Science 2001*, ed. Gary Horne and Mary Leonardi (US Marine Corps, 2001).

13 F. W. Lanchester, *Aircraft in Warfare* (London, England: Constable & Co, 1916).

14 Taylor thereof conducted a detailed mathematical analysis of the Lanchester Equations and derivations in 1983. – J. G. Taylor, *Lanchester Models of Warfare* (USA, Operations Research Society of America, 1983).

15 Andy Illachinski, *Irreducible Semi-Autonomous Adaptive Combat (ISAAC): An Artificial Life Approach to Land Warfare*, Research Memorandum CRM 97-61 (Center for Naval Analyses, 1997).

16 Illachinski, *Irreducible Semi-Autonomous Adaptive Combat*.

17 Michael Lauren, *Complexity Theory and Land Warfare*, Briefing slides for the 4th Project Albert International Workshop (2001).

18 Land Warfare Doctrine 1: *Fundamentals of Land Warfare* (Canberra, Australia: Defence Publishing Service, 2000).

19 Andy Illachinski, *Land Warfare and Complexity, Part 1: Mathematical Background and Technical Sourcebook*, Information Manual CIM-461 (Alexandria, VA: Center for Naval

Analyses, 1996); Andy Illachinski, *Land Warfare and Complexity, Part 2: An Assessment of the Applicability of Nonlinear Dynamics and Complex Systems Theory to the Study of Land Warfare*, Research Memorandum CRM-68 (Alexandria, VA: Center for Naval Analyses, 1996).

[20] Michael Lauren, *Characterising the Difference between Complex Adaptive and Conventional Combat Models* (Auckland, New Zealand: Defence Operational Technology Support Establishment, 1999).

[21] Illachinski, *Land Warfare and Complexity.*

[22] Illachinski, *Land Warfare and Complexity.*

[23] Michael Lauren, *Beyond Lanchester: A Fractal-Based Approach to Equations of Attrition* (Auckland, New Zealand: Defence Technology Agency, 2001).

[24] Stan Franklin and Art Graesser, "Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents," *Third International Workshop on Agents, Theories, Architectures and Languages ATAL* (1996).

[25] A Cellular Automata is a regular spatial lattice of cells, where each cell may have any one of a finite number of states at a given time step, and where the state of each cell is updated according to a local rule which may depend on the state of the cell and its neighbors at the previous time step.

[26] Nino Boccara, O. Roblin and M. Roger, "Automata network predator-prey model with pursuit and evasion," *Physical Review E* 50, 6 (1994): 4531-4541.

[27] Gil Tidhar C. Heinze, Simon Goss, G. Murray, D. Appla, and I. Lloyd, *Using Intelligent Agents in Military Simulation or "Using Agents Intelligently"* (Australia: Defence Science and Technology Organisation, 2000).

[28] Andrew Lucas, *et.al.*, "Towards Complex Team Behaviour in Multi-Agent Systems," *SimTecT Proceedings 2001* (Canberra, Australia, 2001), 89-92.

[29] Kerry Bennett, T. Josefsson, Simon Goss, M. Cross, S. Waugh, and T. Truong, "An Application of DSTO's Battle Model using Agents and Humans-in-the Loop," *SimTecT Proceedings 2001* (Canberra, Australia, 2001), 99-110.

[30] Lauren, *Complexity Theory and Land Warfare.*

[31] Brandstein, "Operational Synthesis: Applying Science to Military Science."

[32] Horne, "Beyond Point Estimates: Operational Synthesis and Data Farming."

[33] Lauren, *Beyond Lanchester: A Fractal-Based Approach to Equations of Attrition.*

[34] Lauren, *Characterising the Difference between Complex Adaptive and Conventional Combat Models.*

[35] An agent control paradigm is the algorithm or technique that is used to control an agent's behaviour.

[36] G. Battista, *et.al.*, "Algorithms for Drawing Graphs: An Annotated Bibliography," *Computer Geometry and Theory Application* 4 (1994): 235-282.

[37] Illachinski, *Irreducible Semi-Autonomous Adaptive Combat (ISAAC): An Artificial Life Approach to Land Warfare.*

[38] Meta-personalities refer to the ability of an agent to exhibit alternative behaviours depending upon the situation that it is immersed in at a given time.

[39] Illachinski, *Towards a Science of Experimental Complexity.*

[40] R. Stephen, *Maui Agent-Based Combat Model*, Briefing slides, Project Albert 3rd International Workshop (Auckland, New Zealand, 2001).

[41] N. Bent, *Socrates v1.1 User Manual* (USA: Emergent Information Technologies Inc., 2001).

[42] Illachinski, *Towards a Science of Experimental Complexity.*

Dr. MICHAEL BARLOW is a senior lecturer within the School of Computer Science at the University of New South Wales, at the Australian Defence Force Academy (ADFA). He is also the founding director of the Virtual Environments and Simulation Laboratory (VESL) at ADFA. Dr. Barlow has published over thirty papers in the areas of speech and speaker recognition, visualisation, virtual environments, agents and cellular automata. He is also co-author of a book covering the media APIs of Java to be published by Sam's Publishing in mid 2002. Dr. Barlow's interests cover agent technologies, automatic speech understanding and speaker recognition systems, virtual environments and visualisation, and educational technology. Mail Address: Dr. Michael Barlow, School of Computer Science, University of NSW / ADFA, Northcott Drive ACT 2600, Australia. Email: spike@adfa.edu.au.

ADAM EASTON is a lieutenant in the Australian Army. Adam Easton is a new researcher with one prior publication. Adam Easton's interests cover agents, graphics, and simulation engine technologies.